

Anne Auger · Carlos M. Fonseca  
Nuno Lourenço · Penousal Machado  
Luís Paquete · Darrell Whitley (Eds.)

LNCS 11101

# Parallel Problem Solving from Nature – PPSN XV

15th International Conference  
Coimbra, Portugal, September 8–12, 2018  
Proceedings, Part I

1  
Part I



 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, Lancaster, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Zurich, Switzerland*

John C. Mitchell

*Stanford University, Stanford, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

C. Pandu Rangan

*Indian Institute of Technology Madras, Chennai, India*

Bernhard Steffen

*TU Dortmund University, Dortmund, 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, Saarbrücken, Germany*



More information about this series at <http://www.springer.com/series/7407>

Anne Auger · Carlos M. Fonseca  
Nuno Lourenço · Penousal Machado  
Luís Paquete · Darrell Whitley (Eds.)

# Parallel Problem Solving from Nature – PPSN XV

15th International Conference  
Coimbra, Portugal, September 8–12, 2018  
Proceedings, Part I

*Editors*

Anne Auger  
Inria Saclay  
Palaiseau  
France

Penousal Machado  
University of Coimbra  
Coimbra  
Portugal

Carlos M. Fonseca  
University of Coimbra  
Coimbra  
Portugal

Luís Paquete  
University of Coimbra  
Coimbra  
Portugal

Nuno Lourenço  
University of Coimbra  
Coimbra  
Portugal

Darrell Whitley  
Colorado State University  
Fort Collins, CO  
USA

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-319-99252-5              ISBN 978-3-319-99253-2 (eBook)  
<https://doi.org/10.1007/978-3-319-99253-2>

Library of Congress Control Number: 2018951432

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer Nature Switzerland AG 2018

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.

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.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

During September 8–12, 2018, researchers from all over the world gathered in Coimbra, Portugal, for the 15th International Conference on Parallel Problem Solving from Nature (PPSN XV). Far more than a European event, this biennial meeting has established itself among the most important and highly respected international conferences in nature-inspired computation worldwide since its first edition in Dortmund in 1990. These two LNCS volumes contain the proceedings of the conference.

We received 205 submissions from 44 countries. An extensive review process involved over 200 reviewers, who evaluated and reported on the manuscripts. All papers were assigned to at least three Program Committee members for review. A total of 745 review reports were received, or over 3.6 reviews on average per manuscript. All review reports were analyzed in detail by the Program Chairs. Where there was disagreement among reviewers, the Program Chairs also evaluated the papers themselves. In some cases, discussion among reviewers with conflicting reviews was promoted with the aim of making as accurate and fair a decision as possible. Overall, 79 manuscripts were selected for presentation and inclusion in the proceedings, which represents an acceptance rate just below 38.6%. This makes PPSN 2018 the most selective PPSN conference of the past 12 years, and reinforces its position as a major, high-quality evolutionary computation scientific event.

The meeting began with an extensive program of 23 tutorials and six workshops covering a wide range of topics in evolutionary computation and related areas, including machine learning, statistics, and mathematical programming. Tutorials offered participants the opportunity to learn more about well-established, as well as more recent, research, while workshops provided a friendly environment where new ideas could be presented and discussed by participants with similar interests.

In addition, three distinguished invited speakers delivered keynote addresses at the conference. Ahmed Elgammal (Rutgers University, USA), Francis Heylighen (Vrije Universiteit Brussel, Belgium), and Kurt Mehlhorn (Max Planck Institute for Informatics, Saarbrücken, Germany) spoke on advances in the area of artificial intelligence and art, foundational concepts and mechanisms that underlie parallel problem solving in nature, and models of computation by living organisms, respectively.

We thank the authors of all submitted manuscripts, and express our appreciation to all the members of the Program Committee and external reviewers who provided thorough evaluations of those submissions. We thank the keynote speakers, tutorial speakers, and workshop organizers for significantly enriching the scientific program with their participation. To all members of the Organizing Committee and local organizers, we extend our deep gratitude for their dedication in preparing and running the conference. Special thanks are due to the University of Coimbra for hosting the conference and, in particular, to INESC Coimbra, CISUC, the Department of Informatics Engineering, the Department of Mathematics, and the International Relations Unit, for their invaluable contribution to the organization of this event, and to the

sponsoring institutions for their generosity. Finally, we wish to personally thank Carlos Henggeler Antunes for his unconditional support.

September 2018

Anne Auger  
Carlos M. Fonseca  
Nuno Lourenço  
Penousal Machado  
Luís Paquete  
Darrell Whitley

# Organization

PPSN 2018 was organized by INESC Coimbra and CISUC, and was hosted by the University of Coimbra, Portugal. Established in 1290, the University of Coimbra is the oldest university in the country and among the oldest in the world. It is a UNESCO World Heritage site since 2013.

## Organizing Committee

### General Chairs

Carlos M. Fonseca	University of Coimbra, Portugal
Penousal Machado	University of Coimbra, Portugal

### Honorary Chair

Hans-Paul Schwefel	TU Dortmund University, Germany
--------------------	---------------------------------

### Program Chairs

Anne Auger	Inria Saclay, France
Luís Paquete	University of Coimbra, Portugal
Darrell Whitley	Colorado State University, USA

### Workshop Chairs

Robin C. Purshouse	University of Sheffield, UK
Christine Zarges	Aberystwyth University, UK

### Tutorial Chairs

Michael T. M. Emmerich	Leiden University, The Netherlands
Gisele L. Pappa	Federal University of Minas Gerais, Brazil

### Publications Chair

Nuno Lourenço	University of Coimbra, Portugal
---------------	---------------------------------

### Local Organization Chair

Pedro Martins	University of Coimbra, Portugal
---------------	---------------------------------

### Webmasters

Catarina Maçãs	University of Coimbra, Portugal
Evgheni Polisciuc	University of Coimbra, Portugal

## Steering Committee

David W. Corne	Heriot-Watt University Edinburgh, UK
Carlos Cotta	Universidad de Malaga, Spain
Kenneth De Jong	George Mason University, USA
Agoston E. Eiben	Vrije Universiteit Amsterdam, The Netherlands
Bogdan Filipič	Jožef Stefan Institute, Slovenia
Emma Hart	Edinburgh Napier University, UK
Juan Julián Merelo Guervós	Universidad de Granada, Spain
Günter Rudolph	TU Dortmund University, Germany
Thomas P. Runarsson	University of Iceland, Iceland
Robert Schaefer	University of Krakow, Poland
Marc Schoenauer	Inria, France
Xin Yao	University of Birmingham, UK

## Keynote Speakers

Ahmed Elgammal	Rutgers University, USA
Francis Heylighen	Vrije Universiteit Brussel, Belgium
Kurt Mehlhorn	Max Planck Institute for Informatics, Germany

## Program Committee

Youhei Akimoto	Shinshu University, Japan
Richard Allmendinger	University of Manchester, UK
Dirk Arnold	Dalhousie University, Canada
Asma Atamna	Inria, France
Anne Auger	Inria, France
Dogan Aydin	Dumlupinar University, Turkey
Jaume Bacardit	Newcastle University, UK
Helio Barbosa	Laboratório Nacional de Computação Científica, Brasil
Thomas Bartz-Beielstein	Cologne University of Applied Sciences, Germany
Heder Bernardino	Universidade Federal de Juiz de Fora, Brasil
Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria
Mauro Birattari	Université Libre de Bruxelles, Belgium
Christian Blum	Spanish National Research Council, Spain
Peter Bosman	Centrum Wiskunde & Informatica, The Netherlands
Pascal Bouvry	University of Luxembourg, Luxembourg
Juergen Branke	University of Warwick, UK
Dimo Brockhoff	Inria and Ecole Polytechnique, France
Will Browne	Victoria University of Wellington, New Zealand
Alexander Brownlee	University of Stirling, Scotland
Larry Bull	University of the West of England, England
Arina Buzdalova	ITMO University, Russia
Maxim Buzdalov	ITMO University, Russia
Stefano Cagnoni	University of Parma, Italy
David Cairns	University of Stirling, Scotland

Mauro Castelli	Universidade Nova de Lisboa, Portugal
Wenxiang Chen	Colorado State University, USA
Ying-Ping Chen	National Chiao Tung University, Taiwan
Marco Chiarandini	University of Southern Denmark, Denmark
Francisco Chicano	University of Málaga, Spain
Miroslav Chlebik	University of Sussex, UK
Sung-Bae Cho	Yonsei University, South Korea
Alexandre Chotard	Inria, France
Carlos Coello Coello	CINVESTAV-IPN, Mexico
Dogan Corus	University of Nottingham, UK
Ernesto Costa	University of Coimbra, Portugal
Carlos Cotta	University of Málaga, Spain
Kenneth De Jong	George Mason University, USA
Antonio Della Cioppa	University of Salerno, Italy
Bilel Derbel	University of Lille, France
Benjamin Doerr	École Polytechnique, France
Carola Doerr	Sorbonne University, Paris, France
Marco Dorigo	Université Libre de Bruxelles, Belgium
Johann Dréo	Thales Research & Technology, France
Rafal Drezewski	AGH University of Science and Technology, Poland
Michael Emmerich	Leiden University, The Netherlands
Andries Engelbrecht	University of Pretoria, South Africa
Anton Eremeev	Omsk Branch of Sobolev Institute of Mathematics, Russia
Katti Faceli	Universidade Federal de São Carlos, Brasil
João Paulo Fernandes	University of Coimbra, Portugal
Pedro Ferreira	University of Lisbon, Portugal
José Rui Figueira	University of Lisbon, Portugal
Bogdan Filipic	Jožef Stefan Institute, Slovenia
Steffen Finck	Vorarlberg University of Applied Sciences, Austria
Andreas Fischbach	Cologne University of Applied Sciences, Germany
Peter Fleming	University of Sheffield, UK
Carlos M. Fonseca	University of Coimbra, Portugal
Martina Friese	Cologne University of Applied Sciences, Germany
Marcus Gallagher	University of Queensland, Australia
José García-Nieto	University of Málaga, Spain
Antonio Gaspar-Cunha	University of Minho, Portugal
Mario Giacobini	University of Torino, Italy
Tobias Glasmachers	Institut für Neuroinformatik, Germany
Roderich Gross	University of Sheffield, UK
Andreia Guerreiro	University of Coimbra, Portugal
Jussi Hakanen	University of Jyväskylä, Finland
Hisashi Handa	Kindai University, Japan
Julia Handl	University of Manchester, UK
Jin-Kao Hao	University of Angers, France
Emma Hart	Napier University, UK
Nikolaus Hansen	Inria, France



Verena Heidrich-Meisner	Christian-Albrechts-Universität zu Kiel, Germany
Carlos Henggeler Antunes	University of Coimbra, Portugal
Hisao Ishibuchi	Southern University of Science and Technology, China
Christian Jacob	University of Calgary, Canada
Domagoj Jakobovic	University of Zagreb, Croatia
Thomas Jansen	Aberystwyth University, Wales
Yaochu Jin	University of Surrey, England
Laetitia Jourdan	University of Lille, France
Bryant Julstrom	St. Cloud State University, USA
George Karakostas	McMaster University, Canada
Graham Kendall	University of Nottingham, UK
Timo Kötzing	Hasso-Plattner-Institut, Germany
Krzysztof Krawiec	Poznan University of Technology, Poland
Martin Krejca	Hasso-Plattner-Institut, Germany
Algirdas Lančinskas	Vilnius University, Lithuania
William Langdon	University College London, England
Frederic Lardeux	University of Angers, France
Jörg Lässig	University of Applied Sciences Zittau/Görlitz, Germany
Per Kristian Lehre	University of Birmingham, UK
Johannes Lengler	ETH Zurich, Switzerland
Arnaud Liefoghe	University of Lille, France
Andrei Lissovoi	University of Sheffield, UK
Giosuè Lo Bosco	Università di Palermo, Italy
Fernando Lobo	University of Algarve, Portugal
Daniele Loiacono	Politecnico di Milano, Italy
Manuel López-Ibáñez	University of Manchester, UK
Nuno Lourenço	University of Coimbra, Portugal
Jose A. Lozano	University of the Basque Country, Spain
Gabriel Luque	University of Málaga, Spain
Thibaut Lust	Sorbonne University, France
Penousal Machado	University of Coimbra, Portugal
Jacek Mańdziuk	Warsaw University of Technology, Poland
Vittorio Maniezzo	University of Bologna, Italy
Elena Marchiori	Radboud University, The Netherlands
Giancarlo Mauri	University of Milano-Bicocca, Italy
James McDermott	University College Dublin, Republic of Ireland
Alexander Melkozerov	Tomsk State University of Control Systems and Radioelectronics, Russia
J. J. Merelo	University of 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
Edmondo Minisci	University of Strathclyde, Scotland
Gara Miranda	University of La Laguna, Spain
Marco A. Montes De Oca	“clypd, Inc.”, USA

Sanaz Mostaghim	Otto von Guericke University Magdeburg, Germany
Boris Naujoks	Cologne University of Applied Sciences, Germany
Antonio J. Nebro	University of Málaga, Spain
Ferrante Neri	De Montfort University, England
Frank Neumann	University of Adelaide, Australia
Phan Nguyen	University of Birmingham, UK
Miguel Nicolau	University College Dublin, Republic of Ireland
Kouhei Nishida	Shinshu University, Japan
Michael O' Neill	University College Dublin, Republic of Ireland
Gabriela Ochoa	University of Stirling, Scotland
Pietro S Oliveto	University of Sheffield, UK
José Carlos Ortiz-Bayliss	Tecnológico de Monterrey, Mexico
Ben Paechter	Napier University, UK
Gregor Papa	Jožef Stefan Institute, Slovenia
Gisele Pappa	Universidade Federal de Minas Gerais, Brasil
Luis Paquete	University of Coimbra, Portugal
Andrew J. Parkes	University of Nottingham, UK
Margarida Pato	Universidade de Lisboa, Portugal
Mario Pavone	University of Catania, Italy
David Pelta	University of Granada, Spain
Martin Pilat	Charles University in Prague, Czech Republic
Petr Pošík	Czech Technical University in Prague, Czech Republic
Mike Preuss	University of Münster, Germany
Robin Purshouse	University of Sheffield, UK
Günther Raidl	Vienna University of Technology, Austria
William Rand	North Carolina State University, USA
Khaled Rasheed	University of Georgia, USA
Tapabrata Ray	University of New South Wales, Australia
Eduardo Rodriguez-Tello	CINVESTAV-Tamaulipas, Mexico
Günter Rudolph	TU Dortmund University, Germany
Andrea Roli	University of Bologna, Italy
Agostinho Rosa	University of Lisbon, Portugal
Jonathan Rowe	University of Birmingham, UK
Thomas Runarsson	University of Iceland, Iceland
Thomas A. Runkler	Siemens Corporate Technology, Germany
Conor Ryan	University of Limerick, Republic of Ireland
Frédéric Saubion	University of Angers, France
Robert Schaefer	AGH University of Science and Technology, Poland
Andrea Schaefer	University of Udine, Italy
Manuel Schmitt	ALYN Woldenberg Family Hospital, Israel
Marc Schoenauer	Inria, France
Oliver Schuetze	CINVESTAV-IPN, Mexico
Eduardo Segredo	Napier University, UK
Martin Serpell	University of the West of England, England
Roberto Serra	University of Modena and Reggio Emilia, Italy
Marc Sevaux	Université de Bretagne-Sud, France
Shinichi Shirakawa	Yokohama National University, Japan

Kevin Sim	Napier University, UK
Moshe Sipper	Ben-Gurion University of the Negev, Israel
Jim Smith	University of the West of England, England
Christine Solnon	Institut National des Sciences Appliquées de Lyon, France
Sebastian Stich	EPFL, Switzerland
Catalin Stoean	University of Craiova, Romania
Jörg Stork	Cologne University of Applied Sciences, Germany
Thomas Stützle	Université Libre de Bruxelles, Belgium
Dirk Sudholt	University of Sheffield, UK
Andrew Sutton	University of Minnesota Duluth, USA
Jerry Swan	University of York, UK
Ricardo H. C. Takahashi	Universidade Federal de Minas Gerais, Brasil
El-Ghazali Talbi	University of Lille, France
Daniel Tauritz	Missouri University of Science and Technology, USA
Jorge Tavares	Microsoft, Germany
Hugo Terashima	Tecnológico de Monterrey, Mexico
German Terrazas Angulo	University of Nottingham, UK
Andrea Tettamanzi	University Nice Sophia Antipolis, France
Lothar Thiele	ETH Zurich, Switzerland
Dirk Thierens	Utrecht University, The Netherlands
Renato Tinós	University of São Paulo, Brasil
Alberto Tonda	Institut National de la Recherche Agronomique, France
Heike Trautmann	University of Münster, Germany
Leonardo Trujillo	Instituto Tecnológico de Tljuana, Mexico
Tea Tusar	Jožef Stefan Institute, Slovenia
Nadarajen Veerapen	University of Stirling, UK
Sébastien Verel	Université du Littoral Côte d'Opale, France
Markus Wagner	University of Adelaide, Australia
Elizabeth Wanner	Aston University, UK
Carsten Witt	Technical University of Denmark, Denmark
Man Leung Wong	Lingnan University, Hong Kong
John Woodward	Queen Mary University of London, UK
Ning Xiong	Mälardalen University, Sweden
Shengxiang Yang	De Montfort University, UK
Gary Yen	Oklahoma State University, USA
Martin Zaefferer	Cologne University of Applied Sciences, Germany
Ales Zamuda	University of Maribor, Slovenia
Christine Zarges	Aberystwyth University, UK

## Additional Reviewers

Matthew Doyle  
 Yue Gu  
 Stefano Mauceri  
 Anil Özdemir  
 Isaac Vandermeulen

## **Invited Talks**

# **The Shape of Art History in the Eyes of the Machine**

Ahmed Elgammal

Art and Artificial Intelligence Laboratory, Rutgers University

Advances in Artificial Intelligence are changing things around us. Is art and creativity immune from the perceived AI takeover? In this talk I will highlight some of the advances in the area of Artificial Intelligence and Art. I will argue about how investigating perceptual and cognitive tasks related to human creativity in visual art is essential for advancing the fields of AI and multimedia systems. On the other hand, how AI can change the way we look at art and art history.

The talk will present results of recent research activities at the Art and Artificial Intelligence Laboratory at Rutgers University. We investigate perceptual and cognitive tasks related to human creativity in visual art. In particular, we study problems related to art styles, influence, and the quantification of creativity. We develop computational models that aim at providing answers to questions about what characterizes the sequence and evolution of changes in style over time. The talk will also cover advances in automated prediction of style, how that relates to art history methodology, and what that tells us about how the machine sees art history. The talk will also delve into our recent research on quantifying creativity in art in regard to its novelty and influence, as well as computational models that simulate the art-producing system.

# **Self-organization, Emergence and Stigmergy: Coordination from the Bottom-up**

Francis Heylighen

Evolution, Complexity and Cognition Group,  
Center Leo Apostel, Vrije Universiteit Brussel

The purpose of this presentation is to review and clarify some of the foundational concepts and mechanisms that underlie parallel problem solving in nature. A problem can be conceived as a tension between the present, “unfit” state and some fit state in which the tension would be relaxed [2]. Formulated in terms of dynamic systems, the solution is then a fitness peak, a potential valley, or most generally an attractor in the state space of the system under consideration. Solving the problem means finding a path that leads from the present state to such an attractor state. This spontaneous descent of a system into an attractor is equivalent to the self-organization of the components or agents in the system, meaning that the agents mutually adapt so as to achieve a stable interaction pattern. The interaction between agents can be conceived as a propagation of challenges: a challenge is a state of tension that incites an agent to act so as to reduce the tension. That action, however, typically creates a new challenge for one or more neighboring agents, who act in turn, thus creating yet further challenges. The different actions take place in parallel, producing a “wave” of activity that propagates across the environment. Because of the general relaxation dynamics, this activity eventually settles in an attractor. The stability of the resulting global configuration means that the different agents have now “coordinated” their actions into a synergetic pattern: a global “order” has emerged out of local interactions [1]. Such self-organization and “natural problem solving” are therefore in essence equivalent. Two mechanisms facilitate this process: (1) order from noise [4] notes that injecting random variation accelerates the exploration of the state space, and thus the discovery of deep attractors; (2) stigmergy means that agents leave traces of their action in a shared medium. These traces challenge other agents to build further on the activity. They function like a collective memory and communication medium that facilitates coordination without requiring either top-down control or direct agent-to-agent communication [3].

## **References**

1. Heylighen, F.: The science of self-organization and adaptivity. *Enycl. Life Support Syst.* **5**(3), 253–280 (2001)
2. Heylighen, F.: Challenge Propagation: towards a theory of distributed intelligence and the global brain. *Spanda J.* **V**(2), 51–63 (2014)

3. Heylighen, F.: Stigmergy as a universal coordination mechanism I: definition and components. *Cogn. Syst. Res.* **38**, 4–13 (2016). <https://doi.org/10.1016/j.cogsys.2015.12.002>
4. Von Foerster, H.: On self-organizing systems and their environments. In: *Self-organizing Systems*, pp. 31–50 (1960)

# On Physarum Computations

Kurt Mehlhorn

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken

Let  $c$  be a positive vector in  $\mathbb{R}^m$ , let  $A \in \mathbb{R}^{n \times m}$  and  $b \in \mathbb{R}^n$ . Consider

$$\text{minimize } c^T |f| \text{ subject to } Af = b. \quad (1)$$

The solution is a feasible  $f$  of minimum weighted 1-norm. The Physarum dynamics operates on a state  $x \in \mathbb{R}_{>0}^m$ . The state evolves according to the system of differential equations

$$\dot{x} = q - x,$$

where  $q$  is the minimum energy feasible solution, i.e.,

$$q = \operatorname{argmin}_f \left\{ \sum_e r_e f_e^2 \mid Af = b \right\} \quad \text{and} \quad r_e = c_e / x_e. \quad (2)$$

In [1] it is shown that the dynamics (2) converges to an optimal solution of (1). Previously, this was known for the special case of the undirected shortest path problem [2–4]; here  $A$  is the node-arc incidence matrix of a directed graph and  $b$  is the demand vector. Further work can be found in [8–11].

The theoretical investigation of the Physarum dynamics was motivated by wet-lab experiments [5]. The theoretical model was introduced by [6], and convergence for the case of parallel links was shown in [7].

## References

1. Becker, R., Bonifaci, V., Karrenbauer, A., Kolev, P., Mehlhorn, K.: Two results on slime mold computations (2017). CoRR abs/1707.06631. <https://arxiv.org/abs/1707.06631>
2. Bonifaci, V., Mehlhorn, K., Varma, G.: Physarum can compute shortest paths. *J. Theor. Biol.* **309**, 121–133 (2012). <http://arxiv.org/abs/1106.0423>
3. Bonifaci, V.: Physarum can compute shortest paths: a short proof. *Inf. Process. Lett.* **113**(1–2), 4–7 (2013)
4. Bonifaci, V.: A revised model of fluid transport optimization in physarum polycephalum. CoRR abs/1606.04225 (2016)
5. Nakagaki, T., Yamada, H., Tóth, A.: Maze-solving by an amoeboid organism. *Nature* **407**, 470 (2000)
6. Tero, A., Kobayashi, R., Nakagaki, T.: A mathematical model for adaptive transport network in path finding by true slime mold. *J. Theor. Biol.*, 553–564 (2007)
7. Miyaji, T., Ohnishi, I.: Physarum can solve the shortest path problem on riemannian surface mathematically rigorously. *Int. J. Pure Appl. Math.* **47**, 353–369 (2008)



8. Ito, K., Johansson, A., Nakagaki, T., Tero, A.: Convergence properties for the Physarum solver (2011). arXiv:1101.5249v1
9. Straszak, D., Vishnoi, N.K.: IRLS and slime mold: Equivalence and convergence (2016). CoRR abs/1601.02712
10. Straszak, D., Vishnoi, N.K.: On a natural dynamics for linear programming. In: ITCS, p. 291. ACM, New York (2016)
11. Straszak, D., Vishnoi, N.K.: Natural algorithms for flow problems. In: SODA, pp. 1868–1883 (2016)

# Contents – Part I

## Numerical Optimization

A Comparative Study of Large-Scale Variants of CMA-ES . . . . .	3
<i>Konstantinos Varelas, Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Ouassim Ait ElHara, Yann Semet, Rami Kassab, and Frédéric Barbaresco</i>	
Design of a Surrogate Model Assisted (1 + 1)-ES . . . . .	16
<i>Arash Kayhani and Dirk V. Arnold</i>	
Generalized Self-adapting Particle Swarm Optimization Algorithm . . . . .	29
<i>Mateusz Uliński, Adam Żychowski, Michał Okulewicz, Mateusz Zaborski, and Hubert Kordulewski</i>	
PSO-Based Search Rules for Aerial Swarms Against Unexplored Vector Fields via Genetic Programming . . . . .	41
<i>Palina Bartashevich, Illya Bakurov, Sanaz Mostaghim, and Leonardo Vanneschi</i>	
Towards an Adaptive CMA-ES Configurator . . . . .	54
<i>Sander van Rijn, Carola Doerr, and Thomas Bäck</i>	

## Combinatorial Optimization

A Probabilistic Tree-Based Representation for Non-convex Minimum Cost Flow Problems . . . . .	69
<i>Behrooz Ghasemishabankareh, Melih Ozlen, Frank Neumann, and Xiaodong Li</i>	
Comparative Study of Different Memetic Algorithm Configurations for the Cyclic Bandwidth Sum Problem . . . . .	82
<i>Eduardo Rodriguez-Tello, Valentina Narvaez-Teran, and Frédéric Lardeux</i>	
Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic . . . . .	95
<i>Renato Tinós, Keld Helsgaun, and Darrell Whitley</i>	
Escherization with a Distance Function Focusing on the Similarity of Local Structure . . . . .	108
<i>Yuichi Nagata</i>	

Evolutionary Search of Binary Orthogonal Arrays . . . . .	121
<i>Luca Mariot, Stjepan Picek, Domagoj Jakobovic, and Alberto Leporati</i>	
Heavy-Tailed Mutation Operators in Single-Objective Combinatorial Optimization . . . . .	134
<i>Tobias Friedrich, Andreas Göbel, Francesco Quinzan, and Markus Wagner</i>	
Heuristics in Permutation GOMEA for Solving the Permutation Flowshop Scheduling Problem. . . . .	146
<i>G. H. Aalvanger, N. H. Luong, P. A. N. Bosman, and D. Thierens</i>	
On the Performance of Baseline Evolutionary Algorithms on the Dynamic Knapsack Problem . . . . .	158
<i>Vahid Roostapour, Aneta Neumann, and Frank Neumann</i>	
On the Synthesis of Perturbative Heuristics for Multiple Combinatorial Optimisation Domains . . . . .	170
<i>Christopher Stone, Emma Hart, and Ben Paechter</i>	
<b>Genetic Programming</b>	
EDDA-V2 – An Improvement of the Evolutionary Demes Despeciation Algorithm . . . . .	185
<i>Illya Bakurov, Leonardo Vanneschi, Mauro Castelli, and Francesco Fontanella</i>	
Extending Program Synthesis Grammars for Grammar-Guided Genetic Programming . . . . .	197
<i>Stefan Forstenlechner, David Fagan, Miguel Nicolau, and Michael O’Neill</i>	
Filtering Outliers in One Step with Genetic Programming . . . . .	209
<i>Uriel López, Leonardo Trujillo, and Pierrick Legrand</i>	
GOMGE: Gene-Pool Optimal Mixing on Grammatical Evolution . . . . .	223
<i>Eric Medvet, Alberto Bartoli, Andrea De Lorenzo, and Fabiano Tarlao</i>	
Self-adaptive Crossover in Genetic Programming: The Case of the Tartarus Problem. . . . .	236
<i>Thomas D. Griffiths and Anikó Ekárt</i>	

**Multi-objective Optimization**

A Decomposition-Based Evolutionary Algorithm for Multi-modal Multi-objective Optimization . . . . . 249  
*Ryoji Tanabe and Hisao Ishibuchi*

A Double-Niched Evolutionary Algorithm and Its Behavior on Polygon-Based Problems . . . . . 262  
*Yiping Liu, Hisao Ishibuchi, Yusuke Nojima, Naoki Masuyama, and Ke Shang*

Artificial Decision Maker Driven by PSO: An Approach for Testing Reference Point Based Interactive Methods . . . . . 274  
*Cristóbal Barba-González, Vesa Ojalehto, José García-Nieto, Antonio J. Nebro, Kaisa Miettinen, and José F. Aldana-Montes*

A Simple Indicator Based Evolutionary Algorithm for Set-Based Minmax Robustness . . . . . 286  
*Yue Zhou-Kangas and Kaisa Miettinen*

Extending the Speed-Constrained Multi-objective PSO (SMPSO) with Reference Point Based Preference Articulation. . . . . 298  
*Antonio J. Nebro, Juan J. Durillo, José García-Nieto, Cristóbal Barba-González, Javier Del Ser, Carlos A. Coello Coello, Antonio Benítez-Hidalgo, and José F. Aldana-Montes*

Improving lby1EA to Handle Various Shapes of Pareto Fronts. . . . . 311  
*Yiping Liu, Hisao Ishibuchi, Yusuke Nojima, Naoki Masuyama, and Ke Shang*

New Initialisation Techniques for Multi-objective Local Search: Application to the Bi-objective Permutation Flowshop . . . . . 323  
*Aymeric Blot, Manuel López-Ibáñez, Marie-Éléonore Kessaci, and Laetitia Jourdan*

Towards a More General Many-objective Evolutionary Optimizer . . . . . 335  
*Jesús Guillermo Falcón-Cardona and Carlos A. Coello Coello*

Towards Large-Scale Multiobjective Optimisation with a Hybrid Algorithm for Non-dominated Sorting . . . . . 347  
*Margarita Markina and Maxim Buzdalov*

Tree-Structured Decomposition and Adaptation in MOEA/D . . . . . 359  
*Hanwei Zhang and Aimin Zhou*

Use of Reference Point Sets in a Decomposition-Based Multi-Objective Evolutionary Algorithm . . . . . 372  
*Edgar Manóatl Lopez and Carlos A. Coello Coello*

Use of Two Reference Points in Hypervolume-Based Evolutionary Multiobjective Optimization Algorithms . . . . .	384
<i>Hisao Ishibuchi, Ryo Imada, Naoki Masuyama, and Yusuke Nojima</i>	

### Parallel and Distributed Frameworks

Introducing an Event-Based Architecture for Concurrent and Distributed Evolutionary Algorithms . . . . .	399
<i>Juan J. Merelo Guervós and J. Mario García-Valdez</i>	
Analyzing Resilience to Computational Glitches in Island-Based Evolutionary Algorithms . . . . .	411
<i>Rafael Nogueras and Carlos Cotta</i>	
Spark Clustering Computing Platform Based Parallel Particle Swarm Optimizers for Computationally Expensive Global Optimization . . . . .	424
<i>Qiqi Duan, Lijun Sun, and Yuhui Shi</i>	
Weaving of Metaheuristics with Cooperative Parallelism . . . . .	436
<i>Jheisson López, Danny Múnera, Daniel Diaz, and Salvador Abreu</i>	

### Applications

Conditional Preference Learning for Personalized and Context-Aware Journey Planning . . . . .	451
<i>Mohammad Haqqani, Homayoon Ashrafzadeh, Xiaodong Li, and Xinghuo Yu</i>	
Critical Fractile Optimization Method Using Truncated Halton Sequence with Application to SAW Filter Design . . . . .	464
<i>Kiyoharu Tagawa</i>	
Directed Locomotion for Modular Robots with Evolvable Morphologies . . . . .	476
<i>Gongjin Lan, Milan Jelisavcic, Diederik M. Roijers, Evert Haasdijk, and A. E. Eiben</i>	
Optimisation and Illumination of a Real-World Workforce Scheduling and Routing Application (WSRP) via Map-Elites . . . . .	488
<i>Neil Urquhart and Emma Hart</i>	
Prototype Discovery Using Quality-Diversity . . . . .	500
<i>Alexander Hagg, Alexander Asteroth, and Thomas Bäck</i>	
Sparse Incomplete LU-Decomposition for Wave Farm Designs Under Realistic Conditions. . . . .	512
<i>Dídac Rodríguez Arbonès, Nataliia Y. Sergiienko, Boyin Ding, Oswin Krause, Christian Igel, and Markus Wagner</i>	

Understanding Climate-Vegetation Interactions in Global Rainforests  
Through a GP-Tree Analysis . . . . . 525  
*Anuradha Kodali, Marcin Szubert, Kamalika Das, Sangram Ganguly,  
and Joshua Bongard*

**Author Index** . . . . . 537

## Contents – Part II

### Runtime Analysis and Approximation Results

A General Dichotomy of Evolutionary Algorithms on Monotone Functions . . . <i>Johannes Lengler</i>	3
Artificial Immune Systems Can Find Arbitrarily Good Approximations for the NP-Hard Partition Problem . . . . . <i>Dogan Corus, Pietro S. Oliveto, and Donya Yazdani</i>	16
A Simple Proof for the Usefulness of Crossover in Black-Box Optimization. . . <i>Eduardo Carvalho Pinto and Carola Doerr</i>	29
Destructiveness of Lexicographic Parsimony Pressure and Alleviation by a Concatenation Crossover in Genetic Programming . . . . . <i>Timo Kötzing, J. A. Gregor Lagodzinski, Johannes Lengler, and Anna Melnichenko</i>	42
Exploration and Exploitation Without Mutation: Solving the <i>Jump</i> Function in $\Theta(n)$ Time . . . . . <i>Darrell Whitley, Swetha Varadarajan, Rachel Hirsch, and Anirban Mukhopadhyay</i>	55
Fast Artificial Immune Systems . . . . . <i>Dogan Corus, Pietro S. Oliveto, and Donya Yazdani</i>	67
First-Hitting Times for Finite State Spaces . . . . . <i>Timo Kötzing and Martin S. Krejca</i>	79
First-Hitting Times Under Additive Drift . . . . . <i>Timo Kötzing and Martin S. Krejca</i>	92
Level-Based Analysis of the Population-Based Incremental Learning Algorithm. . . . . <i>Per Kristian Lehre and Phan Trung Hai Nguyen</i>	105
Precise Runtime Analysis for Plateaus . . . . . <i>Denis Antipov and Benjamin Doerr</i>	117
Ring Migration Topology Helps Bypassing Local Optima . . . . . <i>Clemens Frahnöw and Timo Kötzing</i>	129

Runtime Analysis of Evolutionary Algorithms for the Knapsack Problem  
with Favorably Correlated Weights . . . . . 141  
*Frank Neumann and Andrew M. Sutton*

Theoretical Analysis of Lexicase Selection in Multi-objective Optimization. . . . . 153  
*Thomas Jansen and Christine Zarges*

Towards a Running Time Analysis of the (1+1)-EA for OneMax and  
LeadingOnes Under General Bit-Wise Noise . . . . . 165  
*Chao Bian, Chao Qian, and Ke Tang*

**Fitness Landscape Modeling and Analysis**

A Surrogate Model Based on Walsh Decomposition  
for Pseudo-Boolean Functions . . . . . 181  
*Sébastien Verel, Bilel Derbel, Arnaud Liefooghe, Hernán Aguirre,  
and Kiyoshi Tanaka*

Bridging Elementary Landscapes and a Geometric Theory  
of Evolutionary Algorithms: First Steps . . . . . 194  
*Marcos Diez García and Alberto Moraglio*

Empirical Analysis of Diversity-Preserving Mechanisms on Example  
Landscapes for Multimodal Optimisation . . . . . 207  
*Edgar Covantes Osuna and Dirk Sudholt*

Linear Combination of Distance Measures for Surrogate Models  
in Genetic Programming . . . . . 220  
*Martin Zaefferer, Jörg Stork, Oliver Flasch,  
and Thomas Bartz-Beielstein*

On Pareto Local Optimal Solutions Networks. . . . . 232  
*Arnaud Liefooghe, Bilel Derbel, Sébastien Verel, Manuel López-Ibáñez,  
Hernán Aguirre, and Kiyoshi Tanaka*

Perturbation Strength and the Global Structure of QAP Fitness Landscapes . . . . . 245  
*Gabriela Ochoa and Sebastian Herrmann*

Sampling Local Optima Networks of Large Combinatorial Search Spaces:  
The QAP Case . . . . . 257  
*Sébastien Verel, Fabio Daolio, Gabriela Ochoa, and Marco Tomassini*

**Algorithm Configuration, Selection, and Benchmarking**

Algorithm Configuration Landscapes: More Benign Than Expected? . . . . . 271  
*Yasha Pushak and Holger Hoos*



A Model-Based Framework for Black-Box Problem Comparison Using Gaussian Processes . . . . .	284
<i>Sobia Saleem, Marcus Gallagher, and Ian Wood</i>	
A Suite of Computationally Expensive Shape Optimisation Problems Using Computational Fluid Dynamics . . . . .	296
<i>Steven J. Daniels, Alma A. M. Rahat, Richard M. Everson, Gavin R. Tabor, and Jonathan E. Fieldsend</i>	
Automated Selection and Configuration of Multi-Label Classification Algorithms with Grammar-Based Genetic Programming. . . . .	308
<i>Alex G. C. de Sá, Alex A. Freitas, and Gisele L. Pappa</i>	
Performance Assessment of Recursive Probability Matching for Adaptive Operator Selection in Differential Evolution. . . . .	321
<i>Mudita Sharma, Manuel López-Ibáñez, and Dimitar Kazakov</i>	
Program Trace Optimization . . . . .	334
<i>Alberto Moraglio and James McDermott</i>	
Sampling Heuristics for Multi-objective Dynamic Job Shop Scheduling Using Island Based Parallel Genetic Programming . . . . .	347
<i>Deepak Karunakaran, Yi Mei, Gang Chen, and Mengjie Zhang</i>	
Sensitivity of Parameter Control Mechanisms with Respect to Their Initialization . . . . .	360
<i>Carola Doerr and Markus Wagner</i>	
Tailoring Instances of the 1D Bin Packing Problem for Assessing Strengths and Weaknesses of Its Solvers . . . . .	373
<i>Ivan Amaya, José Carlos Ortiz-Bayliss, Santiago Enrique Conant-Pablos, Hugo Terashima-Marín, and Carlos A. Coello Coello</i>	
<b>Machine Learning and Evolutionary Algorithms</b>	
Adaptive Advantage of Learning Strategies: A Study Through Dynamic Landscape . . . . .	387
<i>Nam Le, Michael O’Neill, and Anthony Brabazon</i>	
A First Analysis of Kernels for Kriging-Based Optimization in Hierarchical Search Spaces . . . . .	399
<i>Martin Zaefferer and Daniel Horn</i>	
Challenges in High-Dimensional Reinforcement Learning with Evolution Strategies . . . . .	411
<i>Nils Müller and Tobias Glasmachers</i>	

Lamarckian Evolution of Convolutional Neural Networks . . . . .	424
<i>Jonas Prellberg and Oliver Kramer</i>	
Learning Bayesian Networks with Algebraic Differential Evolution . . . . .	436
<i>Marco Bautoletti, Alfredo Milani, and Valentino Santucci</i>	
Optimal Neuron Selection and Generalization: NK Ensemble Neural Networks . . . . .	449
<i>Darrell Whitley, Renato Tinós, and Francisco Chicano</i>	
What Are the Limits of Evolutionary Induction of Decision Trees? . . . . .	461
<i>Krzysztof Jurczuk, Daniel Reska, and Marek Kretowski</i>	
<b>Tutorials and Workshops at PPSN 2018</b>	
Tutorials at PPSN 2018 . . . . .	477
<i>Gisele Lobo Pappa, Michael T. M. Emmerich, Ana Bazzan, Will Browne, Kalyanmoy Deb, Carola Doerr, Marko Đurasević, Michael G. Epitropakis, Saemundur O. Haraldsson, Domagoj Jakobovic, Pascal Kerschke, Krzysztof Krawiec, Per Kristian Lehre, Xiaodong Li, Andrei Lissovoi, Pekka Malo, Luis Martí, Yi Mei, Juan J. Merelo, Julian F. Miller, Alberto Moraglio, Antonio J. Nebro, Su Nguyen, Gabriela Ochoa, Pietro Oliveto, Stjepan Picek, Nelishia Pillay, Mike Preuss, Marc Schoenauer, Roman Senkerik, Ankur Sinha, Ofer Shir, Dirk Sudholt, Darrell Whitley, Mark Wineberg, John Woodward, and Mengjie Zhang</i>	
Workshops at PPSN 2018 . . . . .	490
<i>Robin Purshouse, Christine Zarges, Sylvain Cussat-Blanc, Michael G. Epitropakis, Marcus Gallagher, Thomas Jansen, Pascal Kerschke, Xiaodong Li, Fernando G. Lobo, Julian Miller, Pietro S. Oliveto, Mike Preuss, Giovanni Squillero, Alberto Tonda, Markus Wagner, Thomas Weise, Dennis Wilson, Borys Wróbel, and Aleš Zamuda</i>	
<b>Author Index</b> . . . . .	499

# **Numerical Optimization**



# A Comparative Study of Large-Scale Variants of CMA-ES

Konstantinos Varelas<sup>1,2(✉)</sup>, Anne Auger<sup>1</sup>, Dimo Brockhoff<sup>1</sup>,  
Nikolaus Hansen<sup>1</sup>, Ouassim Ait ElHara<sup>1</sup>, Yann Semet<sup>3</sup>,  
Rami Kassab<sup>2</sup>, and Frédéric Barbaresco<sup>2</sup>

<sup>1</sup> Inria, RandOpt team, CMAP, École Polytechnique, Palaiseau, France  
{konstantinos.varelas, anne.auger, dimo.brockhoff, nikolaus.hansen,  
ouassim.elHara}@inria.fr

<sup>2</sup> Thales LAS France SAS - Limours, Limours, France

<sup>3</sup> Thales Research Technology, Palaiseau, France  
yann.semet@thalesgroup.com

**Abstract.** The CMA-ES is one of the most powerful stochastic numerical optimizers to address difficult black-box problems. Its intrinsic time and space complexity is quadratic—limiting its applicability with increasing problem dimensionality. To circumvent this limitation, different large-scale variants of CMA-ES with subquadratic complexity have been proposed over the past ten years. To-date however, these variants have been tested and compared only in rather restrictive settings, due to the lack of a comprehensive large-scale testbed to assess their performance. In this context, we introduce a new large-scale testbed with dimension up to 640, implemented within the [COCO](#) benchmarking platform. We use this testbed to assess the performance of several promising variants of CMA-ES and the standard limited-memory L-BFGS. In all tested dimensions, the best CMA-ES variant solves more problems than L-BFGS for larger budgets while L-BFGS outperforms the best CMA-ES variant for smaller budgets. However, over all functions, the cumulative runtime distributions between L-BFGS and the best CMA-ES variants are close (less than a factor of 4 in high dimension).

Our results illustrate different scaling behaviors of the methods, expose a few defects of the algorithms and reveal that for dimension larger than 80, LM-CMA solves more problems than V<sub>k</sub>D-CMA while in the cumulative runtime distribution over all functions the V<sub>k</sub>D-CMA dominates LM-CMA for budgets up to  $10^4$  times dimension and for all budgets up to dimension 80.

## 1 Introduction

The CMA-ES is a stochastic derivative-free optimization algorithm, recognized as one of the most powerful optimizers for solving difficult black-box optimization problems, i.e., non-linear, non quadratic, non-convex, non-smooth, and/or noisy problems [6]. Its intrinsic complexity in terms of memory and internal computational effort is quadratic in the dimensionality,  $n$ , of the black-box objective

function to be solved, denoted in a generic manner as:  $f : x \in \mathbb{R}^n \mapsto \mathbb{R}$ . This complexity restricts its application when the number  $n$  of variables is in the order of a few hundred. For this reason, different “large”-scale variants of CMA-ES have been introduced over the past ten years. They all aim at a sub-quadratic space and time complexity [3, 7, 9, 11, 12, 14, 15]. The common feature of the variants is to restrict the model of the covariance matrix and provide a sparse representation that can be stored, sampled and updated in  $\mathcal{O}(n \times m)$  operations with  $m \ll n$ . Yet the approaches to do so are quite different. On the one-hand, the seminal limited memory BFGS, L-BFGS [10], inspired the introduction of the limited memory CMA (LM-CMA, [11, 12]) where the main idea is to approximate at iteration  $t \gg m$  the sum over  $t$  terms composing the covariance matrix by a sum over  $m$  terms. This same approach is used in the RmES algorithm [9]. On the other-hand, the sep-CMA [14] and VkD-CMA [3] algorithms enforce a predefined structure of the covariance matrix (for instance diagonal for the sep-CMA) and project at each iteration the updated matrix onto the restricted space.

After designing a novel algorithm, the next step is to assess its performance and compare it with its competitors. This benchmarking step is crucial but is known to be non-trivial and tedious. For this reason, during the past ten years, an important effort went into the development of the COCO platform to introduce a thorough benchmarking methodology and to automatize the tedious benchmarking process [4]. With COCO, algorithms are put at a standardized test and performance assessment is greatly facilitated as users can download and compare datasets of 180+ previously benchmarked algorithms.<sup>1</sup>

Yet so far, the testbeds provided with COCO are not suitable for benchmarking large-scale algorithms. One bottleneck with the current suite is the use of full orthogonal matrices with  $n^2$  coefficients in the definition of many of the functions which makes the computation too expensive for thorough benchmarking studies. For this reason, it was proposed to replace these matrices by orthogonal matrices with a sparse structure: permuted block-diagonal matrices [1]. We utilize this idea to introduce a large-scale test suite with search space dimensions from 20 up to 640.

In this context, the main contributions of this paper are (i) the introduction of a large-scale testbed within the COCO framework and (ii) the comparative review and performance assessment of the currently most promising large-scale variants of CMA-ES and their comparison to the well established L-BFGS algorithm. Besides the general performance quantification and comparison, the benchmarking allows to identify defects of the algorithms or of their implementations (that shall be fixed in the near future).

---

<sup>1</sup> All raw datasets are available for download at <http://coco.gforge.inria.fr/doku.php?id=algorithms> while already postprocessed results are available (without the need to install COCO) at <http://coco.gforge.inria.fr/ppdata-archive>.

## 2 The **bbob-Largescale COCO** Testbed

Performance assessment is a crucial part of algorithm design and an important aspect when recommending algorithms for practical use. Choosing a representative testbed and setting up an assessment methodology are non-trivial and tedious. Hence, it is desirable to automatize the assessment in a standardized benchmarking process. In recent years, the Comparing Continuous Optimizers platform (**COCO**, [4]) has been developed particularly for this purpose and became a quasi-standard in the optimization community in the case of medium-scale unconstrained black-box optimization. The specific aspects of the **COCO** platform are: (i) a *quantitative performance assessment* by reporting runlengths which results in a *budget-free* experimental setting, (ii) *fully scalable* test functions in standard dimensions 2–40, (iii) *full automation* of the experiments with example code in C/C++, Java, MATLAB/Octave, python, and R, (iv) the availability of (pseudo-random) *instances of parametrized functions* which allow to naturally compare deterministic and stochastic algorithms, (v) *extensive post-processing functionalities* to visualize and analyze the experimental data, and finally, (vi) a *large amount of publicly available results* to compare with (from running, so far, 180+ algorithm implementations).

Each test problem in the **COCO** platform comes in the form of instances which are constructed in an “onion-style” through basic pseudo-random transformations of a raw function:  $f(x) = H_1 \circ \dots \circ H_{k_1}(f_{\text{raw}}(T_1 \circ \dots \circ T_{k_2}(x)))$ , where  $f_{\text{raw}}$  is the underlying raw function—usually the simplest representative of the function class (like the sphere function with optimum in zero). The  $T_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$  are search space transformations and  $H_i : \mathbb{R} \rightarrow \mathbb{R}$  are function value transformations. Examples of the former are rotations of the search space or translations of the optimum. An example of the latter are strictly increasing (monotone) functions. The transformations applied to the raw function are actually (pseudo)-random, rendering an *instance* of a parametrized transformation [4].

All currently available test suites of **COCO** such as the noiseless, single-objective **bbob** suite with its 24 functions [5] are scalable in the problem dimension and could be used for benchmarking in a large-scale setting. However, their internal computation scales quadratically with the dimension due to the search space rotations applied in most functions—rendering the experiments in higher dimension too costly to be practicable. Also, real-world problems in higher dimension will, most likely, not have quadratically many degrees of freedom. In consequence, artificial test functions, that aim at capturing the typical real-world challenges, shall likely also not have quadratically many internal parameters.

In [1], the authors therefore suggest search space rotations that have linear internal computation costs by being less “rich” than the rotation matrices of the standard **bbob** test suite. Full rotation matrices  $\mathbf{R}$  are replaced by a sequence of three matrices  $\mathbf{P}_{\text{left}} \mathbf{B} \mathbf{P}_{\text{right}}$  in which  $\mathbf{P}_{\text{left}}$  and  $\mathbf{P}_{\text{right}}$  are permutation matrices (with exactly one “1” per row and column) and  $\mathbf{B}$  is an orthogonal block-diagonal matrix. The permutation matrices  $\mathbf{P}_{\text{left}}$  and  $\mathbf{P}_{\text{right}}$  are constructed by  $n_s$  so-called *truncated uniform swaps* [1]: Each swap chooses a first variable  $i$  uniform at random and the second variable within the vicinity of the first variable, i.e.,

uniformly at random within the set  $\{lb(i), \dots, ub(i)\}$  with  $lb(i) = \max(1, i - r_s)$  and  $ub(i) = \min(n, i + r_s)$  and where  $r_s$  is a parameter indicating the distance *range* between the two swapped variables. The computation of  $\mathbf{P}_{\text{left}}\mathbf{B}\mathbf{P}_{\text{right}}$  can be done in linear time, see [1] for details.

In this paper, we introduce the new large-scale variant of the standard **bbob** test suite of **COCO**, denoted as **bbob-largescale**, based on the above ideas. Implemented in the **COCO** platform<sup>2</sup>, it is built on the same 24 raw functions of **bbob** with the default dimensions 20, 40, 80, 160, 320, and 640—the first two overlapping with the original **bbob** suite for compatibility and consistency reasons. The full rotation matrices of the **bbob** suite are replaced by the above construction of permutation and block matrices. Following the recommendations of [1], we chose to do  $n_s = n$  swaps with a range of  $r_s = \lfloor n/3 \rfloor$  and to have all blocks of the same size of  $\min\{40, n\}$  except for the last, possibly smaller block.

One additional change concerns functions with distinct axes: three of the **bbob** functions, namely the *Discus*, the *Sharp Ridge* and the *Bent Cigar* function, have been modified in order to have a constant proportion of distinct axes when the dimension increases [1].

All function instances have their (randomly chosen) global optimum in  $[-5, 5]^n$  and for all but the linear function also the entire (hyper-)ball of radius 1 with the optimum as center lies within this range. Except for the Schwefel, Schaffer, Weierstrass, Gallagher and Katsuura functions, the function value is corrected by  $\min\{1, 40/n\}$  to make the target values comparable over a large range of dimensions. The optimal function value offset is randomly drawn between  $-1000$  and  $1000$ .

Compared to the CEC'08 testbed [17], the **bbob-largescale** test suite has a wider range of problems and difficulties, allows to investigate scaling, applies various regularity-breaking transformations and provides pseudo-random instances to compare naturally deterministic and stochastic algorithms.

### 3 The CMA-ES Algorithm and Some Large-Scale Variants

We introduce in this section the CMA-ES algorithm and give then an overview of large-scale variants that have been introduced in recent years, with an emphasis on the variants that are later empirically investigated.

#### 3.1 The $(\mu/\mu_w, \lambda)$ -CMA-ES

The  $(\mu/\mu_w, \lambda)$ -CMA-ES algorithm samples  $\lambda \geq 2$  candidate solutions from a multivariate normal distribution  $\mathcal{N}(\mathbf{m}_t, \sigma_t^2 \mathbf{C}_t)$  where the mean  $\mathbf{m}_t \in \mathbb{R}^n$  is the incumbent solution,  $\sigma_t$  is a scalar referred to as step-size and  $\mathbf{C}_t \in \mathbb{R}^{n \times n}$  is a positive definite covariance matrix. The algorithm adapts mean, step-size and

<sup>2</sup> The source code of the new test suite (incl. adaptations in **COCO**'s postprocessing) can be found in the [devel-LS-development](#) branch of the **COCO** Github page.

covariance matrix so as to learn second order information on convex-quadratic functions. The CMA-ES is hence a stochastic counterpart of quasi-Newton methods like the BFGS algorithm [10].

The sampling of the candidate solutions  $(\mathbf{x}_t^i)_{1 \leq i \leq \lambda}$  is typically done by computing the eigen-decomposition of the covariance matrix as  $\mathbf{C}_t = \mathbf{B}_t \mathbf{D}_t^2 \mathbf{B}_t^\top$  where  $\mathbf{B}_t$  contains an orthonormal basis of eigenvectors, and  $\mathbf{D}_t$  is a diagonal matrix containing the square roots of the corresponding eigenvalues. The square root of  $\mathbf{C}_t$  is computed as  $\mathbf{C}_t^{1/2} = \mathbf{B}_t \mathbf{D}_t \mathbf{B}_t^\top$  and used for sampling the candidate solutions as  $\mathbf{x}_t^i = \mathbf{m}_t + \sigma_t \mathbf{C}_t^{1/2} \mathbf{z}_t^i$  with  $\mathbf{z}_t^i \sim \mathcal{N}(0, \mathbf{I})$ , where  $\mathcal{N}(0, \mathbf{I})$  denotes a multivariate normal distribution with mean zero and covariance matrix identity. The eigendecomposition has a complexity of  $\mathcal{O}(n^3)$  but is done only every  $\mathcal{O}(n)$  evaluations (*lazy-update*) reducing the complexity of the sampling to  $\mathcal{O}(n^2)$ .

The candidate solutions are then evaluated on  $f$  and ranked from the best to the worse,  $f(\mathbf{x}_t^{1:\lambda}) \leq \dots \leq f(\mathbf{x}_t^{\lambda:\lambda})$ . Mean, step-size and covariance matrix are then updated using the ranked solutions. More precisely the new mean equals  $\mathbf{m}_{t+1} = \sum_{i=1}^{\mu} w_i \mathbf{x}_t^{i:\lambda}$  where  $\mu$  (typically) equals  $\lfloor \lambda/2 \rfloor$  and  $w_i$  are weights satisfying  $w_1 \geq w_2 \geq \dots \geq w_{\mu} > 0$ . Two mechanisms exist to update the covariance matrix, namely the rank-one and rank-mu update. The rank one update adds the rank-one matrix  $\mathbf{p}_{t+1}^c [\mathbf{p}_{t+1}^c]^\top$  to the current covariance matrix, where  $\mathbf{p}_{t+1}^c$  is the evolution path and defined as  $\mathbf{p}_{t+1}^c = (1 - c_c) \mathbf{p}_t^c + \sqrt{c_c(2 - c_c)} \mu_{\text{eff}} (\mathbf{m}_{t+1} - \mathbf{m}_t) / \sigma_t$ , with  $\mu_{\text{eff}} = 1 / \sum_{i=1}^{\mu} w_i^2$  and  $c_c < 1$ . The rank-mu update adds the matrix  $\mathbf{C}_{t+1}^{\mu} = \sum_{i=1}^{\mu} w_i \mathbf{z}_t^{i:\lambda} [\mathbf{z}_t^{i:\lambda}]^\top$  with  $\mathbf{z}_t^{i:\lambda} = (\mathbf{x}_t^{i:\lambda} - \mathbf{m}_t) / \sigma_t$  such that overall the update of the covariance matrix reads

$$\mathbf{C}_{t+1} = (1 - c_1 - c_{\mu}) \mathbf{C}_t + c_1 \mathbf{p}_{t+1}^c [\mathbf{p}_{t+1}^c]^\top + c_{\mu} \mathbf{C}_{t+1}^{\mu} \quad (1)$$

where  $c_1, c_{\mu}$  belong to  $(0, 1)$ . The step-size is updated using the *Cumulative step-size adaptation* (CSA) that utilizes an evolution path cumulating steps of the mean in the isotropic coordinate system with principal axes of  $\mathbf{C}_t$ :  $\mathbf{p}_{t+1}^{\sigma} = (1 - c_{\sigma}) \mathbf{p}_t^{\sigma} + \sqrt{c_{\sigma}(2 - c_{\sigma})} \mu_{\text{eff}} \mathbf{C}_t^{-\frac{1}{2}} \frac{\mathbf{m}_{t+1} - \mathbf{m}_t}{\sigma_t}$ , where  $c_{\sigma} < 1$  and compares the length of the evolution path with its expected length under random selection in order to increase the step-size when the first is larger, or decrease it otherwise. The update of the step-size reads  $\sigma_{t+1} = \sigma_t \exp(\frac{d_{\sigma}}{c_{\sigma}} (\|\mathbf{p}_{t+1}^{\sigma}\| / E[\|\mathcal{N}(0, \mathbf{I})\|]))$  with  $d_{\sigma} > 0$ . Remark that the computation of  $\mathbf{C}_t^{-\frac{1}{2}}$  is immediate via  $\mathbf{C}_t^{-\frac{1}{2}} = \mathbf{B}_t \mathbf{D}_t^{-1} \mathbf{B}_t^\top$  (it is done at the same time than the eigendecomposition of  $\mathbf{C}_t$  every  $\mathcal{O}(n)$  iterations with a complexity of  $\mathcal{O}(n^3)$ ).

*Cholesky-CMA.* An alternative to the previous algorithm was proposed in [16]. Instead of using the eigendecomposition of the covariance matrix to sample candidate solutions, it uses a decomposition of  $\mathbf{C}_t$  as  $\mathbf{C}_t = \mathbf{A}_t \mathbf{A}_t^\top$ . Indeed assume that  $\mathbf{A}_t$  is known, then sampling  $\mathbf{x}_t^i$  as  $\mathbf{m}_t + \sigma_t \mathbf{A}_t \mathbf{z}_t^i$  with  $\mathbf{z}_t^i \sim \mathcal{N}(0, \mathbf{I})$  results in a vector following  $\mathcal{N}(\mathbf{m}_t, \sigma_t^2 \mathbf{C}_t)$ . When  $\mathbf{A}_t$  is lower (or upper) triangular the decomposition is unique and called Cholesky factorization. However, in [16] the term Cholesky factorization is used without assuming that the matrix  $\mathbf{A}_t$  is triangular. We will continue to use Cholesky-CMA for the ensuing algorithm to be consistent with the previous algorithm name.



The key idea for the Cholesky-CMA is that instead of adapting the covariance matrix  $\mathbf{C}_t$ , the Cholesky factor  $\mathbf{A}_t$  is directly updated (and hence sampling does not require factorization a matrix). The method solely conducts the rank-one update of the covariance matrix,  $\mathbf{C}_{t+1} = (1 - c_1)\mathbf{C}_t + c_1\mathbf{p}_{t+1}^c[\mathbf{p}_{t+1}^c]^\top$ , by updating the matrix  $\mathbf{A}_t$  such that  $\mathbf{C}_{t+1} = \mathbf{A}_{t+1}\mathbf{A}_{t+1}^\top$ . Indeed, let  $\mathbf{v}_{t+1}$  be defined implicitly via  $\mathbf{A}_t\mathbf{v}_{t+1} = \mathbf{p}_{t+1}^c$ , then the update of  $\mathbf{A}_t$  reads

$$\mathbf{A}_{t+1} = \sqrt{1 - c_1}\mathbf{A}_t + \frac{\sqrt{1 - c_1}}{\|\mathbf{v}_{t+1}\|^2} \left( \sqrt{1 + \frac{c_1}{1 - c_1}\|\mathbf{v}_{t+1}\|^2} - 1 \right) \mathbf{p}_{t+1}^c \mathbf{v}_{t+1}^\top, \quad (2)$$

if  $\mathbf{v}_{t+1} \neq \mathbf{0}$  and  $\mathbf{A}_{t+1} = \sqrt{1 - c_1}\mathbf{A}_t$  if  $\mathbf{v}_{t+1} = \mathbf{0}$  (see [16, Theorem 1]). A similar expression holds for the inverse  $\mathbf{A}_{t+1}^{-1}$  (see [16, Theorem 2]). Sampling of a multivariate normal distribution using the Cholesky factor still requires  $\mathcal{O}(n^2)$  operations due to the matrix-vector multiplication. However, the Cholesky-CMA has been used as foundation to construct numerically more efficient algorithms as outlined below. Recently, a version of CMA using Cholesky factorization enforcing triangular shapes for the Cholesky factors has been proposed [8].

### 3.2 Large-Scale Variants of CMA-ES

The quadratic time and space complexity of CMA-ES (both the original and Cholesky variant) becomes critical with increasing dimension. This has motivated the development of large-scale variants with less rich covariance models, i.e., with  $o(n^2)$  parameters. Reducing the number of parameters reduces the memory requirements and, usually, the internal computational effort, because fewer parameters must be updated. It also has the advantage that learning rates can be increased. Hence, learning of parameters can be achieved in fewer *number of evaluations*. Given the model is still rich enough for the problem at hand, this further reduces the computational costs to solve it in particular even when the  $f$ -computation dominates the overall costs. Hence, in the best case scenario, reducing the number of parameters from  $n^2$  to  $n$  reduces the time complexity to solve the problem from  $n^2$  to  $n$  if  $f$ -computations dominate the computational costs and from  $n^4$  to  $n^2$  if internal computations dominate.

We review a few large-scale variants focussing on those benchmarked later in the paper.

*sep-CMA-ES* [14]. The separable CMA-ES restricts the full covariance matrix to a diagonal one and thus has a linear number of parameters to be learned. It loses the ability of learning the dependencies between decision variables but allows to exploit problem separability. The sep-CMA-ES achieves linear space and time complexity.

*VkD-CMA-ES* [2, 3]. A richer model of the covariance matrix is used in the VkD-CMA-ES algorithm where the eligible covariance matrices are of the form  $\mathbf{C}_t = \mathbf{D}_t(\mathbf{I} + \mathbf{V}_t\mathbf{V}_t^\top)\mathbf{D}_t$  where  $\mathbf{D}_t$  is a  $n$ -dimensional positive definite diagonal matrix and  $\mathbf{V}_t = [\mathbf{v}_t^1 \dots \mathbf{v}_t^k]$  where  $\mathbf{v}_t^i \in \mathbb{R}^n$  are orthogonal vectors [3]. The parameter  $k$  ranges from 0 to  $n - 1$ : when  $k = 0$  the method recovers the separable CMA-ES

while for  $k = n-1$  it recovers the (full)-CMA-ES algorithm. The elements of  $\mathbf{C}_{t+1}$  are determined by projecting the covariance matrix updated by CMA-ES given in (1) denoted as  $\hat{\mathbf{C}}_{t+1}$  onto the set of eligible matrices. This projection is done by approximating the solution of the problem  $\underset{(\mathbf{D}, \mathbf{V})}{\operatorname{argmin}} \|\mathbf{D} (\mathbf{I} + \mathbf{V}\mathbf{V}^\top) \mathbf{D} - \hat{\mathbf{C}}_{t+1}\|_F$

where  $\|\cdot\|_F$  stands for the Frobenius norm. This projection can be computed without computing  $\hat{\mathbf{C}}_{t+1}$ . The space complexity of VxD-CMA-ES is  $\mathcal{O}(nr)$  and the time complexity is  $\mathcal{O}(nr \max(1, r/\lambda))$ , where  $r = k + \mu + \lambda + 1$ . Note that the algorithm exploits both the rank-one and rank-mu update of CMA-ES as the projected matrices result from the projection of the matrix  $\hat{\mathbf{C}}_{t+1}$  updated with both updates.

A procedure for the online adaptation of  $k$  has been proposed in [2]. It tracks in particular how the condition number of the covariance matrix varies with changing  $k$ . The variant with the procedure of online adaptation of  $k$  as well as with fixed  $k = 2$  is benchmarked in the following. The VxD-CMA algorithm uses *Two Point Adaptation* (TPA) to adapt the step-size. The TPA is based on the ranking difference between two symmetric points around the mean along the previous mean shift.

*The limited-memory (LM) CMA* [11, 12]. The LM-CMA is inspired by the gradient based limited memory BFGS method [10] and builds on the Cholesky CMA-ES. If  $\mathbf{A}_0 = \mathbf{I}$ , setting  $a = \sqrt{1 - c_1}$  and  $b_t = \frac{\sqrt{1 - c_1}}{\|\mathbf{v}_{t+1}\|^2} \left( \sqrt{1 + \frac{c_1}{1 - c_1} \|\mathbf{v}_{t+1}\|^2} - 1 \right)$ , then (2) can be re-written as  $\mathbf{A}_{t+1} = a^t \mathbf{I} + \sum_{i=1}^t a^{t-i} b_{i-1} \mathbf{p}_i^c \mathbf{v}_i^\top$ . This latter equation is approximated by taking  $m$  elements in the sum instead of  $t$ . Initially,  $m$  was proposed to be fixed to  $\mathcal{O}(\log(n))$ . Later, better performance has been observed with  $m$  in the order of  $\sqrt{n}$  [11], imposing  $\mathcal{O}(n^{3/2})$  computational cost. Sampling can be done without explicitly computing  $\mathbf{A}_{t+1}$  and the resulting algorithm has  $\mathcal{O}(mn)$  time and space complexity. The choice of the  $m$  elements of the sum to approximate  $\mathbf{A}_{t+1}$  seems to be essential. In L-BFGS the last  $m$  iterations are taken while for LM-CMA the backward  $N_{\text{steps}} \times k$  iterations for  $k = 0, \dots, m-1$  are considered (that is we consider the current iteration, the current iteration minus  $N_{\text{steps}}$  and so on). The parameter  $N_{\text{steps}}$  is typically equal to  $n$ . Since  $\mathbf{A}_t \mathbf{v}_{t+1} = \mathbf{p}_{t+1}^c$ , the inverse factor  $\mathbf{A}_t^{-1}$  is employed for the computation of  $\mathbf{v}_{t+1}$ , but an explicit computation is not needed, similarly as for  $\mathbf{A}_t$ . To adapt the step-size, the LM-CMA uses the *population success rule* (PSR) [12].

A variant of LM-CMA was recently proposed, the LM-MA, which is however not tested here because (i) the code is not available online and (ii) the performance of LM-MA seems not to be superior to LM-CMA [13].

*The RmES* [9]. The idea for the RmES algorithm is similar to the LM-CMA algorithm. Yet, instead of using the Cholesky-factor, the update of  $\mathbf{C}_t$  is considered. Similarly as for LM-CMA, if  $\mathbf{C}_0 = \mathbf{I}$  and solely the rank-one update is used for CMA-ES we can write the update as  $\mathbf{C}_t = (1 - c_1)^m \mathbf{I} + c_1 \sum_{i=1}^m (1 - c_1)^{m-i} \hat{\mathbf{p}}_i^c \hat{\mathbf{p}}_i^{c^\top}$ . In RmES,  $m$  terms of the sum are considered and  $m = 2$  is advocated. Additionally, like in LM-CMA, the choice of terms entering the sum is by maintaining a temporal distance between generations. Sampling

of new solutions is done from the  $m$  vectors without computing the covariance matrix explicitly. The RmES adapts the step-size similarly to PSR.

A main difference to LM-CMA is that RmES is formulated directly on the covariance matrix, thus an inverse Cholesky factor is not needed. This does not improve the order of complexity, though, which is  $\mathcal{O}(mn)$  as in LM-CMA.

The presented algorithms do not of course form an exhaustive list of proposed methods for large-scale black-box optimization. We refer to [13] for a more thorough state-of-the-art and point out that our choice is driven by variants that currently appear to be the most promising or by variants like sep-CMA, important to give baseline performance.

## 4 Experimental Results

We assess the performance of implementations of the algorithms presented in the previous section on the `bbob-largescale` suite. We are particularly interested to identify the scaling of the methods, possible algorithm defects, and to quantify the impact of population size. Because we benchmark algorithm *implementations*, as opposed to mathematical algorithms, observations may be specific to the investigated implementation only.

*Experimental Setup.* We run the algorithms sep-CMA, LM-CMA, VkD-CMA, RmES on the default `bbob` test suite in dimensions 2, 3, 5, 10 and on the proposed `bbob-largescale` suite implemented in `COCO`. Additionally, we run the limited memory BFGS, L-BFGS, still considered as the state-of-the-art algorithm for gradient based optimization [10]. Gradients are estimated via finite-differences.

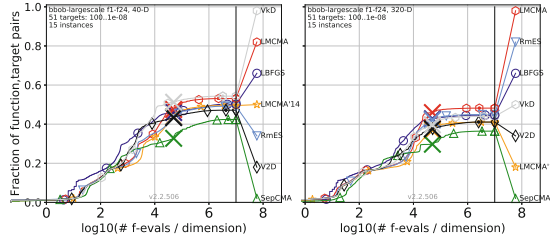
For VkD-CMA, the Python implementation from `pycma`, version 2.6.0, was used, for sep-CMA the version from [sites.google.com/site/ecjlmcma](https://sites.google.com/site/ecjlmcma), and for L-BFGS the optimization toolbox of `scipy` 0.12.1. We consider two versions of LM-CMA provided by the author at [sites.google.com/site/ecjlmcma](https://sites.google.com/site/ecjlmcma) and `.../lmcmaes` related to the articles [12] denoted LM-CMA'14 and [11] denoted LM-CMA. The implementation of RmES was kindly provided by its authors [9].

Experiments were conducted with default<sup>3</sup> parameter values of each algorithm and a maximum budget of  $5 \cdot 10^4 n$ . Automatic restarts are conducted once a default stopping criterion is met until the maximum budget is reached. For each function, fifteen instances are presented. For the first run and for all (automatic) restarts, the initial point was uniform at random between  $[-4, 4]^n$  for all algorithms, while the initial step-size was set to 2 for all CMA variants.

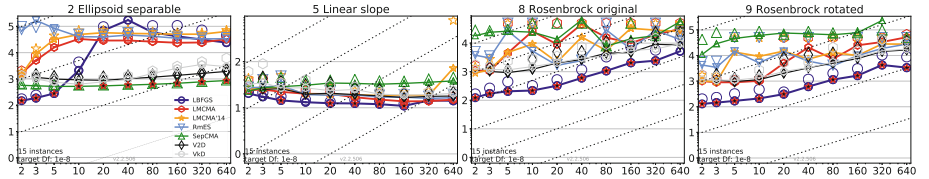
For LM-CMA, sep-CMA and RmES, population sizes of  $4 + \lfloor 3 \log n \rfloor$ ,  $2n + \lfloor 10/n \rfloor$  and  $10n$  were tested and the experiments were conducted for the same budget and instances. A suffix P2 (P10) is used to denote the respective algorithms. For VkD-CMA, a second experiment has been run where the number of vectors was fixed to  $k = 2$ , denoted as V2D-CMA.

---

<sup>3</sup> Except L-BFGS, where the `factr` parameter was set to 1.0 for very high precision.



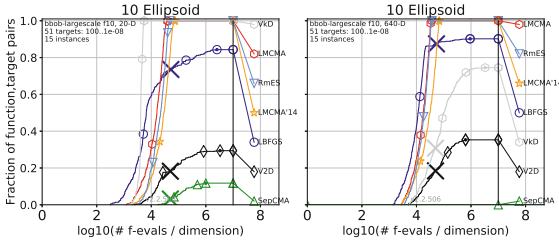
**Fig. 1.** Bootstrapped ECDF of the number of objective function evaluations divided by dimension (FEvals/D) for 51 targets in  $10^{[-8..2]}$  for all functions in 40-D (left) and 320-D.



**Fig. 2.** Scaling graphs: Average Runtime (aRT) divided by dimension to reach a target of  $10^{-8}$  versus dimension for selected functions. Light symbols give the maximum number of evaluations from the longest trial divided by dimension.

*Performance assessment.* We measure the number of function evaluations to reach a specified target function value, denoted as *runtime*, RT. The average runtime, aRT, for a single function and target value is computed as the sum of all evaluations in unsuccessful trials plus the sum of runtimes in all successful trials, both divided by the number of successful trials. For Empirical Cumulative Distribution Functions (ECDF) and in case of unsuccessful trials, runtimes are computed via simulated restarts [4] (bootstrapped ECDF). The *success rate* is the fraction of solved problems (function-target pairs) under a given budget as denoted by the y-axis of ECDF graphs. Horizontal differences between ECDF graphs represent runtime ratios to solve the same respective fraction of problems (though not necessarily the same problems) and hence reveal how much faster or slower an algorithm is.

*Overview.* A complete presentation of the experimental results is available at [cocoexprm.gforge.inria.fr](http://cocoexprm.gforge.inria.fr). Figure 1 presents for each algorithm the runtime distribution aggregated over all functions. Overall, the distributions look surprisingly similar in particular in larger dimension. After  $5 \cdot 10^4 n$  evaluations in 320-D, between 30% (sepCMA) and 46% (LMCMA) of all problems have been solved. In all dimensions, for a restricted range of budgets, the success rate of L-BFGS is superior to all CMA variants. The picture becomes more diverse with increasing budget where L-BFGS is outperformed by CMA variants. We emphasize that even domination over the entire ECDF does not mean that the algorithm is



**Fig. 3.** Bootstrapped ECDF of the number of objective function evaluations divided by dimension (FEvals/D) for 51 targets in  $10^{[-8..2]}$  for the ellipsoid function in 20-D and 640-D.

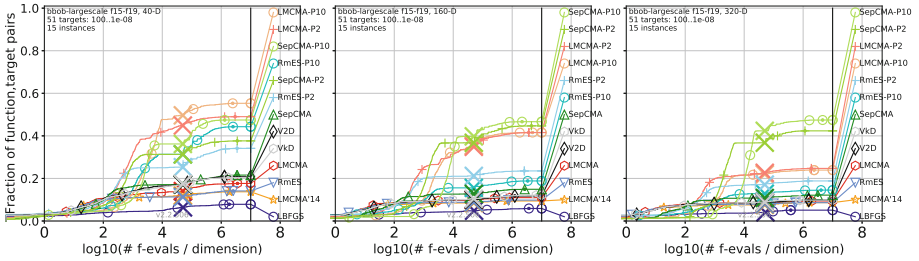
faster on every single problem, because runtimes are shown in increasing order *for each algorithm*, hence the order of problems as shown most likely differs.

Up to a budget of  $10^4 n$ , the performance similarity between LM-CMA and RmES is striking. The performance is almost identical on the [Sphere](#), [Ellipsoid](#), [Linear Slope](#) and [Sum of Different Powers](#) functions in dimensions equal or larger to 20. On the [Bent Cigar](#) function in dimensions greater or equal to 80 and for a budget larger than  $10^4 n$ , LM-CMA is notably superior to RmES.

*Scaling with dimension.* Fig. 2 shows the average runtime scaling with dimension on selected functions. On the [separable Ellipsoid](#) for  $n \geq 20$  sep-CMA with population size  $\geq 2n$  (not shown in Fig. 2) and VxD scale worse than linear. Starting from dimension 20, LM-CMA and RmES show runtimes of  $\text{aRT} \approx 2\text{--}7 \times 10^4 n$ . With default population size, sep-CMA performs overall best and is for  $n \geq 20$  even more than twenty times faster than L-BFGS. The latter scales roughly quadratically for small dimensions and (sub-)linear (with a much larger coefficient) for large dimensions. This behavior is a result of a transition when the dimension exceeds the rank (here 10) of the stored matrix. On the [linear](#) function, algorithms scale close to linear with a few exceptions. With population size  $2n + \lfloor 10/n \rfloor$  or larger (not shown in Fig. 2), the scaling becomes worse in all cases (which means a constant number of iterations is not sufficient to solve the “linear” problem). In particular, sep-CMA reveals in this case a performance defect due to a diverging step-size (which disappears with option ‘AdaptSigma’: ‘CMAAdaptSigmaTPA’), as verified with single runs. On both Rosenbrock functions, L-BFGS scales roughly quadratically.

*Restricting the model.* The particular case of the ill-conditioned non-separable ellipsoidal function in Fig. 3 illustrates interesting results: in 20D, VxD-CMA solves the function, i.e. reaches the best target value faster (by a factor of 10 at least) than any other method. In 640-D any other CMA variant with default parameter values except sep-CMA outperforms it.

On the [Ellipsoid](#) function only VxD-CMA scales quadratically with the dimension. All other algorithms either scale linearly or do not solve the problem for larger dimension. On the [Discus](#) function (with a fixed proportion of



**Fig. 4.** Bootstrapped ECDF of the number of objective function evaluations divided by dimension (FEvals/D) for 51 targets in  $10^{[-8..2]}$  for the group of multimodal functions with adequate structure in 40-D (left), 160-D (middle) and 320-D (right).

short axes), VxD-CMA slows down before to reach the more difficult targets and exhausts the budget. An unusual observation is that LM-CMA performs considerably better on the [Attractive Sector](#) function in the smallest *and largest dimensions*. We do not see this effect on LM-CMA'14, where the choice of the number of the direction vectors is smaller and random. Thus, these effects indicate the importance of properly choosing  $m$  [12]. Even though the covariance matrix model provided by VxD-CMA is richer, the method is outperformed by RmES and LM-CMA, e.g. on the [Discus](#) and [Ellipsoid](#) functions in dimension greater than 80. This suggests that  $k$  is adapted to too large values thereby impeding the learning speed of the covariance matrix.

*Fixed versus adapted  $k$ .* In order to investigate the effect of  $k$ -adaptation, we compare VxD-CMA with adaptive and fixed  $k = 2$ . Only in few cases the latter shows better performance. This is in particular true for the intrinsically not difficult to solve [Attractive Sector](#) function, indicating that the procedure of  $k$  adaptation could impose a defect.

*Impact of population size.* In Fig. 4, the effect of larger populations is illustrated for the multimodal functions with adequate global structure. The CMA variants with default population size and L-BFGS are clearly outperformed, solving less than half as many problems. That is, increased population size variants reach better solutions. Yet, the overall performance drops notably with increasing dimension. As expected, on the weakly-structured multimodal functions f20-f24, larger populations do not achieve similar performance improvements.

## 5 Discussion and Conclusion

This paper has (i) introduced a novel large-scale testbed for the [COCO](#) platform and (ii) assessed the performance of promising large-scale variants of CMA-ES compared to the quasi-Newton L-BFGS algorithm. We find that in all dimensions, L-BFGS generally performs best with lower budgets and is outperformed

by CMA variants as the budget increases. On multi-modal functions with global structure, CMA-ES variants with increased population size show the expected decisive advantage over L-BFGS. For larger dimension, the performance on these multi-modal functions is however still unsatisfying. The study has revealed some potential defects of algorithms (k-adaptation in V<sub>k</sub>D-CMA on the Attractive Sector, Ellipsoid and Discus) and has confirmed the impact and criticality of the choice of the  $m$  parameter in LM-CMA. The V<sub>k</sub>D-CMA that appears to be a more principled approach and includes a diagonal component and the rank- $\mu$  update of the original CMA-ES, overall outperforms LM-CMA and RmES in smaller dimension, while LM-CMA overtakes for the large budgets in larger dimensions. On single functions, the picture is more diverse, suggesting possible room for improvement in limited memory and V<sub>k</sub>D-CMA approaches.

**Acknowledgement.** The PhD thesis of Konstantinos Varelas is funded by the French MoD DGA/MRIS and Thales Land & Air Systems.

## References

1. Ait ElHara, O., Auger, A., Hansen, N.: Permuted orthogonal block-diagonal transformation matrices for large scale optimization benchmarking. In: Genetic and Evolutionary Computation Conference (GECCO 2016), pp. 189–196. ACM (2016)
2. Akimoto, Y., Hansen, N.: Online model selection for restricted covariance matrix adaptation. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 3–13. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_1](https://doi.org/10.1007/978-3-319-45823-6_1)
3. Akimoto, Y., Hansen, N.: Projection-based restricted covariance matrix adaptation for high dimension. In: Genetic and Evolutionary Computation Conference (GECCO 2016), pp. 197–204. Denver, USA, July 2016
4. Hansen, N., Auger, A., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: A platform for comparing continuous optimizers in a black-box setting (2016). [arXiv:1603.08785](https://arxiv.org/abs/1603.08785)
5. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Research Report RR-6829, INRIA (2009)
6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* **9**(2), 159–195 (2001)
7. Knight, J.N., Lunacek, M.: Reducing the space-time complexity of the CMA-ES. In: Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 658–665. ACM (2007)
8. Krause, O., Arbonès, D.R., Igel, C.: CMA-ES with optimal covariance update and storage complexity. In: NIPS Proceedings (2016)
9. Li, Z., Zhang, Q.: A simple yet efficient evolution strategy for large scale black-box optimization. *IEEE Trans. Evol. Comput.* (2017, accepted)
10. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. *Math. Program.* **45**(3), 503–528 (1989)
11. Loshchilov, I.: LM-CMA: an alternative to L-BFGS for large scale black-box optimization. *Evol. Comput.* **25**, 143–171 (2017)

12. Loshchilov, I.: A computationally efficient limited memory CMA-ES for large scale optimization. In: Genetic and Evolutionary Computation Conference (GECCO 2014), pp. 397–404 (2014)
13. Loshchilov, I., Glasmachers, T., Beyer, H.: Limited-memory matrix adaptation for large scale black-box optimization. CoRR abs/1705.06693 (2017)
14. Ros, R., Hansen, N.: A simple modification in CMA-ES achieving linear time and space complexity. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 296–305. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-87700-4\\_30](https://doi.org/10.1007/978-3-540-87700-4_30)
15. Sun, Y., Gomez, F.J., Schaul, T., Schmidhuber, J.: A linear time natural evolution strategy for non-separable functions. CoRR abs/1106.1998 (2011)
16. Suttorp, T., Hansen, N., Igel, C.: Efficient covariance matrix update for variable metric evolution strategies. Mach. Learn. **75**(2), 167–197 (2009)
17. Tang, K., et al.: Benchmark functions for the CEC 2008 special session and competition on large scale global optimization (2007)





# Design of a Surrogate Model Assisted (1 + 1)-ES

Arash Kayhani and Dirk V. Arnold<sup>✉</sup>

Faculty of Computer Science, Dalhousie University,  
Halifax, Nova Scotia B3H 4R2, Canada  
Arash.Kayhani@dal.ca, dirk@cs.dal.ca

**Abstract.** Surrogate models are employed in evolutionary algorithms to replace expensive objective function evaluations with cheaper though usually inaccurate estimates based on information gained in past iterations. Implications of the trade-off between computational savings on the one hand and potentially poor steps due to the inaccurate assessment of candidate solutions on the other are generally not well understood. We study the trade-off in the context of a surrogate model assisted (1 + 1)-ES by considering a simple model for single steps. Based on the insights gained, we propose a step size adaptation mechanism for the strategy and experimentally evaluate it using several test functions.

## 1 Introduction

Surrogate models have been proposed as an approach for evolutionary algorithms (EAs) to deal with optimization problems where each evaluation of the objective function requires a considerable amount of time or incurs a significant cost. Surrogate models are built using information on candidate solutions that have been evaluated previously using the true objective function. Evaluating a new candidate solution using a surrogate model yields a potentially inaccurate estimate of its true objective function value at a much lower cost than would be incurred in the exact evaluation. Surrogate modelling is useful if the benefit of reduced cost outweighs the potentially poorer steps made due to the inexact evaluation of candidate solutions.

Numerous approaches for incorporating surrogate models in EAs exist and have been comprehensively surveyed by Jin [8] and Loshchilov [11]. Algorithms usually are heuristic in nature, and potential consequences of design decisions are not always well understood. Most recent work on surrogate model assisted EAs considers relatively sophisticated algorithms. Strategies usually are evaluated by comparing the approach that uses surrogate modelling techniques with a corresponding algorithm that does not. A potential pitfall in such comparisons arises in connection with the use of large populations: if an algorithm for a given optimization problem uses a larger than optimal population size, then efficiency can be gained simply by using a trivial surrogate modelling approach that classifies a fraction of candidate solutions as poor, at no computational cost.

Clearly, the computational savings in this case are due to the effective reduction of the population size rather than to surrogate modelling.

We contend that it is desirable to develop an improved understanding of the potential implications of the use of surrogate modelling techniques, and that such an understanding can be gained by analyzing the behaviour of surrogate model assisted EAs using simple test functions that allow comparing the performance of the algorithms against a well established baseline. The contributions of this paper are as follows: after briefly reviewing related work in Sect. 2, in Sect. 3 we propose a simple model for surrogate model assisted EAs and use it to study the single-step behaviour of a surrogate model assisted (1 + 1)-ES<sup>1</sup> on quadratic sphere functions. We then use the insights gained to propose a step size adaptation mechanism for that algorithm in Sect. 4, and we evaluate its performance using several test functions. Section 5 concludes with a brief discussion and future work.

## 2 Related Work

The use of surrogate models in EAs can be traced back to the 1980s. Both Jin [8] and Loshchilov [11] present comprehensive surveys of the development of the field. Notable strategies include, though are not limited to, the Gaussian Process Optimization Procedure (GPOP) by Büche et al. [4] and the Local Meta-Model Covariance Matrix Adaptation Evolution Strategy (lmm-CMA-ES) by Kern et al. [9]. GPOP iterates the optimization of a Gaussian process based model of the objective using CMA-ES [7] and the subsequent evaluation and addition of the solution obtained to the training set. With computational cost determined by the number of (exact) objective function evaluations required to reach the optimal solution to within some target accuracy, Büche et al. [4] report a speed-up by a factor between four and five compared to CMA-ES on quadratic sphere functions and on Schwefel’s function, and smaller speed-ups on Rosenbrock’s function. lmm-CMA-ES use locally weighted regression models in connection with an approximate ranking procedure within the CMA-ES. With full quadratic models, Kern et al. [9] report a speed-up by a factor between two and eight compared to CMA-ES on unimodal functions, including the quadratic sphere, Schwefel’s function, and Rosenbrock’s function. More recent surrogate model assisted CMA-ES variants include the Surrogate-Assisted Covariance Matrix Adaptation Evolution Strategy (<sup>sm</sup>ACM-ES) by Loshchilov et al. [13] as well as several further algorithms surveyed and compared by Pitra et al. [14].

It is interesting to note that when considering unimodal test functions and comparing with relatively sophisticated black box optimization algorithms such as CMA-ES, the speed-ups reported as a result of using surrogate models appear to be a small factor (usually less than eight, frequently no larger than four), irrespective of the dimension of the problem. While larger speed-ups can be achieved when using surrogate models that perfectly fit the functions being optimized (e.g., quadratic models for optimizing quadratic functions), this observation is

---

<sup>1</sup> See Hansen et al. [6] for evolution strategy terminology.

not altogether unexpected in light of the performance bounds for black box optimization algorithms derived by Teytaud and Gelly [17].

A further interesting observation is that surrogate model assisted EAs tend to be relatively complicated and combine multiple heuristics for good performance. Notably, no surrogate model assisted version of the (1+1)-ES can be found in the literature. A seeming exception proposed by Chen and Zou [5] is not invariant to translations of the coordinate system — a property considered crucial for solving general unconstrained optimization problems — and does not include a mechanism for the adaptation of its step size. The Model Assisted Steady-State Evolution Strategy (MASS-ES) by Ulmer et al. [18] is a  $(\mu + \lambda)$ -ES that can in principle be run with  $\mu = \lambda = 1$ , but was not designed with those settings in mind and it is unclear whether its step size adaptation approach is effective under those conditions. Given the relative efficiency of the (1+1)-ES for unimodal black box problems and the relatively large body of knowledge regarding its convergence properties on convex functions, we argue that it is natural to ask to what degree the algorithm can be accelerated through the use of surrogate models, and how its step size can be adapted successfully.

### 3 Analysis

In order to gain a better understanding of potential implications of the use of surrogate models in EAs, in this section, we employ a simple model for the use of surrogate models. Specifically, we propose that an EA have the options of either evaluating a candidate solution accurately, at the cost of one objective function call, or of obtaining an inaccurate estimate of the solution’s objective function value at vanishing cost. For simplicity, we assume that the inaccurate objective function value is a Gaussian random variable with a mean that coincides with the candidate solution’s exact objective function value and some variance that models the accuracy of the surrogate model. As a result, techniques previously employed for the analysis of the behaviour of evolution strategies in the presence of Gaussian noise become applicable (see [1] and references therein). It would be straightforward to extend the analysis to biased surrogate models (i.e., models where the distribution mean differs from the exact objective function value). Models with a skew distribution of estimation errors could likely be considered based on analyses of the effects of non-Gaussian noise on the performance of evolution strategies (see [2]). Also not directly addressed in the present work are comparison based surrogate models. Loshchilov et al. [12] persuasively argue for such models in order to preserve invariance properties of comparison based optimization algorithms. We expect that an analysis analogous to what follows can be performed for such models.

We consider minimization of the quadratic sphere function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  with  $f(\mathbf{x}) = \mathbf{x}^T \mathbf{x}$  using a surrogate model assisted (1+1)-ES, where throughout this section the simple model described above substitutes for a “true” surrogate model. We initially consider a single iteration of the strategy and defer the discussion of step size adaptation until Sect. 4. The algorithm in each iteration generates single offspring candidate solution  $\mathbf{y} = \mathbf{x} + \sigma \mathbf{z}$ , where  $\mathbf{x} \in \mathbb{R}^n$  is

the best candidate solution obtained so far and is referred to as the parent,  $\mathbf{z} \in \mathbb{R}^n$  is a standard normally distributed random vector, and  $\sigma > 0$  is a step size parameter the adaptation of which is to be discussed below. The strategy uses the surrogate model to obtain an estimate  $f_\epsilon(\mathbf{y})$  of the objective function value that according to the above assumptions is a random variable with mean  $f(\mathbf{y})$  and some standard deviation  $\sigma_\epsilon > 0$ . Better surrogate models result in smaller values of  $\sigma_\epsilon$ . If  $f_\epsilon(\mathbf{y}) > f(\mathbf{x})$  (i.e., if the surrogate model suggests that the offspring candidate solution is inferior to the parent), then  $\mathbf{y}$  is discarded and the strategy proceeds to the next iteration; otherwise it computes  $f(\mathbf{y})$  at the cost of one objective function call and replaces  $\mathbf{x}$  with  $\mathbf{y}$  if and only if  $f(\mathbf{y}) < f(\mathbf{x})$  (i.e., if the offspring candidate solution truly is superior to the parent). In the terminology of Loshchilov [11] this procedure can be considered a natural implementation of preselection in the (1+1)-ES.

The expected step of the strategy can be studied by using a decomposition of  $\mathbf{z}$  first proposed by Rechenberg [15]. Vector  $\mathbf{z}$  is written as the sum of two components: one in direction of the negative gradient direction  $-\nabla f(\mathbf{x})$  and the other orthogonal to that. Due to symmetry, the length of the former component is standard normally distributed; the squared length of the latter is governed by a  $\chi^2$ -distribution with  $n-1$  degrees of freedom. The mean of that distribution is  $n-1$  and its coefficient of variation tends to zero as  $n$  increases. Referring to  $\delta = n(f(\mathbf{x}) - f(\mathbf{y})) / (2R^2)$ , where  $R = \|\mathbf{x}\|$ , as the normalized fitness advantage of  $\mathbf{y}$  over its parent, and introducing normalized step size  $\sigma^* = n\sigma/R$ , it follows

$$\begin{aligned} \delta &= \frac{n}{2R^2} (\mathbf{x}^T \mathbf{x} - (\mathbf{x} + \sigma \mathbf{z})^T (\mathbf{x} + \sigma \mathbf{z})) = \frac{n}{2R^2} (-2\sigma \mathbf{x}^T \mathbf{z} - \sigma^2 \|\mathbf{z}\|^2) \\ &\stackrel{n \rightarrow \infty}{=} \sigma^* z_1 - \frac{\sigma^{*2}}{2}, \end{aligned} \quad (1)$$

where  $z_1 = -\mathbf{x}^T \mathbf{z} / R$  is a standard normally distributed random variable representing the length of the component of  $\mathbf{z}$  in the direction of  $-\nabla f(\mathbf{x})$  and  $\stackrel{n \rightarrow \infty}{=}$  denotes convergence in distribution. Moreover, introducing  $\sigma_\epsilon^* = n\sigma_\epsilon / (2R^2)$ , the estimated normalized fitness advantage (i.e., the normalized fitness advantage estimated by using the surrogate model to evaluate  $\mathbf{y}$ ) is  $\delta_\epsilon = \delta + \sigma_\epsilon^* z_\epsilon$ , where  $z_\epsilon$  is standard normally distributed.

From the above, the estimated normalized fitness advantage is normally distributed with mean  $-\sigma^{*2}/2$  and variance  $\sigma^{*2} + \sigma_\epsilon^{*2}$  and thus has probability density

$$p_{\delta_\epsilon}(y) = \frac{1}{\sqrt{2\pi(\sigma^{*2} + \sigma_\epsilon^{*2})}} \exp\left(-\frac{1}{2} \frac{(y + \sigma^{*2}/2)^2}{\sigma^{*2} + \sigma_\epsilon^{*2}}\right). \quad (2)$$

Moreover, the probability density of  $z_1$  conditional on the estimated normalized fitness advantage  $\delta_\epsilon$  can be obtained as<sup>2</sup>

$$p_{z_1|\delta_\epsilon}(z|y) = \frac{\sqrt{\sigma^{*2} + \sigma_\epsilon^{*2}}}{\sqrt{2\pi\sigma_\epsilon^*}} \exp\left(-\frac{1}{2} \frac{((\sigma^{*2} + \sigma_\epsilon^{*2})z - \sigma^*(y + \sigma^{*2}/2))^2}{(\sigma^{*2} + \sigma_\epsilon^{*2})\sigma_\epsilon^{*2}}\right). \quad (3)$$

<sup>2</sup> Detailed derivations of Eqs. (3), (4), (5), and (6) can be found in a separate document at [web.cs.dal.ca/~dirk/PPSN2018addendum.pdf](http://web.cs.dal.ca/~dirk/PPSN2018addendum.pdf).

As  $\mathbf{y}$  is evaluated using the objective function if and only if it appears superior to the parent based on the surrogate model, we write  $p_{\text{eval}} = \text{Prob}[\delta_\epsilon > 0]$  for the probability of making a call to the objective function. From Eq. (2),

$$\begin{aligned} p_{\text{eval}} = \text{Prob}[\delta_\epsilon > 0] &= \int_0^\infty p_{\delta_\epsilon}(y) dy \\ &= \Phi\left(\frac{-\sigma^{*2}/2}{\sqrt{\sigma^{*2} + \sigma_\epsilon^{*2}}}\right), \end{aligned} \quad (4)$$

where  $\Phi(\cdot)$  denotes the cumulative distribution function of the standard normal distribution. Due to the accounting for computational costs,  $p_{\text{eval}}$  represents the expected cost per iteration of the algorithm. Similarly, as  $\mathbf{y}$  replaces  $\mathbf{x}$  if and only if  $\delta_\epsilon > 0$  and  $\delta > 0$ , we write  $p_{\text{step}} = \text{Prob}[\delta_\epsilon > 0 \wedge \delta > 0]$  for the probability of the offspring replacing the parent. From Eqs. (2) and (3),

$$\begin{aligned} p_{\text{step}} = \text{Prob}[\delta_\epsilon > 0 \wedge \delta > 0] &= \int_0^\infty p_{\delta_\epsilon}(y) \int_{\sigma^*/2}^\infty p_{z_1|\delta_\epsilon}(z|y) dz dy \\ &= \frac{1}{\sqrt{2\pi}} \int_{\sigma^*/2}^\infty e^{-z^2/2} \Phi\left(\frac{\sigma^*z - \sigma^{*2}/2}{\sigma_\epsilon^*}\right) dz \end{aligned} \quad (5)$$

as  $\delta > 0$  is equivalent to  $z_1 > \sigma^*/2$ . Finally, the expected value of the normalized change in objective function value

$$\Delta = \begin{cases} \delta & \text{if } \delta_\epsilon > 0 \text{ and } \delta > 0 \\ 0 & \text{otherwise} \end{cases}$$

from one iteration to the next can be computed as

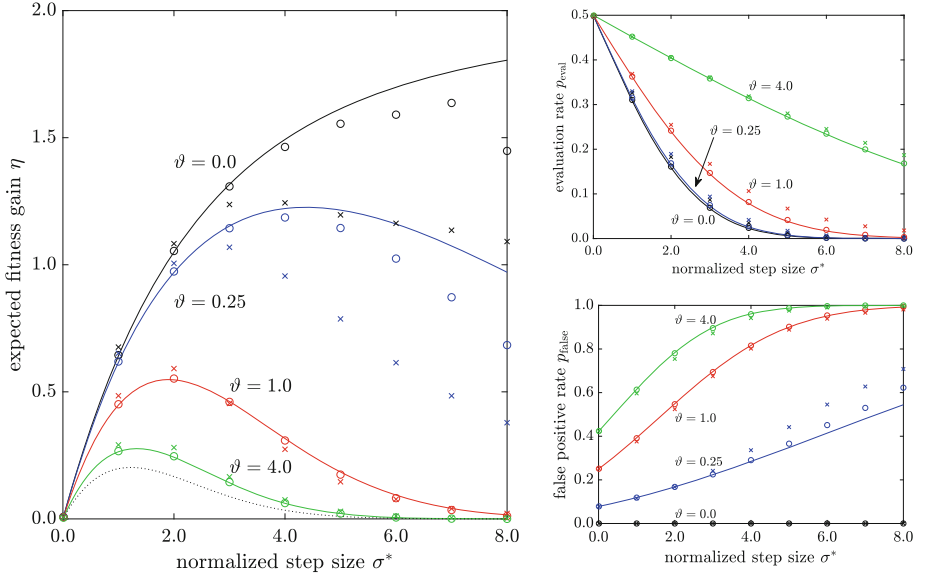
$$\begin{aligned} \text{E}[\Delta] &= \int_0^\infty p_{\delta_\epsilon}(y) \int_{\sigma^*/2}^\infty \left(\sigma^*z - \frac{\sigma^{*2}}{2}\right) p_{z_1|\delta_\epsilon}(z|y) dz dy \\ &= \frac{1}{\sqrt{2\pi}} \int_{\sigma^*/2}^\infty \left(\sigma^*z - \frac{\sigma^{*2}}{2}\right) e^{-z^2/2} \Phi\left(\frac{\sigma^*z - \sigma^{*2}/2}{\sigma_\epsilon^*}\right) dz. \end{aligned} \quad (6)$$

Equations (4), (5), and (6) describe the behaviour of the algorithm for  $n \rightarrow \infty$  and can serve as approximations for finite but not too small  $n$ .

If a step size adaptation mechanism and surrogate modelling approach are in place such that the distributions of  $\sigma^*$  and  $\sigma_\epsilon^*$  are independent of the iteration number, then the algorithm converges in expectation linearly with dimension-normalized rate of convergence

$$c = -\frac{n}{2} \text{E}\left[\log\left(\frac{f(\mathbf{x}_{t+1})}{f(\mathbf{x}_t)}\right)\right] = -\frac{n}{2} \text{E}\left[\log\left(1 - \frac{2\Delta}{n}\right)\right], \quad (7)$$

where subscripts denote iteration number. However, the rate of convergence does not account for computational cost as costs are incurred only in those iterations

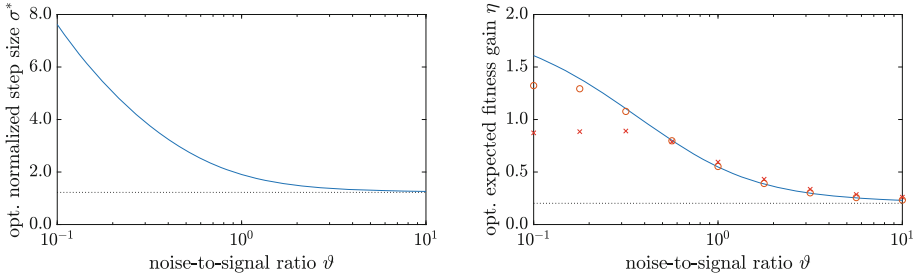


**Fig. 1.** Expected single step behaviour of the surrogate model assisted (1 + 1)-ES with unbiased Gaussian surrogate error. The solid lines represent results obtained analytically in the limit  $n \rightarrow \infty$ . The dots show values observed experimentally for  $n = 10$  (crosses) and  $n = 100$  (circles). The dotted line in the left hand plot illustrates the corresponding relationship for the (1 + 1)-ES without surrogate model assistance.

where a call to the objective function is made. We thus use  $\eta = c/p_{\text{eval}}$  (normalized rate of convergence per objective function call) as performance measure and refer to it as the expected fitness gain. For  $n \rightarrow \infty$  the logarithm in Eq. (7) can be linearized and the expected fitness gain is simply  $\eta = E[\Delta]/p_{\text{eval}}$ .

We define noise-to-signal ratio  $\vartheta = \sigma_{\epsilon}^*/\sigma^*$  as a measure for the quality of the surrogate model relative to the step size of the algorithm and in Fig. 1 plot the evaluation rate  $p_{\text{eval}}$ , the false positive rate  $p_{\text{false}} = 1 - p_{\text{step}}/p_{\text{eval}}$  (i.e., the probability of a candidate solution that is deemed superior by the surrogate model to be inferior to the parent according to the true objective function), and the expected fitness gain against the normalized step size. The lines show results obtained from Eqs. (4), (5), and (6). The dots show corresponding values observed in experiments with unbiased Gaussian surrogate error for  $n \in \{10, 100\}$  that have been obtained by averaging over  $10^7$  iterations. Deviations of the experimental measurements from values obtained in the limit  $n \rightarrow \infty$  are considerable primarily for large normalized step size and small noise-to-signal ratio.

It can be seen from Fig. 1 that for given noise-to-signal ratio, the evaluation rate of the algorithm decreases with increasing step size. For very small steps, one out of every two steps is deemed successful by the surrogate model; with larger steps, the algorithm becomes more “selective” when deciding whether to obtain an exact objective function value for a candidate solution. At the same time,

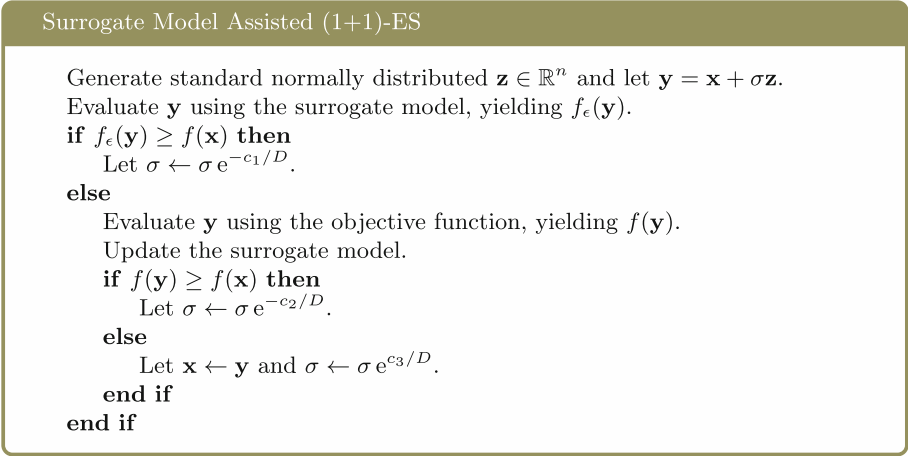


**Fig. 2.** Optimal normalized step size and resulting expected fitness gain of the surrogate model assisted  $(1 + 1)$ -ES plotted against the noise-to-signal ratio. The solid lines represent results obtained analytically in the limit  $n \rightarrow \infty$ . The dots show values observed experimentally for  $n = 10$  (crosses) and  $n = 100$  (circles). The dotted lines represent the optimal values for the  $(1 + 1)$ -ES without surrogate model assistance.

except for the case of zero noise-to-signal ratio, the false positive rate increases with increasing step size. The effect on the expected fitness gain (that accounts for computational costs) is such that for  $\vartheta > 0$  the gain peaks at a finite value of  $\sigma^*$ . With increasing noise-to-signal ratio, the expected fitness gain decreases. For  $\vartheta \rightarrow \infty$  the surrogate model becomes useless and the corresponding relationship for the  $(1 + 1)$ -ES without surrogate model assistance first derived by Rechenberg [15] is recovered (dotted line in the left hand plot in Fig. 1). That strategy achieves a maximal expected fitness gain of 0.202 at a normalized step size of  $\sigma^* = 1.224$ . For moderate values of  $\vartheta$ , the surrogate model assisted algorithm is capable of achieving much larger expected fitness gain values at larger step sizes (e.g., for  $\vartheta = 1.0$ , the maximal achievable expected fitness gain is 0.548 and is achieved at a normalized step size of  $\sigma^* = 1.905$ ). For  $\vartheta = 0$  (i.e., a perfect surrogate model), both the optimal normalized step size and the expected fitness gain with increasing step size tend to infinity. However, it is important to keep in mind that the analytical results have been derived in the limit of  $n \rightarrow \infty$  and merely are approximations in the finite-dimensional case. Figure 2 illustrates the dependence of the optimal normalized step size on the noise-to-signal ratio derived in the limit  $n \rightarrow \infty$  and shows values of the expected fitness gain achieved with that step size, both derived analytically for  $n \rightarrow \infty$  and measured experimentally for  $n \in \{10, 100\}$ . In the finite-dimensional cases the speed-up achieved through surrogate model assistance for small noise-to-signal ratios appears to top out between four and five for  $n = 10$  and between six and seven for  $n = 100$ . Notice that these values are roughly in line with speed-ups reported for surrogate model assisted CMA-ES variants mentioned in Sect. 2.

## 4 Step Size Adaptation and Experiments

In this section we propose a step size adaptation mechanism for the surrogate model assisted  $(1 + 1)$ -ES. We then evaluate the algorithm by using a



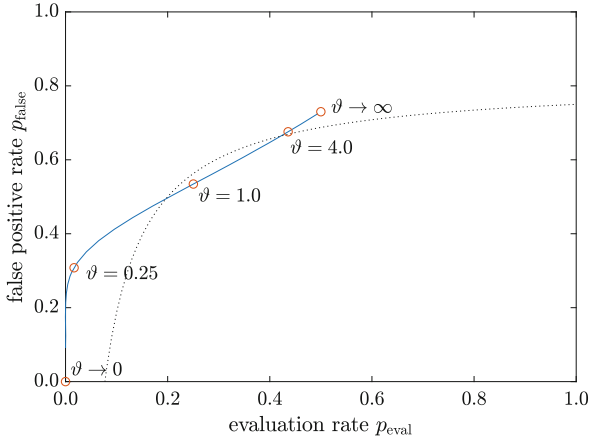
**Fig. 3.** Single iteration of the surrogate model assisted (1+1)-ES.

Gaussian Process surrogate model in place of the simple model for surrogate models employed in Sect. 3 and applying it to several test functions.

The step size of the (1+1)-ES is commonly adapted using the 1/5th rule proposed by Rechenberg [15]. That rule stipulates that the step size of the strategy can be adapted based on the “success rate” (i.e., the probability of the parent being replaced by the offspring candidate solution). If this rate exceeds one fifth then the step size is increased; if it is below one fifth then the step size is decreased. An ingenious implementation of that rule has been proposed by Kern et al. [10]; rather than approximating the success rate by counting successes over a number of iterations, increase the step size by multiplication with  $e^{0.8/D}$  in each iteration where the offspring is successful; decrease it by multiplication with  $e^{-0.2/D}$  whenever the parent prevails. Constant  $D$  controls the magnitude of the step size updates and according to Hansen et al. [6] can be set to  $\sqrt{1+n}$ . If one out of every five offspring generated is successful, then the step size updates cancel each other out on average and the logarithm of the step size remains unchanged. If the success rate exceeds one fifth, then increasing updates occur more frequently and the step size will systematically increase and vice versa.

The one-fifth rule is not suitable for the adaptation of the step size of the surrogate model assisted (1+1)-ES. From Fig. 1, there is no single value of either the evaluation rate or the false positive rate (both of which are observable) such that optimal values of the expected fitness gain are obtained near those rates, for all values of the noise-to-signal ratio that the strategy may operate under. However, we suggest that the step size can be adapted by considering a *combination* of those rates and propose the algorithm shown in Fig. 3. Nonnegative constants  $c_1$ ,  $c_2$ , and  $c_3$  remain to be determined below. The algorithm decreases the step size (potentially by differing rates) if the offspring candidate solution is rejected either based on the objective function value estimate provided by the





**Fig. 4.** False positive rate of the surrogate model assisted (1+1)-ES plotted against the evaluation rate. The solid line represents the optimally performing strategy under the conditions from Sect. 3, the dotted line the solution of Eq. (8) for  $c_1 = 0.05$ ,  $c_2 = 0.2$ ,  $c_3 = 0.6$ .

surrogate model or on the exact value returned by the objective function; it is increased if the offspring candidate solution is successful.

To choose values for the constants in the algorithm in Fig. 3, consider Fig. 4. The solid line in that plot has been obtained by using Eq. (6) to numerically determine the optimal normalized step size for values of the noise-to-signal ratio that vary from the very small to the very large. Corresponding values of the evaluation rate and the false positive rate were then obtained from Eqs. (4) and (5) and plotted against each other to obtain the solid curve in the plot. Considering the algorithm in Fig. 3, the step size will be unchanged in expectation if

$$-(1 - p_{\text{eval}})c_1 - p_{\text{eval}}p_{\text{false}}c_2 + p_{\text{eval}}(1 - p_{\text{false}})c_3 = 0. \quad (8)$$

The solution of Eq. (8) defines a branch of a hyperbola that is shown with a dotted line in Fig. 4 for the case that  $c_1 = 0.05$ ,  $c_2 = 0.2$ , and  $c_3 = 0.6$ . If the combination of evaluation rate and false positive rate falls above the dotted line, then the logarithm of the step size will decrease in expectation; if it falls below, then the step size will increase. One could attempt to tune parameters  $c_1$ ,  $c_2$ , and  $c_3$  to better match the solid curve in the figure. However, the likely inaccuracy of the simple model for surrogate models employed in Sect. 3 may render such efforts futile. For example, biased surrogate models would result in a shift of the solid curve either to the left or to the right.

In order to test the step size adaptation mechanism thus proposed, we use a set of five ten-dimensional test problems: sphere functions  $f(\mathbf{x}) = (\mathbf{x}^T \mathbf{x})^{\alpha/2}$  for  $\alpha \in \{1, 2, 3\}$  that we refer to as linear, quadratic, and cubic spheres, Schwefel's Problem 1.2 with  $f(\mathbf{x}) = \sum_{i=1}^n (\sum_{j=1}^i x_j)^2$  (a convex quadratic function with condition number of the Hessian approximately equal to 175.1; see [16]), and

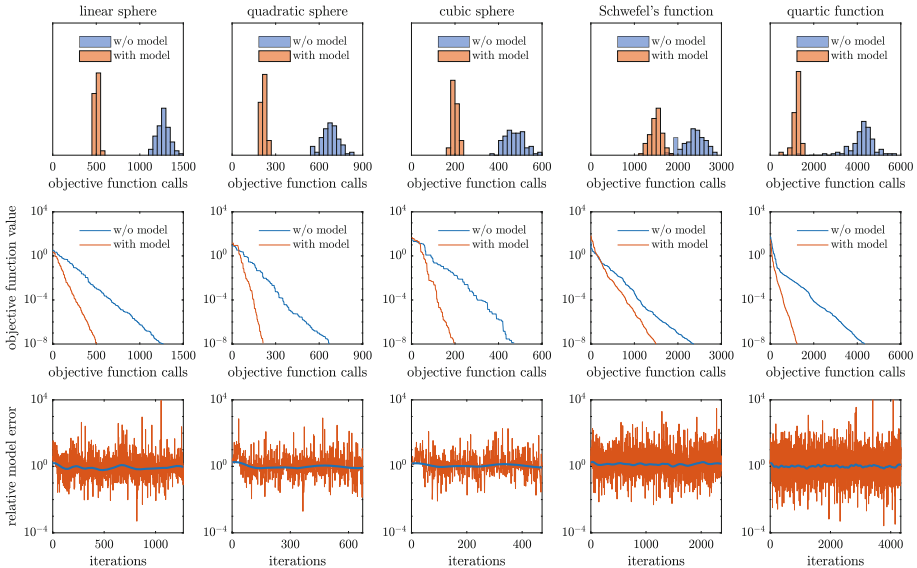
**Table 1.** Median test results.

	Median number of objective function calls		Speed-up
	Without model assistance	With model assistance	
Linear sphere	1270	503	2.5
Quadratic sphere	673	214	3.1
Cubic sphere	472	198	2.4
Schwefel’s function	2367	1503	1.6
Quartic function	4335	1236	3.5

$f(\mathbf{x}) = \sum_{i=1}^{n-1} [\beta(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$  (see [3]). For  $\beta = 100$  the latter function is the Rosenbrock function, the condition number of the Hessian of which at the optimizer exceeds 3,500, making it tedious to solve without adaptation of the shape of the mutation distribution. We use  $\beta = 1$  instead, resulting in the condition number of the Hessian at the optimizer being 49.0, and we refer to it as the quartic function. The optimal function value for all problems is zero. We conduct 101 runs for each problem, both for the surrogate model assisted (1 + 1)-ES and for the strategy that does not use model assistance. For surrogate models, as Büche et al. [4], we employ Gaussian processes. We use a squared exponential kernel and for simplicity set the length scale parameter of that kernel to  $8\sigma\sqrt{n}$ , where  $\sigma$  is the step size parameter of the evolution strategy. The training set consists of the 40 most recently evaluated candidate solutions. The surrogate model assisted algorithm does not start to use surrogate models until after iteration 40. All runs are initialized by sampling the starting point from a Gaussian distribution with zero mean and unit covariance matrix and setting the initial step size to  $\sigma = 1$ . Runs are terminated when a solution with objective function value below  $10^{-8}$  has been found.

Histograms showing the numbers of objective function calls used to solve the test problems to within the required accuracy are shown in the top row of Fig. 5, with median values represented in Table 1. The speed-up reported in the table is the median number of function evaluations used by the algorithm without surrogate model assistance divided by the corresponding number used by the surrogate model assisted (1 + 1)-ES. Speed-ups observed are between 1.6 for Schwefel’s function and 3.5 for the quartic function. Despite the simplicity of the surrogate models, the speed-up of 3.1 observed for the quadratic sphere function is not far below the maximal speed-up between four and five expected from Fig. 2. Speed-ups observed for the linear and cubic sphere functions are below that observed for the quadratic sphere, suggesting that the Gaussian process based models are more accurate for the latter than for the former. Encouragingly, the simple step size adaptation mechanism proved successful in all runs.

Convergence graphs for the median runs are shown in the middle row of Fig. 5. Eventually linear convergence appears to be achieved in all runs. The bottom row of the figure shows values of the relative model error  $|f(\mathbf{y}) - f_\epsilon(\mathbf{y})|/|f(\mathbf{y}) - f(\mathbf{x})|$ , where  $\mathbf{x}$  and  $\mathbf{y}$  are parent and offspring candidate solutions, respectively,



**Fig. 5.** Top row: Histograms showing the numbers of objective function calls used to solve the five test problems. Middle row: Convergence graphs for the median runs. Bottom row: Relative model error measured in the median runs.

observed in the median runs. The bold line in the centre of the plots represents the relative model error smoothed logarithmically by computing its convolution with a Gaussian kernel with a width of 40. We interpret the constancy of the smoothed curves as evidence that the algorithm operates under a relatively constant noise-to-signal ratio. Logarithmically averaging the relative objective model error across the median runs yields values between 0.786 and 0.989 for four of the five test problems, and a value of 1.292 for Schwefel's function.

## 5 Conclusions

To conclude, we have proposed unbiased Gaussian distributed noise as a model for surrogate modelling approaches. Using the model, we have presented an analysis of the behaviour of a surrogate model assisted (1+1)-ES on quadratic sphere functions. Based on that model we have proposed a step size adaptation mechanism for the surrogate model assisted (1 + 1)-ES and numerically evaluated it using a set of test functions. The mechanism successfully adapted the step size in all runs generated.

In future work, we will employ more sophisticated and possibly comparison based surrogate modelling approaches. Further goals include the development of adaptive approaches for setting the parameters  $c_1$ ,  $c_2$ , and  $c_3$  of the step size adaptation mechanism and the evaluation of the approach in the context of a (1 + 1)-ES with covariance matrix adaptation.

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

## References

1. Arnold, D.V.: Noisy Optimization with Evolution Strategies. Kluwer, Dordrecht (2002)
2. Arnold, D.V., Beyer, H.-G.: A general noise model and its effects on evolution strategy performance. *IEEE Trans. Evol. Comput.* **10**(4), 380–391 (2006)
3. Auger, A., Hansen, N., Perez Zepa, J.M., Ros, R., Schoenauer, M.: Experimental comparisons of derivative free optimization algorithms. In: Vahrenhold, J. (ed.) SEA 2009. LNCS, vol. 5526, pp. 3–15. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-02011-7\\_3](https://doi.org/10.1007/978-3-642-02011-7_3)
4. Büche, D., Schraudolph, N.N., Koumoutsakos, P.: Accelerating evolutionary algorithms with Gaussian process fitness function models. *IEEE Trans. Syst. Man Cybern. B Cybern. Part C* **35**(2), 183–194 (2005)
5. Chen, Y., Zou, X.: Performance analysis of a (1+1) surrogate-assisted evolutionary algorithm. In: Huang, D.-S., Bevilacqua, V., Premaratne, P. (eds.) ICIC 2014. LNCS, vol. 8588, pp. 32–40. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-09333-8\\_4](https://doi.org/10.1007/978-3-319-09333-8_4)
6. Hansen, N., Arnold, D.V., Auger, A.: Evolution strategies. In: Kacprzyk, J., Pedrycz, W. (eds.) *Springer Handbook of Computational Intelligence*, pp. 871–898. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-43505-2\\_44](https://doi.org/10.1007/978-3-662-43505-2_44)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* **9**(2), 159–195 (2001)
8. Jin, Y.: Surrogate-assisted evolutionary computation: recent advances and future challenges. *Swarm Evol. Comput.* **1**(2), 61–70 (2011)
9. Kern, S., Hansen, N., Koumoutsakos, P.: Local meta-models for optimization using evolution strategies. In: Runarsson, T.P., Beyer, H.-G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 939–948. Springer, Heidelberg (2006). [https://doi.org/10.1007/11844297\\_95](https://doi.org/10.1007/11844297_95)
10. Kern, S., Müller, S.D., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms – a comparative review. *Nat. Comput.* **3**(1), 77–112 (2004)
11. Loshchilov, I.: Surrogate-Assisted Evolutionary Algorithms. PhD thesis, Université Paris Sud - Paris XI (2013)
12. Loshchilov, I., Schoenauer, M., Sebag, M.: Comparison-based optimizers need comparison-based surrogates. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 364–373. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_37](https://doi.org/10.1007/978-3-642-15844-5_37)
13. Loshchilov, I., Schoenauer, M., Sebag, M.: Intensive surrogate model exploitation in self-adaptive surrogate-assisted CMA-ES. In: *Genetic and Evolutionary Computation Conference – GECCO 2013*, pp. 439–446. ACM Press (2013)
14. Pitra, Z., Bajer, L., Repický, J., Holena, M.: Overview of surrogate-model versions of covariance matrix adaptation evolution strategy. In: *Genetic and Evolutionary Computation Conference Companion*, pp. 1622–1629. ACM Press (2017)
15. Rechenberg, I.: *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Friedrich Frommann Verlag, Stuttgart (1973)

16. Schwefel, H.-P.: Numerical Optimization of Computer Models. Wiley, Hoboken (1981)
17. Teytaud, O., Gelly, S.: General lower bounds for evolutionary algorithms. In: Runarsson, T.P., Beyer, H.-G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 21–31. Springer, Heidelberg (2006). [https://doi.org/10.1007/11844297\\_3](https://doi.org/10.1007/11844297_3)
18. Ulmer, H., Streichert, F., Zell, A.: Model-assisted steady-state evolution strategies. In: Cantú-Paz, E. (ed.) GECCO 2003. LNCS, vol. 2723, pp. 610–621. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-45105-6\\_72](https://doi.org/10.1007/3-540-45105-6_72)



# Generalized Self-adapting Particle Swarm Optimization Algorithm

Mateusz Uliński, Adam Żychowski, Michał Okulewicz<sup>(✉)</sup>, Mateusz Zaborski,  
and Hubert Kordulewski

Faculty of Mathematics and Information Science, Warsaw University of Technology,  
Warsaw, Poland

M.Okulewicz@mini.pw.edu.pl

**Abstract.** This paper presents a generalized view on the family of swarm optimization algorithms. Paper focuses on a few distinct variants of the Particle Swarm Optimization and also incorporates one type of Differential Evolution algorithm as a particle's behavior. Each particle type is treated as an agent enclosed in a framework imposed by a basic PSO. Those agents vary on the velocity update procedure and utilized neighborhood. This way, a hybrid swarm optimization algorithm, consisting of a heterogeneous set of particles, is formed. That set of various optimization agents is governed by an adaptation scheme, which is based on the roulette selection used in evolutionary approaches. The proposed Generalized Self-Adapting Particle Swarm Optimization algorithm performance is assessed a well-established BBOB benchmark set and proves to be better than any of the algorithms its incorporating.

**Keywords:** Particle Swarm Optimization  
Self-adapting metaheuristics

## 1 Introduction

Since its introduction [9] and subsequent modifications [4, 18] Particle Swarm Optimization (PSO) algorithm has attracted many researchers by its simplicity of implementation and easiness of parallelization [13]. PSO has currently a several standard approaches [4], multiple parameter settings considered to be optimal [7] and successful specialized approaches [3]. PSO have also been tried with various topologies [8, 17], and unification [16] and adaptation schemes.

This paper brings various population based approaches together, and puts them in a generalized swarm-based optimization framework (GPSO). The motivation for such an approach comes from the social sciences, where diversity is seen as a source of synergy [10] and our adaptive approach (GAPSO) seeks an emergence of such a behavior within a heterogeneous swarm.

The remainder of this paper is arranged as follows. Section 2 introduces PSO and its adaptive modifications, together with discussing Differential Evolution (DE) algorithm and its hybridization with PSO. In Sect. 3 general overview of

the system’s construction is provided. Section 4 describes adaptation scheme and future system implementation details. Section 5 is devoted to a presentation of the experimental setup, in particular, the benchmark sets and parametrization of the methods used in the experiments. Experimental results are presented in Sect. 6. The last section concludes the paper.

## 2 Particle Swarm Optimization: Modification and Hybridization Approaches

This section reviews optimization algorithms used as basic building blocks within our generalized approach: PSO and DE. Initial paragraphs introduce the basic forms of the PSO and DE algorithms, while the following summarize the research on hybridizing those approaches and creating the adaptive swarm optimizers. Please bear in mind, that in all methods we shall be considering the optimization problem to be a minimization problem.

**Particle Swarm Optimization.** PSO is an iterative global optimization meta-heuristic method utilizing the ideas of swarm intelligence [9, 18]. The underlying idea of the PSO algorithm consists in maintaining the swarm of particles moving in the search space. For each particle the set of neighboring particles which communicate their positions and function values to this particle is defined. Furthermore, each particle maintains its current position  $x$  and velocity  $v$ , as well as remembers its historically best (in terms of solution quality) visited location. In each iteration  $t$ ,  $i$ th particle updates its position and velocity, according to formulas 1 and 2.

*Position update.* The position is updated according to the following equation:

$$\mathbf{x}_{t+1}^i = \mathbf{x}_t^i + \mathbf{v}_{t+1}^i. \quad (1)$$

*Velocity update.* In a basic implementation of PSO (as defined in [4, 18]) velocity  $v_t^i$  of particle  $i$  is updated according to the following rule:

$$\mathbf{v}_{t+1}^i = \omega \cdot \mathbf{v}_t^i + c_1 \cdot (\mathbf{p}_{best}^i - \mathbf{x}_t^i) + c_2 \cdot (\mathbf{neighbors}_{best}^i - \mathbf{x}_t^i) \quad (2)$$

where  $\omega$  is an inertia coefficient,  $c_1$  is a local attraction factor (cognitive coefficient),  $\mathbf{p}_{best}^i$  represents the best position (in terms of optimization) found so far by particle  $i$ ,  $c_2$  is a neighborhood attraction factor (social coefficient),  $\mathbf{neighbors}_{best}^i$  represents the best position (in terms of optimization) found so far by the particles belonging to the neighborhood of the  $i$ th particle (usually referred to as  $\mathbf{g}_{best}$  or  $\mathbf{l}_{best}$ ).

**Differential Evolution.** DE is an iterative global optimization algorithm introduced in [19]. DE’s population is moving in the search space of the objective function by testing the new locations for each of the specimen created by crossing over: (a) a selected  $\mathbf{x}^j$  solution, (b) solution  $\mathbf{y}_t^{(i)}$  created by summing up a scaled difference vector between two random specimen ( $\mathbf{x}^{(1)}$ ,  $\mathbf{x}^{(2)}$ ) with a third

solution ( $\mathbf{x}^{(i)}$ ). One of the most successful DE configurations is DE/rand/1/bin, where in each iteration  $t$ , each specimen  $\mathbf{x}_t^i$  in the population is selected and mutated by a difference vector between random specimens  $\mathbf{x}_t^{(i_1)}$  and  $\mathbf{x}_t^{(i_2)}$  scaled by  $F \in \mathbb{R}$ :

$$\mathbf{y}_t^{(i)} = \mathbf{x}_t^{(i)} + F \times (\mathbf{x}_t^{(i_2)} - \mathbf{x}_t^{(i_1)}) \quad (3)$$

Subsequently,  $y_t^{(3)}$  is crossed-over with  $x_t^{best}$  by binomial recombination:

$$\mathbf{u}_t^i = Bin_p(\mathbf{x}_t^{best}, \mathbf{y}_t^{(i)}) \quad (4)$$

Finally, the new location  $u_t^i$  replaces original  $x_t^i$  iff it provides a better solution in terms of the objective function  $f$ :

$$\mathbf{u}_t^i = \begin{cases} \mathbf{u}_t^i & \text{if } f(\mathbf{u}_t^i) < f(\mathbf{x}_t^i) \\ \mathbf{x}_t^i & \text{otherwise} \end{cases} \quad (5)$$

**Adaptive PSO Approaches.** While a basic version of the PSO algorithm has many promising features (i.e. good quality of results, easiness of implementation and parallelization, known parameters values ensuring theoretical convergence) it still needs to have its parameters tuned in order to balance its exploration vs. exploitation behavior [24]. In order to overcome those limitations a two-stage algorithm has been proposed [24]. That algorithm switches from an exploration stage into an exploitation stage, after the first one seems to be “burned out” and stops bringing much improvement into the quality of the proposed solution. Another adaptive approach that has been proposed for the PSO [23], identifies 4 phases of the algorithm: *exploration*, *exploitation*, *convergence*, and *jumping out*. The algorithm applies fuzzy logic in order to assign algorithm into one of those 4 stages and adapts its inertia ( $\omega$ ), cognitive ( $c_1$ ) and social ( $c_2$ ) coefficients accordingly. Finally, a heterogeneous self-adapting PSO has been proposed [14], but its pool of available behaviors has been limited only to the swarm-based approaches.

**PSO and DE Hybridization.** While DE usually outperforms PSO on the general benchmark tests, there are some quality functions for which the PSO is a better choice, making it worthwhile to create a hybrid approach [1,20]. Initial approaches on hybridizing PSO and DE consisted of utilizing DE mutation vector as an alternative for modifying random particles coordinates, instead of applying a standard PSO velocity update [5,21]. Another approach [22], consists of maintaining both algorithms in parallel and introducing an information sharing scheme between them with additional random search procedure. PSO and DE can also be combined in a sequential way [6,11]. In such an approach first the standard PSO velocity update is performed and subsequently various types of DE trials are performed on particle’s  $p_{best}$  location in order to improve it further.

### 3 Generalized Particle Swarm Optimization

This article follows the approach set for a social simulation experiment [15], by generalizing PSO velocity update formula (Eq. (2)) into a following form (with



$I$  being an indicator function and  $N_k(i\text{th})$  being a  $k$ th neighborhood of  $i$ th particle):

$$\begin{aligned} \mathbf{v}_{t+1}^i &= \omega \cdot \mathbf{v}_t^i + c_1 \cdot (\mathbf{p}_{best}^i - \mathbf{x}_t^i) \\ &+ \sum_{k=1}^{|\mathcal{N}|} \sum_{j=1, j \neq i}^{|\text{particles}|} I(j\text{th} \in N_k(i\text{th})) c'_{j,k} \cdot (\mathbf{p}_{best}^j - \mathbf{x}_t^i) \\ &+ \sum_{k=1}^{\mathcal{N}} \sum_{j=1, j \neq i}^{|\text{particles}|} I(j\text{th} \in N_k(i\text{th})) c''_{j,k} \cdot (\mathbf{x}_t^j - \mathbf{x}_t^i) \end{aligned} \quad (6)$$

In that way the social component extends into incorporating data from multiple neighbors and neighborhoods. The other part of generalization is not imposing an identical neighborhood structure over all particles, but letting each particle decide on the form of neighborhood. That way we take advantage of the agent-like behavior of swarm algorithms, where each individual is making its own decisions on the basis of simple rules and knowledge exchange (the other particles do not need to know behavior of a given particle, only its positions and sampled function values).

Proposed approach would be unfeasible if one would need to set up all  $c'_{j,k}$ 's and  $c''_{j,k}$ 's to individual values. Therefore we would rely on existing particles templates, where either all those coefficients would take the same value or most of them would be equal to zero. Our approach views  $c'_{j,k}$  and  $c''_{j,k}$  as functions. In most cases second index of  $c$  coefficients would be omitted, due to the fact that only a single neighborhood is considered.

In order to test the proposed generalized approach we have implemented five distinctive types of particles, coming from the following algorithms: Standard PSO (SPSO), Fully-Informed PSO (FIPSO), Charged PSO (CPSO), Unified PSO (UPSO), Differential Evolution (DE). Remainder of this section presents how each approach fits within the proposed GPSO framework.

**Standard Particle Swarm Optimization.** SPSO particle acts according to the rules of PSO described in Sect. 2 with a local neighborhood topology (with  $size \in \mathbb{Z}_+$  being its parameter). Therefore, the  $I$  function defining the neighborhood takes a following form:

$$I_{SPSO}(j\text{th} \in N(i\text{th})) = \begin{cases} 1 & |i - j| \leq size \\ 1 & |i - j| \geq |\text{particles}| - size \\ 0 & \sim \end{cases} \quad (7)$$

Particle changes its direction using  $l_{best}$  location. Therefore, all values of  $c'j$ 's and  $c''j$ 's are equal to 0 except the one corresponding to the particle with the best  $p_{best}$  value in the neighborhood.

$$c'_j = \begin{cases} 0 & f(\mathbf{p}_{best}^j) > f(\mathbf{l}_{best}) \\ X \sim U(o, c_2) & f(\mathbf{p}_{best}^j) = f(\mathbf{l}_{best}) \end{cases} \quad (8)$$

**Fully-Informed Particle Swarm Optimization.** FIPSO particle [12] steers its velocity to the location designated by all of its neighbors. All the best solutions found so far by the individual particles are considered with weights  $\mathcal{W}$  corresponding to the relative quality of those solutions. FIPSO particles utilize a complete neighborhood. Therefore, the indicator function  $I_{FIPSO}$  is equal to 1. The FIPSO particle is parametrized with a single value of an attraction coefficient  $c$ . Individual  $c'_j$ 's (and  $c_1$ ) follow the uniform distribution:

$$c'_j \sim U \left[ 0, \frac{c \cdot \mathcal{W}(f(\mathbf{p}_{best}^j))}{|\text{particles}|} \right] \quad (9)$$

**Charged Particle Swarm Optimization.** CPSO particle has been created for the dynamic optimization problems [3] and is inspired by the model of an atom. CPSO recognizes two particle types: neutral and charged. The neutral particles behave like SPSO particles. Charged particles, have a special component added to the velocity update equation. An  $i$ th charged particle has an additional parameter  $q$  controlling its repulse from other charged particles:

$$c''_{j,2} = -\frac{q^2}{\|\mathbf{x}_t^i - \mathbf{x}_t^j\|_2} \quad (10)$$

Charged particles repulse each other, so an individual sub-swarms are formed (as imposed by the neighborhood), which might explore areas corresponding to different local optima.

**Unified Particle Swarm Optimization.** UPSO particle is a fusion of the local SPSO and the global SPSO [16]. The velocity update formula includes both  $l_{best}$  and  $g_{best}$  solutions. In order to express that unification of global and local variants of SPSO the  $I$  indicator function takes the following form:

$$I_{UPSO}(j\text{th} \in N_k(i\text{th})) = \begin{cases} I_{SPSO} & k = 1 \\ 1 & \mathbf{p}_{best}^j \text{ is } \mathbf{g}_{best} \wedge k = 2 \\ 0 & \sim \end{cases} \quad (11)$$

Thus, there are two co-existing topologies of the neighborhood, which justifies the choice of the general formula for the GPSO (cf. Eq. (6)).

**Differential Evolution within the GPSO Framework.** While Differential Evolution (DE) [19] is not considered to be a swarm intelligence algorithm its behavior might be also fitted within the proposed framework GPSO. The reason for that is the fact that within the DE (unlike other evolutionary approaches) we might track a single individual as it evolves, instead of being replaced by its offspring.

DE/best/1/bin configuration and DE/rand/1/bin configurations are somewhat similar to the PSO with a  $g_{best}$  and  $l_{best}$  approaches, respectfully. The most important differences between DE and PSO behavior are the fact, that:

- DE individual always moves from the best found position ( $p_{best}$  in PSO), while PSO particle maintains current position, regardless of its quality,
- DE individual draws the 'velocity' (i.e. difference vector) from the global distribution based on other individuals location, while PSO particle maintains its own velocity.

Therefore, DE individual  $i$  movement might be expressed in the following way:

$$\mathbf{x}_{test}^{(i,t+1)} = Bin(v\mathbf{w} + (\mathbf{p}_{best} - \mathbf{x}_{test}^{(i,t)}), \mathbf{g}_{best}) \quad (12)$$

where  $v$  follows a probability distribution based on random individuals' locations ( $\mathbf{p}_{best}^{rand1}$  and  $\mathbf{p}_{best}^{rand2}$ ) and  $Bin$  is a binomial cross-over operator.

## 4 Adaptation Scheme

Different particle types perform differently on various functions. Moreover, different phases exist during optimization process. Some particle types perform better at the beginning, some perform better at the end of optimization algorithm's execution. Therefore, optimal swarm composition within GPSO framework should be designated in real-time. Swarm composition is modified by switching the behaviors of particles. Principle of work for adaptation scheme forming the Generalized Self-Adapting Particle Swarm Optimization (GPSO) is presented below.

The main idea is to promote particle types that are performing better than others. Adaptation is based on the quality of success. The adaptation utilizes roulette selection approach with probabilities proportional to success measure.

Let's assume that we have  $P$  particle types. Each particle changes its behavior every  $N_a$  iterations. Behavior is chosen according to a list of probabilities (each corresponding to one of  $P$  particles' types). Each particle has the same vector of probabilities. At the beginning all probabilities are set to  $\frac{1}{P}$ . Each  $N_a$  iterations probabilities vector is changing (adapting) according to the following scheme.

The average value of successes per each particle's type from the last  $N_a$  observations is determined. Value of success  $z_t^s$  in iteration  $t$  for particle  $s$  is presented in the following equation:

$$z_t^s = \max\left(0, \frac{f(\mathbf{p}_{best}^s) - f(\mathbf{x}_t^s)}{f(\mathbf{p}_{best}^s)}\right) \quad (13)$$

Let  $swarm_p$  be a set of  $p$  type particles from whole swarm. The average success  $\hat{z}_p$  of given  $swarm_p$  is obtained from  $S_p * N_a$  values, where  $S_p$  is the size of  $swarm_p$ . See the following equation:

$$\hat{z}_t^p = \frac{1}{S_p * N_a} * \sum_{t=T}^{T-N_a} \sum_{s \in swarm_p} z_t^s \quad (14)$$

This procedure produces  $P$  success values. Let us label them as  $z_1, z_2, \dots, z_P$ . Let  $Z$  be sum of given success values:  $Z = \sum_p^P z_p$ . So required vector of probabilities

is  $[\frac{z_1}{Z}, \frac{z_2}{Z}, \dots, \frac{z_P}{Z}]$ . Better average success induces greater probability of assigning given behavior to each particle. On top of the described approach an additional special rule is applied: at least one particle for each behavior has to exist. This rule prevents behaviors for being excluded from further consideration, as they might be needed in a latter phase of optimization process.

## 5 Experiment Setup

The GAPSO algorithm has been implemented in Java<sup>1</sup>. The project consists of individual particles behaviors, an adaptation scheme, a restart mechanism, hill-climbing local optimization procedure for “polishing” the achieved results, and a port to the test benchmark functions. Tests have been performed on 24 noiseless 5D and 20D test functions from BBOB 2017 benchmark<sup>2</sup>.

**Table 1.** Individual algorithms parameters.

Algorithm	Parameters settings	Reference
SPSO	$\omega : 0.9; c1, c2 : 1.2$	[4]
CPPO	$\omega : 0.9; c1, c2 : 1.2$	[3]
FIPSO	$\omega : 0.9; c : 4.5$	[12]
UPSO	$\omega : 0.9; c1, c2 : 1.2; u : 0.5$	[16]
DE	$crossProb : 0.5; varF : 1.4$	[19]

**Table 2.** Framework parameters.

Parameter	Value
Swarm size ( $S$ )	30
Number of neighbors ( $k$ )	5
Generations ( $G$ )	$10^6$
Number of PSO types ( $P$ )	5
Generations to adapt ( $N_a$ )	10
Generations to restart particle ( $N_{rp}$ )	15
Generations to restart swarm ( $N_{rs}$ )	200

**Parameters.** General GAPSO framework setup has been tuned on a small number of initial experiments, while the parameters of the individual optimization agents have been chosen according to the literature. The parameter values are presented in Tables 1 and 2.

**Restarts.** In order to fully utilize the algorithms’ potential within each of the tested methods a particle is restarted if for  $N_{rp}$  iterations at least one of these 2 conditions persisted: (a) particle is its best neighbor, (b) particle has low velocity (sum of squares of velocities in each direction is smaller than 1). Additionally, the whole swarm is restarted (each particle that belongs to it is restarted), if value of best found solution has not changed since  $N_{rs} \cdot D$ , where  $D$  is dimension of function being optimized.

**Local Optimization.** Finally (both in GAPSO and individual approaches), before swarm restart and after the last iteration of the population based algorithms a local *hill-climbing* algorithm is used for  $1000D$  evaluations, initialized with the best found solution.

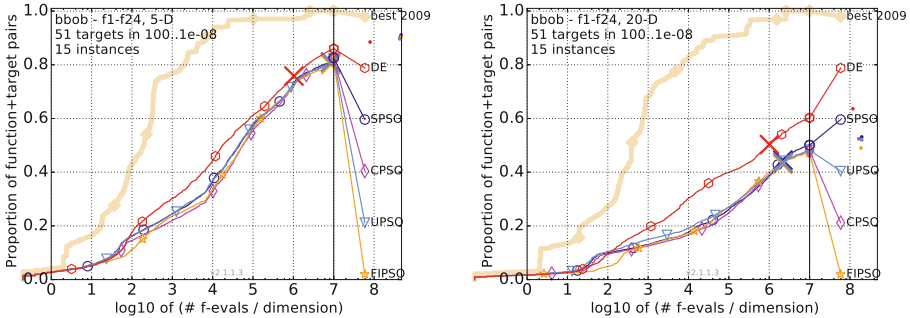
<sup>1</sup> <https://bitbucket.org/pl-edu-pw-mini-optimization/corpoalgorithm>.

<sup>2</sup> <http://coco.gforge.inria.fr/>.

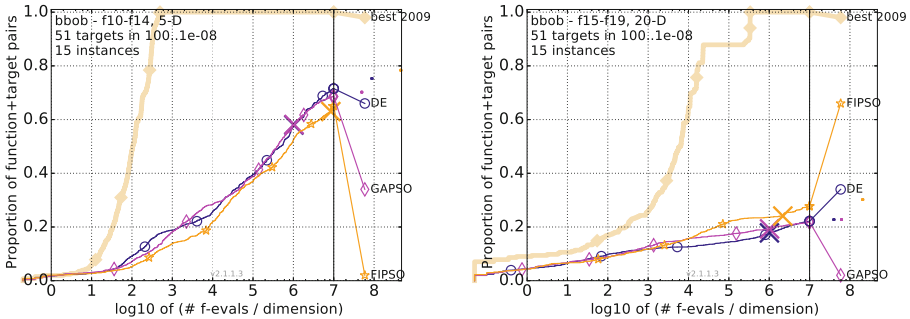
## 6 Results

Results of the experiments are presented on the figures generated within BBOB 2017 test framework, showing ECDF plots of optimization targets achieved on a log scale of objective function evaluations.

Left subplot in Fig. 1 shows efficiency of 5 individual algorithms used in GAPSO tested independently for 5D functions. It can be observed that DE is coinciding to optimum faster than each of the PSO approaches. Advantage of the DE is even more evident for 20D functions (right subplot in Fig. 1).

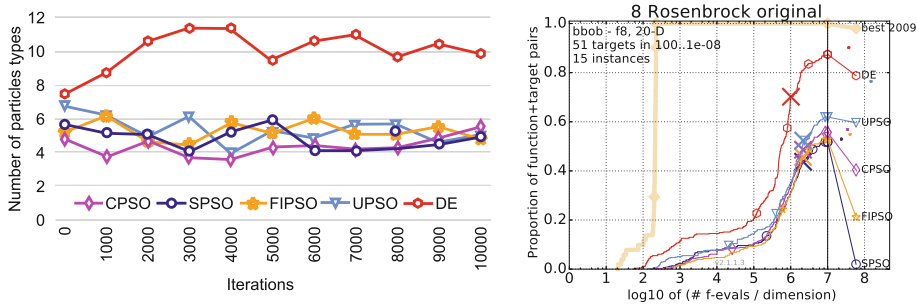


**Fig. 1.** Comparison of individual algorithms performance for all functions in 5 and 20 dimensions.

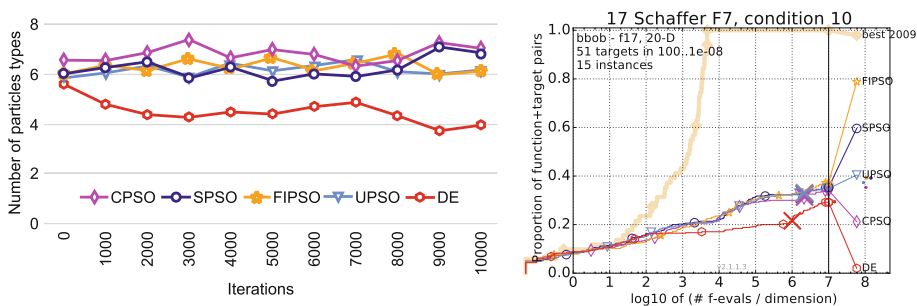


**Fig. 2.** Comparison of the best (DE) and the worst (FIPSO) individual algorithms with GAPSO for functions with high conditioning and unimodal in 5D (top) and multimodal functions with adequate global structure in 20D (right).

Subsequent charts (see Fig. 2) correspond to experiments carried out on selected algorithms with specified functions. In particular cases, differences in the effectiveness of algorithms can be observed. Left subplot in Fig. 2 shows advantage of DE algorithm in optimizing 5D unimodal functions with high conditioning. While another case, shown in right subplot in Fig. 2, presents FIPSO



**Fig. 3.** Average number of particles types in swarm compared with ECDF plot of individual algorithms performance for 20D Rosenbrock function.

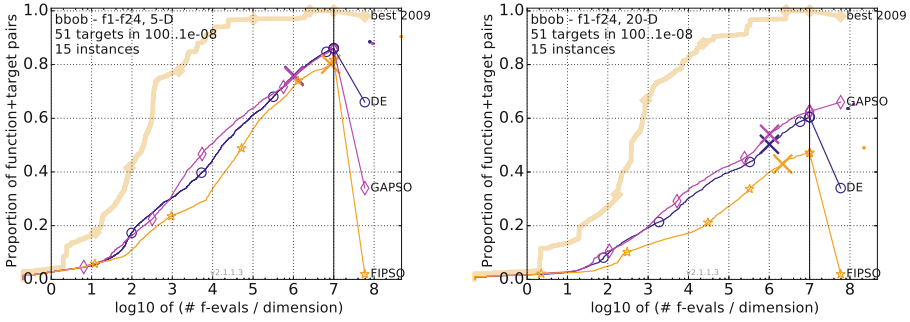


**Fig. 4.** Average number of particles types in swarm compared with ECDF plot of individual algorithms performance for 20D Schaffer function.

as an algorithm performing best for 20D multi-modal functions with adequate global structure. It can be observed that the proposed GAPSO algorithm remains competitive with both “clean” approaches.

Figures 3 and 4 present comparison of average number of particle’s behaviors and efficiency of homogeneous swarms for two selected functions. For Rosenbrock’s function (Fig. 3) DE swarm is significantly better than other kind of swarms and GAPSO algorithm adaptation method leads to greater number of DE particles in swarm. In the case when plain DE performance is worse than all the PSO-based approaches (see Fig. 4) GAPSO swarm contains significantly lower number of DE particles. It indicates that the proposed adaptation method controls the swarm composition according to the particular optimization function. It also can be observed that the performance of various PSO approaches is similar, and there is no noticeable difference between number of particles of particular kind.

Last experiment presents the overall effectiveness of the GAPSO performance on the whole set of 5D and 20D benchmark functions. Figure 5 presents the GAPSO results against the best (DE) and worst (FIPSO) performing algorithms. Results indicate that GAPSO has come out as a more effective approach, even though its adaptation has been performed during the optimization, and not beforehand.



**Fig. 5.** GAPS0 performance compared with the best (DE) and the worst (FIPSO) individual algorithms for all functions in 5D and 20D.

**Table 3.** Aggregated results for 15 independent runs on 24 noiseless test functions from BBOB 2017 benchmark. Number of functions for which given algorithm yielded best results (in term of average number of function evaluations) is presented in *best* columns. Numbers in brackets show how many of results are statistically significantly better according to the rank-sum test when compared to all other algorithms of the table with  $p = 0.05$ . *Target reached* is the number of trials that reached the final target:  $f_{opt} + 10^8$ .

Algorithm	5D		20D	
	Best	Target reached	Best	Target reached
CPSO	2 (0)	217	1 (0)	85
SPSO	1 (0)	221	2 (0)	91
FIPSO	2 (0)	211	4 (0)	83
UPSO	3 (0)	214	3 (0)	87
DE	6 (0)	173	4 (1)	117
GAPS0	10 (0)	172	8 (7)	120

Due to space limitations, Table 3 provides only aggregated results<sup>3</sup>. GAPS0 obtained best results (in terms of number of function evaluation) for 10 (5D) and 8 (20D) functions (out of 24), with 7 of those results being statistically significantly better than individual approaches. None of the other algorithms were statistically significantly better than GAPS0 for any function. These results show that proposed algorithm not only adapted to reach results as good as the best individual particles’ types, but also has the ability to outperform them.

Furthermore, GAPS0 stability other a different initial behavior probabilities vectors was examined. 7 types of vectors were considered: uniform (each behavior with the same probability), randomly generated vector and 5 vectors (one per each behavior) with probability equals 1 to one behavior and 0 for all other. Standard deviations obtained through all approaches on benchmark functions

<sup>3</sup> Detailed outcomes are available at <http://pages.mini.pw.edu.pl/~zychowska/gapso>.

were not significantly different than standard deviations for each approach separately. For all above options just after about 100 generations (10 adaptation procedures) numbers of particles with particular behaviors were nearly the same. It shows, that the proposed method's ability to gaining equilibrium - optimal behaviors (from the algorithm's perspective) is independent of the initial state of behavior probabilities vector.

## 7 Conclusions and Future Work

The proposed generalized GPSO view on the Particle Swarm Optimization made it possible to introduce various types of predefined behaviors and neighborhood topologies within a single optimization algorithm. Including an adaptation scheme in GAPS0 approach allowed to improve the overall performance over both DE individuals and PSO particles types on the test set of 24 quality functions. While the proposed approach remains inferior to algorithms such as CMA-ES [2], the adaptation scheme correctly promoted behaviors (particles) performing well on a given type of a function. It remains to be seen if other types of basic behaviors could be successfully brought into the GAPS0 framework and compete with the state-of-the-art optimization algorithms.

Our future research activities shall concentrate on testing more types of particles and detailed analysis about their cooperation by observing interactions between different particles behaviors in each generation. It would be especially interesting to evaluate a performance of some quasi-Newton method, brought into the framework of GPSO, as it could utilize the already gathered samples of the quality (fitness) function. Furthermore, other adaptation and evaluation schemes can be considered and compared with proposed method.

## References

1. Araújo, T.D.F., Uturbey, W.: Performance assessment of PSO, DE and hybrid PSODE algorithms when applied to the dispatch of generation and demand. *Int. J. Electrical Power Energy Syst.* **47**(1), 205–217 (2013)
2. Beyer, H.G., Sendhoff, B.: Simplify your covariance matrix adaptation evolution strategy. *IEEE Trans. Evol. Comput.* **21**(5), 746–759 (2017)
3. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) *Evolutionary Computation in Dynamic and Uncertain Environments*. SCI, vol. 51, pp. 29–49. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-49774-5\\_2](https://doi.org/10.1007/978-3-540-49774-5_2)
4. Clerc, M.: *Standard particle swarm optimisation* (2012)
5. Das, S., Abraham, A., Konar, A.: Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. *Advances of Computational Intelligence in Industrial Systems*. SCI, vol. 116, pp. 1–38. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78297-1\\_1](https://doi.org/10.1007/978-3-540-78297-1_1)
6. Epitropakis, M., Plagianakos, V., Vrahatis, M.: Evolving cognitive and social experience in particle swarm optimization through differential evolution: a hybrid approach. *Inf. Sci.* **216**, 50–92 (2012)



7. Harrison, K.R., Ombuki-Berman, B.M., Engelbrecht, A.P.: Optimal parameter regions for particle swarm optimization algorithms. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 349–356. IEEE (2017)
8. Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **35**(6), 1272–1282 (2005)
9. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. IV, pp. 1942–1948 (1995)
10. Köppel, P., Sandner, D.: Synergy by Diversity: Real Life Examples of Cultural Diversity in Corporation. Bertelsmann-Stiftung, Gütersloh (2008)
11. Liu, H., Cai, Z., Wang, Y.: Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Appl. Soft Comput.* **10**(2), 629–640 (2010)
12. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. *IEEE Tran. Evol. Comput.* **8**(3), 204–210 (2004)
13. Mussi, L., Daolio, F., Cagnoni, S.: Evaluation of parallel particle swarm optimization algorithms within the CUDA architecture. *Inf. Sci.* **181**(20), 4642–4657 (2011)
14. Nepomuceno, F.V., Engelbrecht, A.P.: A self-adaptive heterogeneous pso for real-parameter optimization. In: 2013 IEEE Congress on Evolutionary Computation, pp. 361–368. IEEE, June 2013
15. Okulewicz, M.: Finding an optimal team. In: FedCSIS Position Papers, pp. 205–210 (2016)
16. Parsopoulos, K.E., Vrahatis, M.N.: Unified particle swarm optimization for solving constrained engineering optimization problems. In: Wang, L., Chen, K., Ong, Y.S. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 582–591. Springer, Heidelberg (2005). <https://doi.org/10.1007/11539902-71>
17. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. *Swarm Intell.* **1**(1), 33–57 (2007)
18. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0040810>
19. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.* **11**(4), 341–359 (1997)
20. Thangaraj, R., Pant, M., Abraham, A., Bouvry, P.: Particle swarm optimization: hybridization perspectives and experimental illustrations. *Appl. Math. Comput.* **217**(12), 5208–5226 (2011)
21. Zhang, W.J., Xie, X.F.: DEPSO: hybrid particle swarm with differential evolution operator. In: SMC 2003 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483). vol. 4, pp. 3816–3821. IEEE (2003)
22. Zhang, C., Ning, J., Lu, S., Ouyang, D., Ding, T.: A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. *Oper. Res. Lett.* **37**(2), 117–122 (2009)
23. Zhan, Z.-H., Zhang, J., Li, Y., Chung, H.H.: Adaptive particle swarm optimization. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **39**(6), 1362–1381 (2009)
24. Zhuang, T., Li, Q., Guo, Q., Wang, X.: A two-stage particle swarm optimizer. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). vol. 2, pp. 557–563. IEEE, June 2008



# PSO-Based Search Rules for Aerial Swarms Against Unexplored Vector Fields via Genetic Programming

Palina Bartashevich<sup>1</sup>(✉), Illya Bakurov<sup>2</sup>, Sanaz Mostaghim<sup>1</sup>,  
and Leonardo Vanneschi<sup>2</sup>

<sup>1</sup> Faculty of Computer Science, University of Magdeburg, Magdeburg, Germany  
{palina.bartashevich,sanaz.mostaghim}@ovgu.de

<sup>2</sup> NOVA IMS, Universidade Nova de Lisboa, 1070-312 Lisbon, Portugal  
{ibakurov,lvanneschi}@novaims.unl.pt

**Abstract.** In this paper, we study Particle Swarm Optimization (PSO) as a collective search mechanism for individuals (such as aerial micro-robots) which are supposed to search in environments with unknown external dynamics. In order to deal with the unknown disturbance, we present new PSO equations which are evolved using Genetic Programming (GP) with a semantically diverse starting population, seeded by the Evolutionary Demes Despeciation Algorithm (EDDA), that generalizes better than standard GP in the presence of unknown dynamics. The analysis of the evolved equations shows that with only small modifications in the velocity equation, PSO can achieve collective search behavior while being unaware of the dynamic external environment, mimicking the zigzag upwind flights of birds towards the food source.

**Keywords:** Particle swarm optimization · Vector fields · Semantics  
Genetic Programming · EDDA

## 1 Introduction

This paper considers the Vector Field PSO (VF-PSO) algorithm [2], which is supposed to be used as a collective search mechanism for a swarm of aerial micro-robots acting under the influence of external unknown dynamics (such as wind) performed by vector fields. The main challenge is that the external dynamics of the environment are unknown to the swarm. As a result, due to the influence of unknown external factors, the velocity vectors of the individuals (e.g. robots) are constantly influenced by the external dynamics, and therefore the whole process of the collective search is misled. A previous study [2] suggested the use of a multi-swarm approach and collection of information about unknown dynamics by an explorer population, while another swarm, called optimizer, uses this information to correct their movements during the search process. However, maintaining explorers in some environments might not be possible, e.g. sensors

not working under certain conditions, or loss of the connection between explorers and optimizers, so they can not access the collected information (which is a realistic assumption for aerial robotic systems).

The goal of this paper is to find out whether it is possible to obtain a reasonably good approximation of the global optimal solution using only PSO equations in complete unawareness of the vector fields structure without explorer population, and how such velocity equations (further denoted as VFPS) should be designed in order to show the collective resistance to the unknown external dynamics. To answer these questions, we refer to previous research [1], which has only investigated the possibility of evolving such particle swarm equations using Geometric Semantic Genetic Programming (GSGP) [13]. GSGP has recently attracted much attention in the GP community due to its operators which in contrast to the traditional ones tend to be more effective as they induce a unimodal error surface for any supervised learning problem [21]. However, in [1] it was indicated that using pure GSGP for evolving new PSO equations is not as efficient as using standard Genetic Programming (GP), while the mixture of the GSGP and GP mutation operators was shown to be beneficial to produce high-quality individuals in the presence of unknown dynamics. The mixture of the above mentioned mutation operators in [1] was simulated by the Evolutionary Demes Despeciation Algorithm (EDDA) [20].

In this work, we use the findings of [1] to generate better VFPS equations, which are more robust to the unknown external dynamics than the standard PSO velocity equation. The study performed in this paper differs from [1], as the key aspect of the current work is the analysis and study of the evolved equations themselves and not of the evolutionary process of getting these equations. Besides, for the evolution of the equations, we use standard GP with EDDA on its top only as initialization technique to seed a better GP run.

So far, several applications of the GP to evolve new search algorithms have been already studied in the literature. The Extended Particle Swarms (XPSO) project [5, 11, 15, 16] demonstrated that by using GP it is possible to automatically evolve new PSO algorithms that perform better than standard ones designed by humans. Besides the framework of the XPSO project, some work regarding the evolution of PSO structures was also carried out by Diođan and Oltean in [6, 7]. Several studies have also applied GP to investigate other population-based metaheuristic optimizations apart from PSO: for instance, Runka et al. [17] and Taveres et al. [18] applied GP to evolve probabilistic rules used in Ant Colony optimization [8] to update its pheromone trails, and Di Chio et al. [4] used GP to evolve particle swarm equations for group-foraging problems in the simulation of behavioral ecology problems.

The paper is organized as follows. We describe the background about EDDA and VF-PSO in Sect. 2. In Sect. 3, we replicate and provide more detailed descriptions of the semantics introduced in [1] that allows us to use EDDA for the evolution of VFPS equations. Section 4 presents the experimental settings and Sect. 5 along with Sect. 6 discusses the obtained VFPS equations. The paper is concluded in Sect. 7.

## 2 Background

**Evolutionary Demes Despeciation Algorithm.** EDDA [20] is developed as a biologically inspired semantics-based initialization technique for GSGP to create not only a syntactically but also a semantically diverse starting population. According to EDDA, the initial population is seeded with good quality individuals that have been previously evolved for few generations in other populations (called demes). For instance, a population of  $N$  individuals will be composed of the best individuals found by  $N$  different demes, which are evolved independently by using different operators. In [20] EDDA was applied to seed GSGP runs, which generated solutions with comparable or even better generalization ability and of significantly smaller size compared to traditional GSGP, where part of the demes was evolved using operators of standard GP, while the another using GSGP. However, according to [21], with only Geometric Semantic mutation (further denoted as GSM) it is already possible to obtain the same performance as using GSGP with both crossover and mutation operators and in some cases even outperform it. Thus, in this paper we use EDDA to seed standard GP, where the GSGP part of EDDA demes is evolved using only GSM in order to keep the individuals of reasonable size. A definition of the term semantics used in this paper is described in Sect. 3, taking into account the fact that we are developing an application aimed at evolving search algorithms.

**Vector Field PSO.** VF-PSO is a collective search mechanism based on the movement of a population of particles, motivated by the real case scenario of aerial micro-robots, acting in an  $n$ -dimensional search space  $S$  under the influence of unknown dynamic conditions (e.g. wind influence). It performs a variation of the standard PSO algorithm [10], which is based on two simple rules for updating the particles  $i$  velocity  $\mathbf{v}_i(t)$  and its corresponding position  $\mathbf{x}_i(t) \in S$  at time step  $t$ :

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) + \sum_{k=0}^K \mathbf{VF}(\mathbf{g}^k) \quad (1)$$

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\phi_1(\mathbf{x}_i^{best}(t) - \mathbf{x}_i(t)) + c_2\phi_2(\mathbf{x}^g(t) - \mathbf{x}_i(t)) \quad (2)$$

The only difference from standard PSO is the additional term in Eq. 1, which incorporates vector fields  $\mathbf{VF}$  to induce the unknown external conditions in the search space  $S$ . According to the definition, a vector field is a function that takes any point in the space  $\mathbf{x} \in S$  and assigns a vector  $\mathbf{VF}(\mathbf{x})$  to it:  $\mathbf{x} \mapsto \mathbf{VF}(\mathbf{x})$ . In a discrete setting, a vector field is defined on the grid of cells  $\{\mathbf{g}^k\}_{k=1}^M \in G \subset S$ , where for each cell  $\mathbf{g}^k, k \in \{1..M\}$  an associated vector exists as a piecewise constant field  $\mathbf{VF}(\mathbf{g}^k)$ . Following this, the sum of vectors at  $K \ll M$  cells, which particle  $i$  intersects along the movement from its previous position  $\mathbf{x}_i(t)$  to the next  $\mathbf{x}_i(t+1)$ , is added to the current position of the particle to simulate the drift in Eq. 1, where  $\mathbf{x}_i(t) \in \mathbf{g}^0$  and  $\mathbf{x}_i(t+1) \in \mathbf{g}^K$ . More description on the other terms used in the equations described above is provided in the first part of the following Sect. 3.

### 3 Semantics for VFPS Evolution in EDDA

As for the evolution of VFPS equations in this work we use standard GP, but with EDDA for the initialization, we have to take into consideration that EDDA uses in the evolution part of the demes which is evolved by GSM. Thus, we have to introduce corresponding semantics. In order to do this, in this section we extend and provide more detailed description of the semantics for VFPS evolution, which was first introduced in [1].

Referring to Pawlak et al. [14], in GP semantics is typically contextualized within a specific *programming task* that is to be solved in a given *program set*  $P$ . Thus, in order to introduce the definition of semantics used in this paper, we have to define what the *program set*  $P$  and the *programming task* are in our case.

**Program Set.** In our case, we consider GP individuals as acceleration vectors  $\mathbf{a}_i$  of the Eq. 2, where  $\mathbf{a}_i(t) = \mathbf{v}_i(t+1) - w\mathbf{v}_i(t)$ . To define the program set  $P$ , we must specify the set of terminal symbols  $\mathcal{T}$  and the set of primitive functions  $\mathcal{F}$  used to code this type of individual, i.e.  $\mathbf{a}_i$ .

According to Eq. 2, an acceleration function  $\mathbf{a}_i$  can be considered as a composition of three atomic elements: the current positions of the particles  $\mathbf{x}_i$ , the local best positions of the particles  $\mathbf{x}_i^{pbest}$  and the global best of the swarm  $\mathbf{x}^g$ . These three elements are part of the terminal set  $\mathcal{T}$ . Furthermore, we are also interested in the “old” velocity vector of the particle, i.e.  $\mathbf{v}_i(t)$ , in the sense of frictional force and its possible combinations with constants, like in air drag or fluid friction. According to Long et al. [12], for small particles air resistance is approximately proportional to their velocities  $\mathbf{v}_i$  and can be expressed in the form:  $\mathbf{F}_{drag} = -b\mathbf{v}_i$  or  $\mathbf{F}_{drag} = -b\mathbf{v}_i^2$ , where  $b$  is a constant that depends on the properties of the particular type of air or fluid. Additionally, we also consider the center of the swarm  $\mathbf{x}^c$  and its diversity  $\sigma_x$  in the terminal set, as in [15], along with the limited set of permissible constants  $C \in \{-0.5, 0.5\}$ , as well as a set of random vectors  $\mathbf{R}(0, 1)$ , each of which contains uniformly distributed numbers different for each dimension, within the range  $[0, 1]$ . So, our terminal set is:  $\mathcal{T} = \{\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_i^{pbest}, \mathbf{x}^g, \mathbf{x}^c, \sigma_x, C, \mathbf{R}(0, 1)\}$ , where  $\mathbf{x}_i$  is the current position of the particle  $i$ ,  $\mathbf{v}_i$  is its “old” velocity,  $\mathbf{x}_i^{pbest}$  is the best location previously visited by particle  $i$ ,  $\mathbf{x}^g$  is the best location visited by the entire swarm and  $\sigma_x$  is the average distance of each particle  $\mathbf{x}_i$  to the center of mass  $\mathbf{x}^c$ .

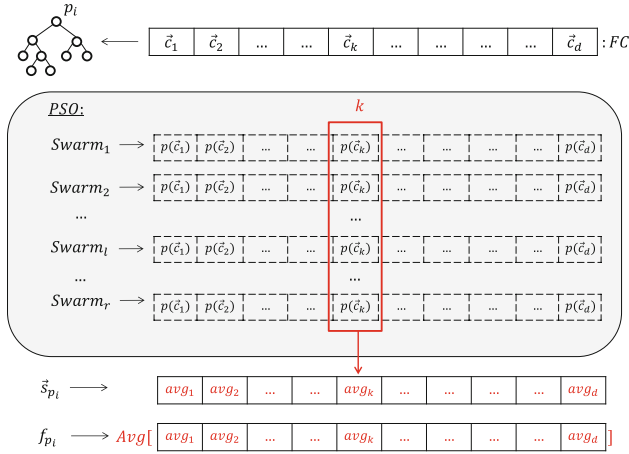
It is worth pointing out that Eq. 2 contains two different random vectors  $\phi_1$  and  $\phi_2$ , which are reflected in the terminal set  $\mathcal{T}$  by the set of random vectors  $\mathbf{R}(0, 1)$ . The role of  $\phi_1$  and  $\phi_2$  in PSO is to diversify the particles, keeping them from moving exactly towards the global  $\mathbf{x}^g$  and personal best  $\mathbf{x}_i^{pbest}$  positions. On the other hand, previous studies [3, 9, 22] showed that the iterated multiplication of random factors  $\phi_1$  and  $\phi_2$  can also lead to delay in convergence and attraction to inappropriate directions, dissimilar direction changes in different vectors and, as the result, to a limitation in the particle movements.

Moreover, among all the evolved acceleration equations reported by Poli et al. [5, 15, 16], only one has three different random vectors –  $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$ . This

equation is:  $\mathbf{R}_1(\mathbf{x}^g - \mathbf{x}_i) - 0.75 \mathbf{R}_2 \mathbf{R}_1 \mathbf{x}_i \mathbf{x}_g^2 - 0.25 \mathbf{R}_3 \mathbf{R}_2 \mathbf{R}_1 \mathbf{x}_i \mathbf{x}_g$ . All the other provided individuals have mostly none or only one random vector  $\mathbf{R}$ . Considering that the probability of more frequent usage of random vectors in constructing VFPS equations by means of GSM is increased due to the larger size of the evolved individuals, we limit the number of different random vectors in the terminal set up to three:  $\{\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3\} \in \mathbf{R}(0, 1)$ .

Function set is introduced as:  $\mathcal{F} = \{+, -, *, \sin\triangleleft, \cos\triangleleft, \langle \cdot, \cdot \rangle, \times, LF\}$ , where, given vectors  $\mathbf{e}_1 = (e_1^1, e_1^2)$  and  $\mathbf{e}_2 = (e_2^1, e_2^2)$ ,  $*$  is the element-by-element multiplication of two vectors, i.e. as the result we get another vector  $\mathbf{e}_1 * \mathbf{e}_2 = (e_1^1 * e_2^1, e_1^2 * e_2^2)$ ;  $\langle \cdot, \cdot \rangle$  is the dot product of two vectors, i.e. the result is a scalar equal to  $\langle \mathbf{e}_1, \mathbf{e}_2 \rangle = e_1^1 * e_2^1 + e_1^2 * e_2^2$ ;  $\times$  is the cross product of two vectors, i.e. the result is a scalar equal to  $\mathbf{e}_1 \times \mathbf{e}_2 = \|\mathbf{e}_1\| * \|\mathbf{e}_2\| * \sin\triangleleft(\mathbf{e}_1, \mathbf{e}_2)$ , where  $\|\mathbf{e}_1\| = \sqrt{(e_1^1)^2 + (e_1^2)^2}$  is the magnitude of the corresponding vector;  $\cos\triangleleft$  is a cosine of the angle between two vectors calculated as  $\cos\triangleleft(\mathbf{e}_1, \mathbf{e}_2) = \frac{\mathbf{e}_1 \cdot \mathbf{e}_2}{\|\mathbf{e}_1\| * \|\mathbf{e}_2\|}$ , so  $\sin\triangleleft(\mathbf{e}_1, \mathbf{e}_2) = \sqrt{1 - \cos\triangleleft(\mathbf{e}_1, \mathbf{e}_2)^2}$ ; and, finally,  $LF$  performs a logistic function applied to each component of the vector  $\mathbf{e}$ , i.e.  $LF(\mathbf{e}) = (\frac{1}{1 + \exp(-e^1)}, \frac{1}{1 + \exp(-e^2)})$ .

**Programming Task.** According to the definition in [14], the programming task, usually denoted as  $(FC, f)$ , is defined by a set of fitness cases  $FC \subseteq X \times O$  and a fitness function  $f : \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ . A *fitness case*  $FC$  is a pair consisting in a program input  $in \in X$  and a corresponding output  $out \in O$ . In this sense,  $FC$  represents the *training set* of the programming task. Figure 1 describes the programming task used in this paper. In our case, *fitness case*  $FC$  represents



**Fig. 1.** Semantics  $\mathbf{s}_p$  of the individual  $p$  on a problem class  $FC$  represented as a vector of average outputs calculated for each of the problem instances  $\{c_j\}_{j=1}^d := FC$  within  $r$  runs, where  $Swarm_l$  denotes the fixed initial population within the corresponding run for all problem instances. Fitness function  $\mathbf{f}_p$  of individual  $p$  is performed as the average of the elements in its semantics vector  $\mathbf{s}_p$ .

a certain optimization function  $h(\mathbf{x} - \mathbf{c}) : \mathbb{R}^n \rightarrow \mathbb{R}$ , where  $\mathbf{c}$  is a corresponding optimum solution. As an input for the program  $p_i$ , we consider a  $d$ -dimensional vector  $\{\mathbf{c}_j\}_{j=1}^d \in \mathbb{R}^n$ ,  $\mathbf{c}_j \neq \mathbf{c}_i \neq 0$  of different global optimal solutions, representing a class of shifted functions  $FC := \{h(\mathbf{x} - \mathbf{c}_j)\}_{j=1}^d$ . So, an output is the vector of respective function values at the global best position  $x_j^g$  found using  $p_i : \mathbf{c}_j \mapsto h(\mathbf{x}_j^g - \mathbf{c}_j), \forall j \in \{1, \dots, d\}$ . Evaluation of  $p_i$  inside VF-PSO lasts for  $r$  runs on each element of the input vector  $\{\mathbf{c}_j\}_{j=1}^d$ . The average of the outputs within runs  $avg_j = \sum_1^r p_i(\mathbf{c}_j), \forall j \in \{1, \dots, d\}$  defines the elements of the vector  $\mathbf{s}_{p_i}$ , which represents a point in the semantic space  $\mathcal{S}$  and *the semantics* of an individual  $p_i$  on a problem class  $FC$ . Without loss of generality, in this paper we considered problem classes, whose function values at the global optimum are equal to zero (see Sect. 4). Thus, as the *target* is a zero vector  $\mathbf{t} \in \mathbb{R}^d$ , the *fitness* for individual  $p_i$  in our case is the average of the elements of its semantics  $\mathbf{s}_{p_i}$ :  $f_{p_i} = \frac{1}{d} \sum_{k=1}^d [\mathbf{s}_{p_i}(k) - \mathbf{t}(k)] = \frac{1}{d} \sum_{k=1}^d \mathbf{s}_{p_i}(k)$ .

## 4 Experimental Study

The objective of the experimental study is to evolve and to analyze new VFPS equations which are able to find the approximate global optimum solution in total unawareness of the external dynamics. Before discussing the experimental results, let us briefly present the experimental settings.

**Experimental VF-PSO Settings.** Following [2, 5, 15, 16], in our experiments we consider the following problem classes:  $FC^1 - Sphere$ ,  $FC^2 - Rosenbrock$  and  $FC^3 - Ackley$ . As in [2], the vector fields are considered in the two-dimensional search space  $S : [-15.0, 15.0] \times [-15.0, 15.0]$  and their function descriptions can be found in Table 1. Every problem instance was considered in combination with each of the five vector fields (VF). Additionally, we also consider the case without vector field (denoted further as VF0). Given that the influence of the considered vector fields is weaker near the origin of the Cartesian system (see the vector fields descriptions in Table 1), we shifted the global optimum for each of the problem instances to the left upper corner of the search space  $S$ , namely to the region  $\Omega : [-11.0, -9.0] \times [9.0, 11.0]$ . We define  $d = 10$  problem instances in each problem class  $FC^i : \{h^i(\mathbf{x} - \mathbf{c}_j)\}_{j=1}^d$ , where  $\mathbf{c}_j$  is a random vector from

**Table 1.** Function descriptions of the vector fields

“Cross”	$\mathbf{VF1}(x_1, x_2) = (x_2, x_1)$
“Rotation”	$\mathbf{VF2}(x_1, x_2) = (-x_2, x_1)$
“Sheared”	$\mathbf{VF3}(x_1, x_2) = (x_1 + x_2, x_2)$
“Wave”	$\mathbf{VF4}(x_1, x_2) = (-\sin(x_2), \cos(x_1 \cdot x_2 - x_1^2))$
“Tornado”	$\mathbf{VF5}(x_1, x_2) = (-x_1 - x_2, x_1)$

the given interval  $\Omega$  and  $h^i(\mathbf{x})$  is the objective function of the corresponding problem class  $FC^i$ . In the training set, we use VF-PSO with 10 particles, whose initial positions are chosen uniformly at random in the whole search space  $S$ . Initial velocity is set to 0. The components of the velocity vector are constrained within  $\mathbf{v} \in [-2.0, +2.0]$ . The inertia weight  $w$  is equal to 0.6. Acceleration coefficients are  $C_1, C_2 = 1$ . Each VFPS has been run for  $N_{max} = 30$  iterations on each problem instance for  $r = 5$  times. In the testing phase, the number of runs was increased up to  $r = 100$  with  $N_{max} = 50$  each. The population was enlarged up to 20 particles and the global optimal solution was fixed at  $c = (-10.0, 10.0)$  for all problem classes. The results of the testing phase are reported in Sect. 6 and compared in terms of the best function values obtained during the all iterations, and the *success rate*, which indicates the percentage of runs, where this fitness is smaller than a certain threshold  $\epsilon$  by the end of the search process. We used  $\epsilon = 0.1$ .

**Experimental EDDA and GP Settings.** For EDDA we used 100 demes, each of which containing 100 individuals. The individuals in each deme were themselves initialized by means of the ramped half-and-half method, with maximum initial depth equal to 3. After initialization, each deme was left to evolve for 5 generations using a given set of genetic operators. In EDDA,  $m\%$  of demes use GSM (GSGP demes), while the remaining  $(100 - m\%)$  use standard genetic operators (GP demes). In our experiments, we test  $m \in \{25, 50, 75\}$ . A maximum depth limit of 5 is imposed only during the evolution of the GP demes, while in the main evolutionary process (MEP) this limit was enlarged to 11. After 5 generations, the best individual is selected and copied into the population that constitutes the MEP. The mutation step  $ms$  of GSM in the GSGP demes was randomly generated with uniform probability in  $[0,1]$  at each mutation event, following [19]. The codomain of the possible outputs of the randomly generated trees in GSM was bounded in  $[0, 1]$  by wrapping them inside a logistic function, as in [19]. To select parents for variation, tournaments of 5% of the population size were used and survival was elitist, as it always copied the best individual into the next population. While evolving VFPS in the GP demes and in MEP, the probability of applying crossover and mutation was set to 0.9 and 0.1 respectively.

## 5 Evolved VFPS

In this section, we present and discuss the best 5 force VFPS equations, that we were able to evolve, in terms of the performance on the training set. All these equations were obtained using EDDA-50% as an initialization technique, and standard GP in the main evolutionary process.

**VFPS1** was evolved on the ‘‘Sphere-VF1’’ (Cross) problem:

$$\mathbf{a}_i = \mathbf{R}_1(\mathbf{x}_i^{pbest} - \mathbf{x}_i) + \sigma_x^2(\mathbf{x}^g - \mathbf{x}_i) \quad (3)$$



Interestingly, it has a similar structure to the acceleration of the standard PSO (i.e. both cognition and social components are present), but with divergence factor for the deterministic social component. Thus, when the particles are too sparse in a swarm, the social component has more weight, so the particles are more attracted by the global best position than by the personal best local ones. On the other hand, when the particles are denser, the influence of the social component in the swarm decreases and the particles are more attracted by their respective local best position, reproducing local search behavior.

**VFPS2** was evolved on the “Sphere-VF1” (Cross) problem:

$$\mathbf{a}_i = (\mathbf{R}_1 + (\mathbf{x}^g - \mathbf{x}_i)) \langle \mathbf{x}_i^{pbest} + \mathbf{v}_i, \mathbf{x}_i^{pbest} + \mathbf{v}_i \rangle \quad (4)$$

It includes a separate independent random component, which is added to the deterministic social component, while their sum is weighted by the squared length value of  $(\mathbf{x}_i^{pbest} + \mathbf{v}_i)$  vector, obtained as its dot product by itself.

**VFPS3** was evolved on the “Sphere-VF3” (Sheared) problem:

$$\mathbf{a}_i = \sigma_x \left( (\mathbf{x}_i^{pbest} - \mathbf{x}_i) + (\mathbf{v}_i + \sigma_x)(\mathbf{x}^g - \mathbf{x}_i) \right) \quad (5)$$

This equation is interesting, because it is completely deterministic, and both cognition and social components are present, as in standard PSO. Similarly to **VFPS1**, it contains a divergence factor, but contrarily to **VFPS1**, this factor influences both the cognition and social components.

**VFPS4** was evolved on the “Sphere-VF1” (Cross) problem:

$$\mathbf{a}_i = \mathbf{R}_1(\mathbf{x}_i^{pbest} - \mathbf{x}_i) + \mathbf{R}_2(\sigma_x + \mathbf{R}_2)(\mathbf{x}^g - \mathbf{x}_i) \quad (6)$$

It contains two random standard PSO components, along with a divergence factor for the social component. It is expected to behave similarly to **VFPS1**, but with the difference that its behavior should be more like the one of standard PSO when the swarm has a high density (i.e. small  $\sigma_x$  values).

**VFPS5** was evolved on the “Sphere-VF1” (Cross) problem

$$\mathbf{a}_i = \langle \mathbf{x}_i, 0.5\sigma_x \rangle ((\mathbf{x}_i^{pbest} - \mathbf{x}_i) + \mathbf{x}^c + \sigma_x \mathbf{x}^g) \quad (7)$$

Contrarily to the other reported equations, it is completely deterministic and contains both the center and the spread of the swarm.

## 6 Analysis and Discussion of the Evolved VFPS

Median values over 100 runs and corresponding standard errors obtained by these five VFPS equations during the test phase, and the ones obtained by standard PSO, are reported in Table 2. The Kolmogorov-Smirnov non-parametric test has been performed to analyze the statistical significance under the alternative hypothesis that VFPS and PSO results are drawn from the same distribution, with significance level  $p = 0.05$ . The results of the success rate for each

VFPS are shown in Table 3. As can be seen from Table 2, most of the obtained median fitness values for the reported evolved equations (denoted in each row of Table 2 as -PS) under the influence of vector fields (each column from VF1-VF5 of the corresponding objective function) are significantly smaller than those ones obtained by standard PSO equation (first row of Table 2 for every column from VF1-VF5). Standard PSO is extremely bad under considered vector fields influence with medium fitnesses  $\gg 1$  along with almost everywhere a 0% success rate in Table 3. While the median fitnesses of the evolved VFPS are mostly  $< 1$  with more than a 50% success rate on Sphere function and a non-zero success rate on other objective functions under any of the considered VF influence. The only exception is the results obtained on VF4, where PSO is the best performer for Ackley problem, and second best in Sphere and Rosenbrock, with a 100% success rate under VF4 conditions for almost all problems. This particular observation is due to the vector field characteristic, i.e. the values of VF4 according to its description in Table 1 are within  $[-1, 1]$ , which perform rather small disturbances in comparison to magnitude of other VFs. Tables 2 and 3 reveal that the evolved VFPS equations can obtain reasonably good approximation of global optimal solution in contrast to standard PSO velocity rule in total unawareness of the external disturbance.

**Table 2.** Median and standard error (in brackets) over 100 runs of fitness values found by each VFPS (denoted as -PS) in five vector fields (VF1-VF5) on Sphere, Rosenbrock and Ackley. VF0 indicates the case without vector field. Best results in bold. An asterisk indicates statistical significance ( $p < 0.05$ ) between a result obtained by VFPS and PSO according to non-parametric Kolmogorov-Smirnov test.

	Sphere					Rosenbrock					Ackley							
	VF0	VF1	VF2	VF3	VF4	VF5	VF0	VF1	VF2	VF3	VF4	VF5	VF0	VF1	VF2	VF3	VF4	VF5
PSO	<b>.000</b> (.000)	13.64 (17.28)	6.49 (7.706)	5.45 (5.003)	.004 (.004)	4.59 (3.654)	.009 (.004)	38.302 (44.25)	1695.85 (2313.22)	9.072 (10.59)	.110 (.156)	346.05 (319.79)	<b>.003</b> (.003)	7.265 (.602)	3.553 (2.322)	4.554 (1.446)	<b>1.29</b> (.052)	3.864 (.88)
-PS1	<b>.000*</b> (.000)	.192* (.159)	.484* (.293)	.089* (.100)	<b>.002</b> (.001)	.353* (.193)	<b>.003</b> (.005)	4.918* (4.31)	2.405* (4.22)	2.266* (1.384)	<b>.082</b> (.097)	.744* (.508)	.031* (.028)	2.515* (.318)	1.616 (1.643)	2.467* (1.071)	.262 (.164)	2.139* (.407)
-PS2	.092* (.068)	.122* (.081)	<b>.036*</b> (.024)	.135* (.129)	.119 (.128)	.126* (.085)	.848* (.843)	.633* (.482)	<b>.590*</b> (.866)	1.29* (.694)	1.74* (2.93)	.961* (1.103)	2.44* (.571)	2.33* (.99)	1.79 (1.08)	2.92* (.415)	2.19* (.890)	2.28* (.282)
-PS3	.012* (.013)	.120* (.091)	.050* (.030)	<b>.038*</b> (.016)	.015 (.013)	<b>.028*</b> (.032)	.052* (.040)	<b>.201*</b> (.283)	4.109* (5.00)	.987* (.203)	.188* (.142)	<b>.477*</b> (.249)	.214* (.105)	1.332* (1.227)	2.459 (1.41)	3.075* (.558)	.499* (.086)	<b>.591*</b> (.529)
-PS4	.096* (.015)	.053* (.016)	.082* (.073)	.091* (.052)	.034* (.020)	.038* (.009)	5.22* (5.96)	.972* (.338)	.624* (.473)	<b>.541*</b> (.66)	2.87* (2.049)	1.53* (2.27)	2.086* (1.39)	1.312* (1.719)	1.450 (1.215)	2.744* (.175)	2.212* (.261)	1.710* (1.503)
-PS5	.052* (.042)	<b>.049*</b> (.029)	.133* (.099)	.088* (.062)	.078 (.079)	.079* (.037)	1.588* (1.705)	.900* (1.116)	.684* (.479)	.729* (.23)	.542* (.798)	.924* (.764)	1.110* (1.483)	<b>1.157*</b> (.865)	2.251 (.409)	1.521* (1.207)	2.456* (.325)	1.774* (.618)

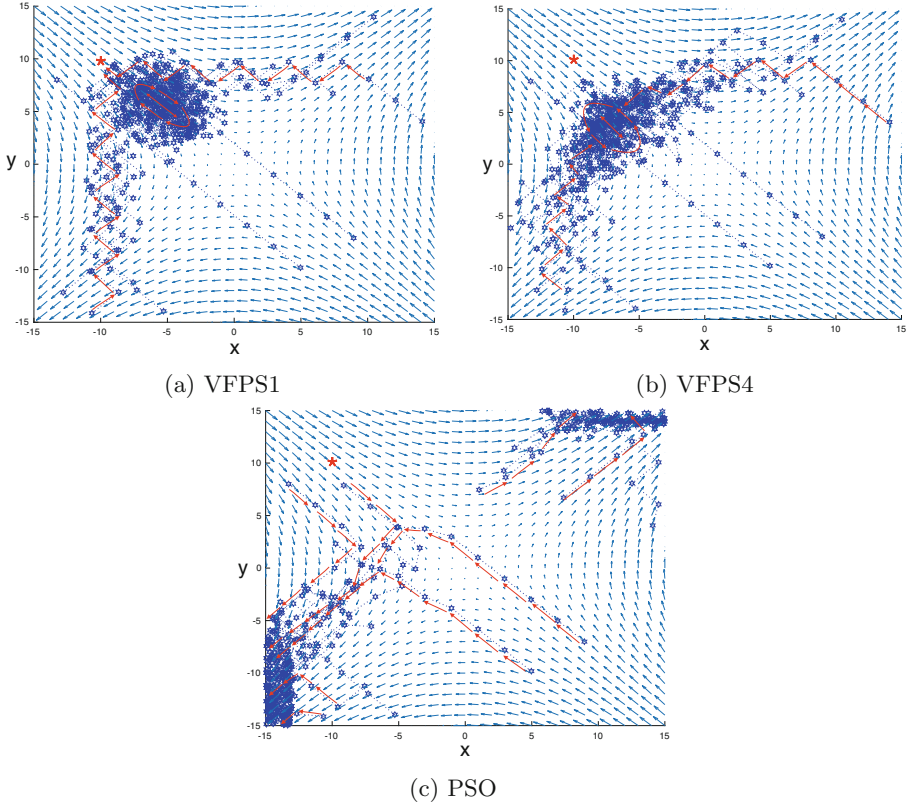
The particles behavior of all the VFPS presented above tend to be resistant to the hard disturbances on unimodal objective functions. Such behavior is obtained since almost all reported VFPS were evolved on the Sphere problem class under VF “Cross” (see Sect. 5), which is characterized by high intensity vectors redirecting away from the goal. Considering that, according to the findings of previous studies [2], the other considered VFs are not so challenging as VF “Cross”, the evolved VFPS are expected to almost always be successful on the Sphere landscape, regardless of the VF type under which they are considered. The results of Tables 2 and 3 confirm this expectation. Moreover,

**Table 3.** Success rate (in %) for 5 evolved VFPS (denoted as -PS) over 100 runs. Best results on each of the VF within considered objective function are in bold.

	Sphere						Rosenbrock						Ackley					
	VF0	VF1	VF2	VF3	VF4	VF5	VF0	VF1	VF2	VF3	VF4	VF5	VF0	VF1	VF2	VF3	VF4	VF5
PSO	<b>100</b>	3	1	3	<b>100</b>	4	99	0	0	0	<b>100</b>	0	<b>100</b>	0	0	0	53	0
-PS1	<b>100</b>	53	53	73	<b>100</b>	52	99	10	7	11	<b>100</b>	3	<b>100</b>	1	1	1	<b>61</b>	0
-PS2	86	77	<b>88</b>	<b>93</b>	82	89	13	24	15	20	24	17	<b>3</b>	0	3	1	2	0
-PS3	<b>100</b>	<b>97</b>	69	82	<b>100</b>	<b>100</b>	98	<b>47</b>	10	17	95	<b>34</b>	99	<b>5</b>	0	0	26	<b>17</b>
-PS4	<b>100</b>	5	9	41	<b>100</b>	11	<b>100</b>	1	0	3	<b>100</b>	4	<b>100</b>	0	0	1	55	1
-PS5	92	94	<b>88</b>	90	94	91	19	13	<b>30</b>	<b>25</b>	18	17	5	0	0	1	2	3

while analyzing the particles trajectories of the evolved VFPS, one can consider that the particles which are initially placed at the lower left corner of the search space, can now reproduce very “straightforward” movements towards the goal in contrast to standard PSO (as an example see Fig. 2) on all problems, despite their landscape structure. That might seem to be beneficial on Ackley, preventing the particles from getting trapped in multiple local optima and moving faster towards the region with the globally optimal solution. However, due to the strong resistance of the evolved VFPS to any appeared disturbance near the goal, they cannot reach the exact global solution (as supposed on Sphere), being misled by the inherited resistant behavior to the local optima surrounding the global one. This is reflected by the high values in Table 3 and the low ones in Table 2. The only exception might be **VFPS3** (evolved on “Sphere-Sheared”), which performs quite well in case of VF5 in comparison to the other VFPS. And, according to its results for the Sphere and Rosenbrock problems, **VFPS3** seems to be a good all-rounder. Characterized by small intensity vectors in the region around the optimal solution, evolution under “Sheared” VF made the structure of **VFPS3** able to seek the particles towards the goal. On Rosenbrock-VFs, in certain cases VFPS are able to obtain a reasonable approximation of the global optimal solution. As mostly VFPS are based on the swarm diversity, getting into the valley region of Rosenbrock, particles start producing more local search behavior. However, constant redirections by VF prevent them from convergence to their local best solutions, so that they are more likely to reach the global optimum.

It is also worth pointing out that, when we observe the trajectories (e.g. in Fig. 2), the VFPS particles, being able to move against the flow, produce zigzag movements (i.e. in Fig. 2a and b), i.e. they behave similarly to a sailboat, which cannot travel directly into the wind but uses a zigzag pattern to move against it, while usual PSO particles are just blown away by the flow (Fig. 2c). This is an interesting finding, considering that such behavior is also typical for birds such as the albatross, which perform a zigzag upwind search in response to odor cues towards the food source [23]. Such behavior is mostly obtained by **VFPS1** (Fig. 2a). As soon as the particles are gathered together, they start to reproduce oscillating behavior near the best found by them point, moving together back and forth.



**Fig. 2.** Trajectories of the particles (in blue) starting from the same initial positions on “Ackley-Cross” obtained by VFPS1, VFPS4 and PSO in (a-c). A five-pointed red snowflake at  $(-10, 10)$  indicates the unknown target (global optimum). Red arrows show the most characteristic general direction of the particles movement at certain regions. (Color figure online)

## 7 Conclusions and Future Work

Analysis of the evolved programs has demonstrated that with small modifications in the velocity rule, PSO can achieve solid collective search behavior in total unawareness of external dynamics, mimicking trajectories of the zigzag upwind birds flights towards the food source. These findings deepen our understanding on the swarm dynamics in the presence of external influence (performed in this study by vector fields) and may shed light on the underlying mechanism of information exchange in natural swarms under dynamic unknown stimuli (e.g. wind), which might find its further applications for outdoor swarm robotics systems. In the future, we plan to increase the functional and terminal sets of the GP system and to test it on other (i.e. non-PSO) metaheuristic algorithms. Also,

we intend to implement our system using a parallel and distributed framework, in order to improve the speed of the overall calculations.


## References

1. Bartashevich, P., Bakurov, I., Mostaghim, S., Vanneschi, L.: Evolving PSO algorithm design in vector fields using geometric semantic GP. In: Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO 2018), Kyoto, July 2018, 2 p. (To appear)
2. Bartashevich, P., Grimaldi, L., Mostaghim, S.: PSO-based search mechanism in dynamic environments: swarms in vector fields. In: 2017 IEEE Congress on Evolutionary Computation, pp. 1263–1270 (2017)
3. Clerc, M.: Stagnation analysis in particle swarm optimization or what happens when nothing happens. Technical report (2006)
4. Di Chio, C., Di Chio, P.: Group-foraging with particle swarms and genetic programming. In: Ebner, M., O’Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) EuroGP 2007. LNCS, vol. 4445, pp. 331–340. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-71605-1\\_31](https://doi.org/10.1007/978-3-540-71605-1_31)
5. Di Chio, C., Poli, R., Langdon, W.B.: Evolution of force-generating equations for PSO using GP. In: Proceedings of the 2005 AI\*IA Workshop on Evolutionary Computation (2005)
6. Diosan, L., Oltean, M.: Evolving the structure of the particle swarm optimization algorithms. In: Gottlieb, J., Raidl, G.R. (eds.) EvoCOP 2006. LNCS, vol. 3906, pp. 25–36. Springer, Heidelberg (2006). [https://doi.org/10.1007/11730095\\_3](https://doi.org/10.1007/11730095_3)
7. Diosan, L., Oltean, M.: What else is the evolution of PSO telling us? *J. Artif. Evol. Appl.* **1**, 1–12 (2008)
8. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. *IEEE Comput. Intell. Mag.* **1**, 28–39 (2006)
9. Erskine, A., Herrmann, J.M.: Critical Dynamics in Particle Swarm Optimization. *CoRR* (2014)
10. Kennedy, J., Eberhart, R.C.: *Swarm Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco (2001)
11. Langdon, W.B., Poli, R.: Evolving problems to learn about particle swarm optimisers and other search algorithms. *IEEE Trans. Evol. Comput.* **11**(5), 561–578 (2007)
12. Lyle, N.L., Howard, W.: The velocity dependence of aerodynamic drag: a primer for mathematicians. *Math. Assoc. Am.* **106**, 127–135 (1999)
13. Moraglio, A., Krawiec, K.: Semantic genetic programming. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 603–627. ACM (2015)
14. Pawlak, T.P., Wieloch, B., Krawiec, K.: Review and comparative analysis of geometric semantic crossovers. *Genet. Program. Evolvable Mach.* **16**, 351–386 (2015)
15. Poli, R., Di Chio, C., Langdon, W.B.: Exploring extended particle swarms: a genetic programming approach. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, New York, USA, pp. 169–176 (2005)
16. Poli, R., Langdon, W.B., Holland, O.: Extending particle swarm optimisation via genetic programming. In: Keijzer, M., Tettamanzi, A., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 291–300. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-31989-4\\_26](https://doi.org/10.1007/978-3-540-31989-4_26)

17. Runka, A.: Evolving an edge selection formula for ant colony optimization. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1075–1082. ACM (2009)
18. Tavares, J., Pereira, F.B.: Evolving strategies for updating pheromone trails: a case study with the TSP. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6239, pp. 523–532. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15871-1\\_53](https://doi.org/10.1007/978-3-642-15871-1_53)
19. Vanneschi, L.: An introduction to geometric semantic genetic programming. In: Schütze, O., Trujillo, L., Legrand, P., Maldonado, Y. (eds.) NEO 2015. SCI, vol. 663, pp. 3–42. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-44003-3\\_1](https://doi.org/10.1007/978-3-319-44003-3_1)
20. Vanneschi, L., Bakurov, I., Castelli, M.: An initialization technique for geometric semantic GP based on demes evolution and despeciation. In: 2017 IEEE Congress on Evolutionary Computation, pp. 113–120 (2017)
21. Vanneschi, L., Silva, S., Castelli, M., Manzoni, L.: Geometric semantic genetic programming for real life applications. In: Riolo, R., Moore, J.H., Kotanchek, M. (eds.) Genetic Programming Theory and Practice XI. GEC, pp. 191–209. Springer, New York (2014). [https://doi.org/10.1007/978-1-4939-0375-7\\_11](https://doi.org/10.1007/978-1-4939-0375-7_11)
22. Wilke, D.N., Kok, S., Groenwold, A.A.: Comparison of linear and classical velocity update rules in particle swarm optimization: notes on scale and frame invariance. *Int. J. Numer. Methods Eng.* **70**(8), 985–1008 (2007)
23. Wyatt, T.: Pheromones and Animal Behavior: Chemical Signals and Signatures. Cambridge University Press, Cambridge (2014)



# Towards an Adaptive CMA-ES Configurator

Sander van Rijn<sup>1</sup>, Carola Doerr<sup>2</sup>, and Thomas Bäck<sup>1</sup>

<sup>1</sup> LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands  
`{s.j.van.rijn,t.h.w.baeck}@liacs.leidenuniv.nl`

<sup>2</sup> Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, Paris, France  
`Carola.Doerr@mpi-inf.mpg.de`

**Abstract.** Recent work has shown that significant performance gains over state-of-the-art CMA-ES variants can be obtained by a recombination of their algorithmic modules. It seems plausible that further improvements can be realized by an adaptive selection of these configurations. We address this question by quantifying the potential performance gain of such an online algorithm selection approach. In particular, we study the advantage of structurally adaptive CMA-ES variants on the functions F1, F10, F15, and F20 of the BBOB test suite. Our research reveals that significant speedups might be possible for these functions. Quite notably, significant performance gains might already be possible by adapting the configuration only once. More precisely, we show that for the tested problems such a single configuration switch can result in performance gains of up to 22%. With such a significant indication for improvement potential, we hope that our results trigger an intensified discussion of online structural algorithm configuration for CMA-ES variants.

**Keywords:** Continuous black-box optimization · CMA-ES  
Online algorithm configuration

## 1 Introduction

Black-box optimization algorithms are an important field of research due to their direct applicability to many problems and their long success history. Many different algorithms are continuously being developed, each with its own performance characteristics to make it better suited to certain problems or problem classes.

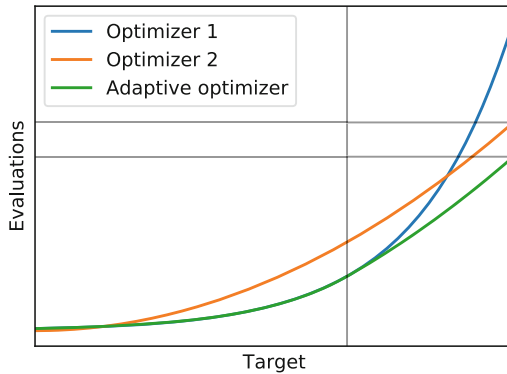
Such optimizers typically use online adaptation of parameters such as step size or population size. A common one is the CMA-ES by Hansen *et al.* [5] and its (B)IPOP derivatives [1, 7]. This adaptation behavior allows a single algorithm to behave differently at different points in time during the optimization process, usually trying to transition from *exploration* to *exploitation*.

When using specific optimizers that are tailored to the specific features of the optimization problem at hand, the efficiency is typically improved. Such

algorithms provide improved convergence rate and/or quality of the final solution found for these problems, but often perform worse than average on other problems. This limits their general applicability and makes it less likely that non-specialists will be aware that the tailored algorithm exists as an option for their particular problem. Furthermore, this also means that many algorithmic variants are not commonly combined with each other, as they are only compared with standard algorithms upon their introduction.

In an effort to explore more of this underlying algorithm combination space, a modular CMA-ES implementation for eleven such variants has been proposed in [12]. This framework easily allows for 4 608 different combinations to be tested and compared, as has been done in [12, 13].

In this paper we consider the potential of *adaptive structural configurations*. That is, we regard an online selection of the most suitable of the 4 608 configurations. In a nutshell, adaptive structural configurations allow to use at every stage of the optimization process the configuration that is most suitable for it. This way, we can in particular switch from a configuration that performs well during the earlier stages of optimization to one that is more suitable in the later parts. Imagine for example the optimization process of the multi-modal Rastrigin function. The search has to avoid many local optima at first, but effectively only has to solve the Sphere problem once the area of the global optimum has been identified.



**Fig. 1.** *Sketch of adaptive configuration.* Sketched performance profiles of two optimizers and an adaptive performance profile based on the profiles of optimizers 1 and 2. The vertical line indicates the transition from using optimizer 1’s behavior to that of optimizer 2. Horizontal lines indicate the required budget to reach the final target for optimizer 2 and the adaptive optimizer. The gained improvement is the difference between these two lines.

Specifically we analyze the potential speed-up that can be gained by simulating such configuration switches using the convergence history of 4 608 CMA-ES-based algorithms. Figure 1 illustrates how we can simulate the potential improvement of using adaptive configurations based on existing performance data. In



Sect. 2 we give a short introduction of the modular CMA-ES framework that is used to create the 4608 CMA-ES configurations. Section 3 defines the procedure applied to calculate the potential speed-up, with Sect. 4 discussing the results. Finally, Sect. 5 concludes this paper and lists some potential avenues for future research.

## 2 Modular CMA-ES

The *modular CMA-ES framework* [12] has been developed to facilitate the testing and exploration of arbitrary combinations of algorithmic variations of the CMA-ES. In the following, such algorithmic variations are called *configurations*. Each configuration is built-up of eleven different *modules* that can be enabled or disabled independently of each other. As two of the possible modules have three options rather than two, this allows for  $2^9 \cdot 3^2 = 4608$  different possible configurations. A list of the modules used in this framework is provided in Table 1. For any further implementation details, see [12].

**Table 1.** *Overview of the available ES modules in the modular CMA-ES framework.* For most of these modules the only available options are *off* and *on*, encoded by the values 0 and 1. For quasi-Gaussian sampling and increasing population, the additional option is encoded by the value 2. The entries in row 9, recombination weights, specify the formula for calculating each weight  $w_i$ .

Module name	0 (default)	1	2
1 Active Update [10]	off	on	-
2 Elitism	$(\mu, \lambda)$	$(\mu + \lambda)$	-
3 Mirrored Sampling [4]	off	on	-
4 Orthogonal Sampling [14]	off	on	-
5 Sequential Selection [4]	off	on	-
6 Threshold Convergence [11]	off	on	-
7 TPA [6]	off	on	-
8 Pairwise Selection [2]	off	on	-
9 Recombination Weights	$\frac{\log(\mu + \frac{1}{2}) - \log(i)}{\sum_j w_j}$	$\frac{1}{\mu}$	-
10 Quasi-Gaussian Sampling [3]	off	Sobol	Halton
11 Increasing Population [1,7]	off	IPOP	BIPOP

## 3 Data Processing

### 3.1 Generation and Pre-processing of the Data

We quantify the theoretical potential of adaptive configuration selection for the four functions F1, F10, F15, and F20 from the BBOB benchmark suite [9], and

for each of these functions we focus on dimensions 5 and 20. Each function is chosen to represent one of the four subgroups from the BBOB suite. For generating and preprocessing the performance data, the following steps are executed:

**Step 1: Generation of runtime data for each of the 4 608 configurations.**

For each ( $F$  = function,  $d$  = dimension) pair we collect running time data from 5 independent runs on 5 different instances, resulting in a total of 25 runs. The BBOB test suite stores the function value of a best-so-far search point after every improvement. The allocated budget is  $10^4 \cdot d$  per run. Since we collect data for each of the 4 608 configurations, this gives us about 4 GB of running time data for each ( $F, d$ ) pair.

**Step 2: Computation of average hitting times AHT.** Following the logic of BBOB, we compute an average hitting time for each of the  $\Gamma = 51$  target precisions  $\{10^{2.0}, 10^{1.8}, \dots, 10^{-7.8}, 10^{-8.0}\}$ , defined as  $\phi_i = 10^{2-(i-1) \cdot 0.2}$ ,  $i \in \{1, 2, \dots, \Gamma\}$ . For every function  $F$ , dimension  $d$ , configuration  $C$ , and target precision  $\phi$  this is done as follows: Let  $0 \leq s \leq 25$  be the number of *successful* runs, i.e., the number of runs of configuration  $C$  that have reached the target precision  $\phi$  on the ( $F, d$ ) pair. The first point in time in which the function has been at most  $\phi$  is referred to as the first hitting time. When  $s = 25$ , i.e., when all runs have been successful, the average hitting time  $\text{AHT}(F, d, C, \phi)$  is simply the average of the hitting times. As we will usually fix the function  $F$  and dimensionality  $d$ , we write  $\text{AHT}(C, \phi)$  as a shorthand for  $\text{AHT}(F, d, C, \phi)$ . When  $s < 25$ , at least one of the runs was not able to reach  $\phi$  and the AHT is set to  $\infty$ . The reasons for this are explained in Sect. 3.3.

**Step 3: Computation of *segmented* average hitting time sAHT.** We next compute from the  $\text{AHT}(C, \phi)$  values an indicator for the performance of the different configurations in the function value *segments*  $S_i = (\phi_{i-1}, \phi_i]$ . For each  $i \in \{1, \dots, \Gamma\}$  let  $\phi_i$  be the  $i$ -th target precision, with  $\phi_0 = \infty$  such that Segment  $S_1 = (\infty, 10^2)$ . This *segmented* average hitting time of configuration  $C$  in segment  $S_i$  for function, dimension pair ( $F, d$ ) is defined as

$$\text{sAHT}(C, i) = \text{AHT}(C, \phi_i) - \text{AHT}(C, \phi_{i-1}) \quad (1)$$

i.e., difference in average hitting times. We can assume that  $\text{sAHT}(C, i) \geq 0$  since the average hitting time is monotonically increasing by construction:

$$\forall i : \text{AHT}(C, \phi_i) \leq \text{AHT}(C, \phi_{i+1}).$$

### 3.2 Constructing Optimal Adaptive Configurations

The full convergence behavior of a configuration  $C$  towards final target  $\phi_\Gamma$  can be defined based on the sAHT measure defined above:

$$\text{AHT}(C, \phi_\Gamma) = \sum_{i=1}^{\Gamma} \text{sAHT}(C, i). \quad (2)$$

**Maximally Adaptive.** We can adapt Eq. (2) to replace any static configuration  $C$  by a sequence of configurations  $\mathbf{C} = \{C_1, C_2, \dots, C_\Gamma\}$ :

$$\text{AHT}(\mathbf{C}, \phi_\Gamma) = \sum_{i=1}^{\Gamma} \text{sAHT}(C_i, i). \quad (3)$$

Under the assumption that the performance of  $C_i$  on section  $S_i$  is independent of  $C_i$ 's behavior up to target  $\phi_{i-1}$ , we can then pick this sequence  $\mathbf{C}$  such that

$$\mathbf{C}_{\text{opt}} = \{C_i \mid i \in \{1, \dots, \Gamma\}, C_i = \arg \min_C (\text{sAHT}(C, i))\} \quad (4)$$

then we can guarantee by construction that

$$\text{AHT}(\mathbf{C}_{\text{opt}}, \phi_\Gamma) \leq \text{AHT}(\mathbf{C}, \phi_\Gamma) \quad (5)$$

for any sequence  $\mathbf{C}$  of configurations.

In other words, this means that we consider  $\mathbf{C}_{\text{opt}}$  to be an algorithm that chooses its internal configuration for each segment  $S_i$  to be precisely the configuration  $C_i$  that has the smallest sAHT for that segment.

**Single Split.** Naturally, this maximally adaptive algorithm is practically quite infeasible for various reasons. However, by restricting our choice of  $\mathbf{C}$  we can create a more feasible alternative that still outperforms any single original configuration  $C$ .

Let  $\mathbf{C}$  be a sequence  $(C_1, C_2, \dots, C_\Gamma)$  such that:  $C_i = C_1 \forall i \in \{1, \dots, s\}$  and  $C_i = C_\Gamma \forall i \in \{s+1, \dots, \Gamma\}$ , where  $1 < s < \Gamma$  is the *split* index. This represents an adaptive approach where a single configuration switch occurs during the runtime, namely when reaching  $\phi_s$ . For a given split  $s$ , we can then find  $C_1$  and  $C_\Gamma$  that minimize the AHT for their respective sequence of segments.

By repeating this for all possible splits, an optimal split  $s$  with corresponding configurations  $C_1$  and  $C_\Gamma$  can be computed from the dataset. By appropriating the argmin notation, we can write this as follows:

$$(C_1, C_\Gamma, s) = \arg \min_{C_1, C_\Gamma, s} \left( \sum_{i=1}^s \text{sAHT}(C_1, \phi_i) + \sum_{i=s+1}^{\Gamma} \text{sAHT}(C_\Gamma, \phi_i) \right) \quad (6)$$

or in terms of AHT:

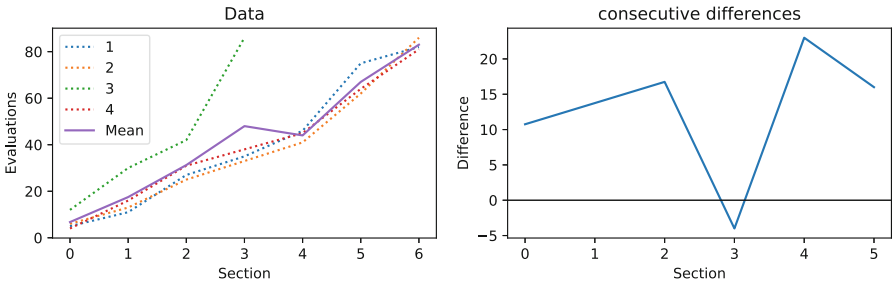
$$(C_1, C_\Gamma, s) = \arg \min_{C_1, C_\Gamma, s} (\text{AHT}(C_1, \phi_s) + (\text{AHT}(C_\Gamma, \phi_\Gamma) - \text{AHT}(C_\Gamma, \phi_s))) \quad (7)$$

In words: we consider an adaptive algorithm that is split at the end of segment  $S_i$ , such that configuration  $C_1$  performs best for  $S_1, \dots, S_i$  and  $C_\Gamma$  performs best for  $S_{i+1}, \dots, S_\Gamma$  and their combined AHT is minimal again. In the worst case scenario,  $C_1 = C_\Gamma$  and we do not get any improvement.

### 3.3 Discarding Partially Successful Configurations

As mentioned in Sect. 3.1, we deliberately choose to ignore any configuration that was not successful in *all* 25 runs for the desired target value  $\phi$ . This absolves us of the associated uncertainty at the cost of reducing the effective size of our dataset. Otherwise, we would run into problems with the previously defined (s)AHT measure using the two most likely options of dealing with unsuccessful runs: ignoring and penalizing.

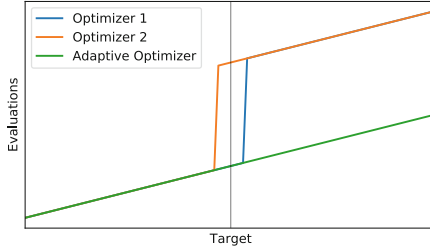
**Ignoring Unsuccessful Runs.** If we were to ignore missing values caused by unsuccessful runs and simply average over all available individual hitting times, we can easily end up with negative sAHT values, as shown in Fig. 2.



**Fig. 2.** Example of non-monotonicity causing negative sAHT values. The original, monotonically increasing data, consists of four runs, one of which is not successful for targets 4 and up. If an average is taken of all valid data points, the unsuccessful run increases the average for targets 0–3, but the average drops once the unsuccessful run is no longer taken into account. The difference, or sAHT, between targets 3 and 4 becomes negative in this case.

**ERT/Simulated AHT.** Methods such as *simulated AHT* or *Estimated Running Time (ERT)* as defined in [8] have been introduced to incorporate these unsuccessful runs into the AHT. These methods artificially increase the hitting time by substituting the unsuccessful runs with (multiples of) the maximum evaluation budget to account for the uncertainty of success. Although this is very useful when considering a configuration’s performance from start to finish, the penalty can be ignored when we only consider the performance between given targets.

Figure 3 shows how this can happen. In this figure, two otherwise equal optimizers 1 and 2 are penalized for an unsuccessful run, each after reaching a different intermediate target. When applying the construction of Eq. (6), the optimal point to switch will be at the vertical line, i.e. *before* optimizer 1 is penalized, but *after* optimizer 2 is. This way, the penalty is completely avoided by the adaptive configuration sequence. The resulting AHT then indicates that a large performance improvement is possible, while this not the case.



**Fig. 3.** *Adaptive optimizer omits penalties for unsuccessful runs.* This figure shows a sketch of two optimizers that both incur a penalty for having unsuccessful runs, shown by a jump in the number of evaluations. The vertical line indicates a single configuration switch. As the penalties occur at either side of the switch, the adaptive optimizer avoids the penalty.

## 4 Results

### 4.1 Maximally Adaptive

Results for the maximally adaptive strategy are listed in Table 2, where the relative improvement is calculated as  $1 - \text{Adaptive}/\text{Static}$ . *Static* refers to a non-adaptive configuration  $\mathbf{C} = (C_1, C_2, \dots, C_T)$  where  $C_1 = C_2 = \dots = C_T$ . They show improvements ranging from 11 up to 47%. It must be noted, however, that, given the budget of  $10^4 \cdot d$ , 5d F20, 20d F15 and 20d F20 have not reached the final target of  $10^{-8}$ . The listed results are only up to the listed target  $\phi$ .

Although the results are impressive, they seem unreliable. After all, CMA-ES performs rather well on the sphere function as-is, so an expected speed-up of 47% for 5d F1 is unrealistic. The explanation for this lies in the fact that CMA-ES are still stochastic processes. If we have a number of configurations that perform similarly over the entire runtime, some local variance is to be expected. As the size of each segment  $S_i$  decreases, we end up cherry-picking sAHT values that are small by chance rather than because of inherent information they contain.

Having said this, these results still provide an upper bound for the improvement that may be obtained by making better use of the available structural configuration space.

### 4.2 Single Split

Table 3 lists the results when only considering a single configuration switch. Potential improvement ranges from 3 up to 22% in this case. The reached targets are the same as for the maximally adaptive method. Convergence behavior of these results is shown in Figs. 4 and 5.

For the easiest F1 functions, these improvements are much smaller than in the maximally adaptive setting: 6.9% versus 47.1% and 3% versus 16.6% for 5d and 20d F1, respectively. This much lower improvement combined with the convergence behavior as shown in Fig. 4 supports the explanation that these

**Table 2.** Results of only successful configurations. The *Static* and *Adaptive* columns indicate the AHT values. Column  $\phi$  lists the smallest target values for which the shown AHT values were obtained. Relative improvement  $r$  is calculated as  $1 - \text{Adaptive}/\text{Static}$ .

$d$	$F$	$\phi$	Static	Adaptive	$r$	Static $C$
5	1	$10^{-8.0}$	412.00	218.05	0.471	00110011010
5	10	$10^{-8.0}$	1437.08	832.00	0.421	11000110022
5	15	$10^{-8.0}$	14812.72	9220.80	0.378	00110011011
5	20	$10^{-0.2}$	14535.16	10951.92	0.247	01010101022
20	1	$10^{-8.0}$	1269.05	1058.45	0.166	00110110021
20	10	$10^{-8.0}$	16565.48	11135.00	0.328	00001000010
20	15	$10^{0.6}$	45279.08	26249.16	0.420	00111000001
20	20	$10^{0.2}$	14476.76	12829.40	0.114	00010001022

improvements for F1 are most likely due to random variance. When comparing the representations for 20d F1 for example, we can see that the best normal configuration is the same as that for the second part of the optimization process, and that the configuration for the first part only differs in the final module: IPOP instead of BIPOP, which should not make difference when optimizing the sphere function F1.

For more difficult functions such as 5d F10 and F15 however, the single split method already shows a large potential improvement of over a third to a half of the upper bound established by the maximally adaptive method. The improvement is clearly visible in the convergence behavior. The convergence of the first part follows that of a rather successful configuration, which does not perform best overall, while for the second part, the behavior of a different configuration is followed. This second configuration may take longer to exhibit this beneficial search behavior, but because of the switch, we can disregard this initial delay and use its good performance in the latter half of the optimization process to (significantly) beat the best static configuration.

On the other hand, the results for 5d F20, 20d F15 and 20d F20 are not very informative as most target values are not reached. For these functions, even higher budgets are required for CMA-ES-based optimizers to reach the target value of  $10^{-8}$ .

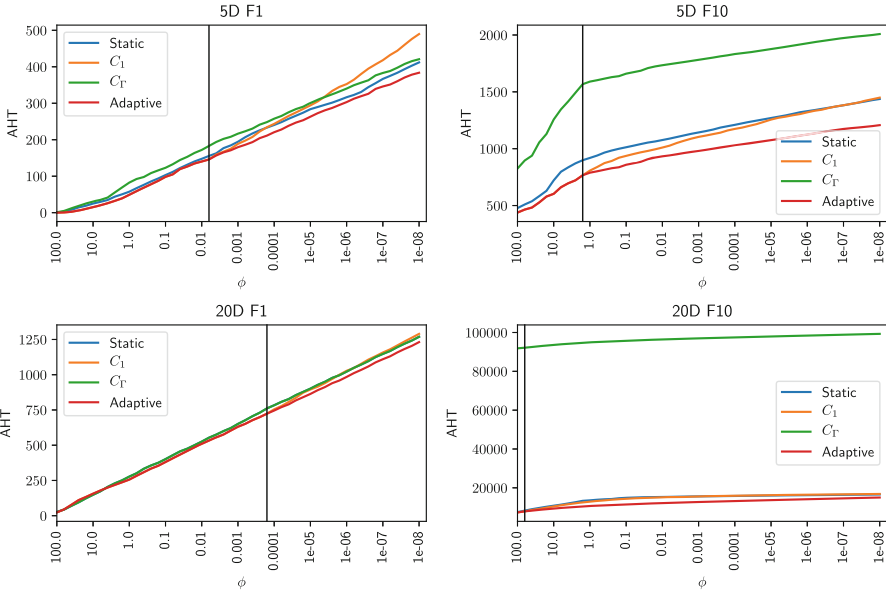
### 4.3 Discussion

As already noted in Sect. 4.1, because the data has been created by a stochastic process, measurement uncertainty has to be taken into account. Although we have tried to reduce its influence by using an average over 25 runs and discarding any AHT values for which not all 25 were successful, some variance remains. This is very visible in the analysis of F1 (sphere function).

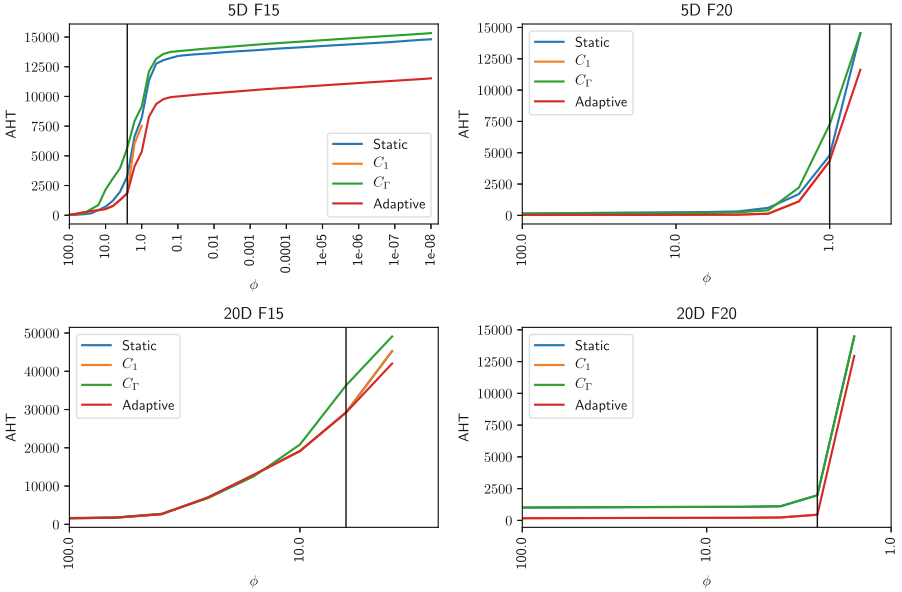
The results of 5d F10 (see Fig. 4) shows a very clear knee-point at the split, after which all three original configurations start performing better, i.e. needing

**Table 3.** Results of only successful configurations. The *Static* and *Adaptive* columns indicate the AHT values. Relative improvement  $r$  is calculated as  $1 - \text{Adaptive}/\text{Static}$ . *Split* indicates the target value  $\phi_i$  after which we switch from one configuration to the other. The final three columns and  $C_T$  show the representation for the static best configuration and the configurations  $C_1, C_T$  that make up the first and second part of the adaptive approach, respectively. This representation can be decoded using Table 1.

$d$	$F$	Static	Adaptive	$r$	Split	Static $C$	$C_1$	$C_T$
5	1	412.00	383.70	0.069	$10^{-2.2}$	00 110 011 010	10 110 111 020	00 110 011 011
5	10	1437.08	1207.12	0.160	$10^{0.2}$	11 000 110 022	01 001 110 120	01 101 000 012
5	15	14812.72	11524.24	0.222	$10^{0.4}$	00 110 011 011	00 110 000 001	00 001 011 011
5	20	14535.16	11628.36	0.200	$10^{0.0}$	01 010 101 022	01 110 011 011	00 100 001 021
20	1	1269.05	1231.40	0.030	$10^{-3.8}$	00 110 110 021	00 110 110 022	00 110 110 021
20	10	16565.48	15003.72	0.094	$10^{1.8}$	00 001 000 010	00 011 000 011	00 001 101 011
20	15	45279.08	42020.80	0.072	$10^{0.8}$	00 111 000 001	00 111 000 001	00 101 000 011
20	20	14476.76	12942.96	0.106	$10^{0.4}$	00 010 001 022	11 100 111 001	00 010 001 022



**Fig. 4.** Convergence behavior of configurations listed in Table 3. These plots show  $AHT(F, d, C, \phi)$ , the average number of evaluations required for a configuration  $C$  to reach a target value  $\phi$ . The vertical line indicates the location of the optimal *split*. The convergence labeled *Adaptive* consist of the behavior of configuration  $C_1$  before the split, and the behavior of configuration  $C_T$  afterwards. If none of the configurations were successful in all 25 runs, there is no data for those convergence targets  $\phi$ .



**Fig. 5.** Convergence behavior of configurations listed in Table 3. These plots show  $AHT(F, d, C, \phi)$ , the average number of evaluations required for a configuration  $C$  to reach a target value  $\phi$ . The vertical line indicates the location of the optimal *split*. The convergence labeled *Adaptive* consist of the behavior of configuration  $C_1$  before the split, and the behavior of configuration  $C_R$  afterwards. If none of the configurations were successful in all 25 runs, there is no data for those convergence targets  $\phi$ .

fewer evaluations to reach the next targets. This change in performance is not the same however, which is exploited by the adaptive configuration, confirming the intuition motivating this research.

A practical caveat, however, is that a fast but less successful configuration such as ‘part 1’ in 5d F15 as seen in Fig. 5 may reach its best target value  $\phi$  by exploiting a local optimum that is not necessarily close to the global optimum. In such cases, continuing the search with a different configuration will *not* result in the speed-up demonstrated in this paper, but rather in a slow-down as the algorithm would have to escape the local optimum, if possible at all, before finding the global optimum on its own.

## 5 Conclusion and Future Work

In this paper we have shown that it is possible to empirically determine upper bounds for the possible speed-up when considering structurally adaptive CMA-ES-based optimization algorithms. The results support the idea that improvements of 5–20% are already viable when switching algorithm configuration only once during the optimization process. We hope this research will inspire further investigation into online structural adaptation of optimization algorithms.



However, the assumption that is required for the presented interpretation (see before Eq. (4)) is non-trivial. As the internal state of CMA-ES-based optimizers is highly dependent on the search history, there is no guarantee that the performance will correctly continue after an actual configuration switch. An important next step is then to run some of the identified adaptive configurations to evaluate their actual performance. This should be done on white-box versions of the used benchmark functions so the switch can be performed at the identified optimal intermediate target. In this paper, we intentionally focused on the data-driven analysis, and investigating the above-mentioned setting will be the next step towards understanding and exploiting adaptive configurations.

Additionally, the process described in this paper can also be further improved with statistical analysis of the (s)AHT values that are used to determine hypothetical performance of the adaptive configurations. E.g. for the sphere function F1, such analysis will give an indication of whether the improvement is due to random variance in the data or due to actual differences in convergence behavior.

**Acknowledgements.** The authors would like to thank Hao Wang for his participation in the discussions leading up to this work.

## References

1. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1769–1776, September 2005. <https://doi.org/10.1109/CEC.2005.1554902>
2. Auger, A., Brockhoff, D., Hansen, N.: Mirrored sampling in evolution strategies with weighted recombination. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011, pp. 861–868. ACM, New York (2011). <https://doi.org/10.1145/2001576.2001694>
3. Auger, A., Jebalia, M., Teytaud, O.: Algorithms (X, sigma, eta): quasi-random mutations for evolution strategies. In: Talbi, E.-G., Liardet, P., Collet, P., Lut-ton, E., Schoenauer, M. (eds.) EA 2005. LNCS, vol. 3871, pp. 296–307. Springer, Heidelberg (2006). [https://doi.org/10.1007/11740698\\_26](https://doi.org/10.1007/11740698_26)
4. 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 2010. LNCS, vol. 6238, pp. 11–21. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_2](https://doi.org/10.1007/978-3-642-15844-5_2)
5. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 312–317, May 1996. <https://doi.org/10.1109/ICEC.1996.542381>
6. Hansen, N.: CMA-ES with Two-Point Step-Size Adaptation. [arXiv:0805.0231](https://arxiv.org/abs/0805.0231) [cs], May 2008
7. Hansen, N.: Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, GECCO 2009, pp. 2389–2396. ACM, New York (2009). <https://doi.org/10.1145/1570256.1570333>, <http://doi.acm.org/10.1145/1570256.1570333>

8. Hansen, N., Auger, A., Brockhoff, D., Tuar, D., Tuar, T.: COCO: Performance Assessment. [arXiv:1605.03560](https://arxiv.org/abs/1605.03560) [cs], May 2016
9. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2009: experimental setup. Report, INRIA (2009). <https://hal.inria.fr/inria-00362649/document>
10. Jastrebski, G.A., Arnold, D.V.: Improving evolution strategies through active covariance matrix adaptation. In: 2006 IEEE International Conference on Evolutionary Computation, pp. 2814–2821 (2006). <https://doi.org/10.1109/CEC.2006.1688662>
11. Piad-Morffis, A., Estvez-Velarde, S., Boluf-Rhler, A., Montgomery, J., Chen, S.: Evolution strategies with threshold convergence. In: 2015 IEEE Congress on Evolutionary Computation (CEC), pp. 2097–2104, May 2015. <https://doi.org/10.1109/CEC.2015.7257143>
12. van Rijn, S., Wang, H., van Leeuwen, M., Bäck, T.: Evolving the structure of evolution strategies. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8, December 2016. <https://doi.org/10.1109/SSCI.2016.7850138>
13. van Rijn, S., Wang, H., van Stein, B., Bäck, T.: Algorithm configuration data mining for CMA evolution strategies. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 737–744. ACM, New York (2017). <https://doi.org/10.1145/3071178.3071205>, <http://doi.acm.org/10.1145/3071178.3071205>
14. Wang, H., Emmerich, M., Bäck, T.: Mirrored orthogonal sampling with pairwise selection in evolution strategies. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC 2014, pp. 154–156. ACM, New York (2014). <https://doi.org/10.1145/2554850.2555089>, <http://doi.acm.org/10.1145/2554850.2555089>

# **Combinatorial Optimization**



# A Probabilistic Tree-Based Representation for Non-convex Minimum Cost Flow Problems

Behrooz Ghasemishabankareh<sup>1(✉)</sup>, Melih Ozlen<sup>1</sup>, Frank Neumann<sup>2</sup>,  
and Xiaodong Li<sup>1</sup>

<sup>1</sup> School of Science, RMIT University, Melbourne, Australia

{behrooz.ghasemishabankareh,melih.ozlen,xiaodong.li}@rmit.edu.au

<sup>2</sup> School of Computer Science, The University of Adelaide, Adelaide, Australia  
frank.neumann@adelaide.edu.au

**Abstract.** Network flow optimisation has many real-world applications. The minimum cost flow problem (MCFP) is one of the most common network flow problems. Mathematical programming methods often assume the linearity and convexity of the underlying cost function, which is not realistic in many real-world situations. Solving large-sized MCFPs with nonlinear non-convex cost functions poses a much harder problem. In this paper, we propose a new representation scheme for solving non-convex MCFPs using genetic algorithms (GAs). The most common representation scheme for solving the MCFP in the literature using a GA is priority-based encoding, but it has some serious limitations including restricting the search space to a small part of the feasible set. We introduce a probabilistic tree-based representation scheme (PTbR) that is far superior compared to the priority-based encoding. Our extensive experimental investigations show the advantage of our encoding compared to previous methods for a variety of cost functions.

**Keywords:** Representation scheme · Genetic algorithm  
Minimum cost flow problem · Mixed integer nonlinear programming

## 1 Introduction

Network flow problems have numerous applications in electrical and power networks, telecommunication, road and rail networks, and airline services [2]. Different types of network flow problems exist, e.g., the shortest path problem, the maximum flow problem, the assignment problem, the transportation problem, and the minimum cost flow problem (MCFP), among which MCFP is one of the most general cases with applications such as distribution problems, optimal loading of a Hopping aeroplane and the racial balancing of schools [2].

MCFPs can be formulated and solved by Linear Programming (LP) techniques, when the underlying cost function is linear or can be approximated by a linear function [17]. However, many real-world MCFPs are nonlinear and require

formulation using a nonlinear cost function, instead of a linear approximation. For example, in a transportation problem, the nonlinearity of a cost function is due to the economy of scale phenomenon, which occurs when cost per unit of the transportation flow decreases with an increasing amount of the total flow [7]. Many studies suggest the appropriateness of employing nonlinear cost functions in the network design problems [4, 15].

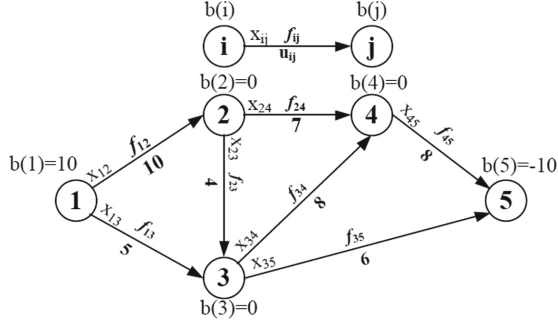
Some attempts have been made in using genetic algorithms (GAs) to solve the network flow problems [1, 7, 13]. Among these works, the *representation scheme* plays a critical role in their success. Several representation schemes exist for the network flow problems such as variable-length encoding [19], fixed-length encoding [3], and priority-based representation (PbR) [16]. The most common representation scheme for solving MCFPs is PbR [8]. PbR scheme has been used to solve the shortest path problems, the transportation problems, as well as the network design problems [8, 13, 16]. Although PbR is widely used for solving network flow problems, it has some serious drawbacks (when dealing with MCFP), most noticeably its restriction on any search algorithm from reaching some parts of the feasible search space (see Sect. 2 for details).

To counteract the above limitations, in this paper we propose a probabilistic tree-based representation (PTbR) for solving nonlinear non-convex MCFP instances using the GA. The PTbR allows all possible feasible solutions to be generated, instead of being restricted to a small part of the feasible region (e.g., PbR scheme). This paper first examines the capabilities of PTbR and compare it with that of the PbR scheme. Then a comparative study is carried out on the performance of the GA employing these two different representation schemes on a set of 35 benchmark instances. This paper has the following contributions: (1) proposing a novel representation scheme (PTbR) to deal with MCFP; (2) providing a close examination between PTbR and PbR to find out which one is more effective for handling MCFPs; (3) conducting extensive experiments to compare the performance of the PTbR-based GA (PtGA) variants with the PbR-based GA (PrGA) for solving non-convex MCFP instances. We also compare our results with those of the mathematical solver packages.

The rest of the paper is structured as follows: Sect. 2 gives the preliminaries and Sect. 3 describes our proposed probabilistic tree-based representation and the GA employing PTbR scheme for solving MCFPs. The experimental studies are presented in Sects. 4 and 5 provides the conclusion.

## 2 Preliminaries

This section describes the problem definition, the PbR, and finally discusses the drawbacks of PbR. Let  $G(N, A)$  be a network consisting of a set  $N$  of  $n$  nodes and a set  $A$  of  $m$  directed arcs. The maximum and minimum amount of flow on each arc  $(i, j)$  are equal to  $u_{ij}$  and 0, respectively.  $b(i)$  denotes the amount of supply or demand for source or sink node.  $b(i) > 0$  denotes that node  $i$  is a supply node and  $b(i) < 0$  shows that node  $i$  is a demand node with a demand of  $-b(i)$  and  $b(i) = 0$  denotes the transshipment node  $i$ . Figure 1 shows an example of the



**Fig. 1.** An example of the MCFP ( $n = 5$ ,  $m = 7$ ).

MCFP with  $n = 5$  nodes and  $m = 7$  arcs, which has one supplier node ( $b(1) = 10$ ) and one demand node ( $b(5) = -10$ ). In this example, we aim to satisfy the demand by sending all supplies through the network while minimising the total cost. The integer flow on an arc  $(i, j)$  is represented by  $x_{ij}$  and the associated cost for the flow ( $x_{ij}$ ) is denoted by  $f_{ij}(x_{ij})$ . The formulation of the MCFP is as follows [2]:

$$\text{Minimise : } z(\mathbf{x}) = \sum_{(i,j) \in A} f_{ij}(x_{ij}), \quad (1)$$

$$\text{s.t. } \sum_{\{j:(i,j) \in A\}} x_{ij} - \sum_{\{j:(j,i) \in A\}} x_{ji} = b(i) \quad \forall i \in N, \quad (2)$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A, \quad (3)$$

$$x_{ij} \in \mathbb{Z} \quad \forall (i, j) \in A, \quad (4)$$

where Eq. 1 minimises the total cost through the network. Equation 2 is a flow balance constraint which states the difference between the total outflow (first term) and the total inflow (second term). The flow on each arc should be between an upper bound and zero (Eq. 3), and finally all the flow values are integer numbers (Eq. 4). In this paper we consider the following assumptions for the MCFP: (1) the network is directed; (2) there are no two or more arcs with the same tail and head in the network; (3) the single-source single-sink MCFP is considered; (4) the total demands and supplies in the network are equal, i.e.,  $\sum_{i=1}^n b(i) = 0$ .

## 2.1 Priority-Based Representation

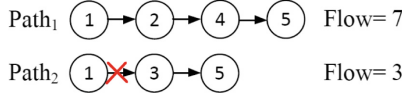
Priority-based representation (PbR) is the most commonly-used representation method for MCFPs [8]. In order to represent a candidate solution for an MCFP, PbR lets the number of genes to be equal to  $n$  and the value of each gene is generated randomly between 1 and  $n$ , which represents the priority of each node for constructing a path among all possible nodes [8]. Figure 2a illustrates the PbR chromosome for the network presented in Fig. 1. In order to obtain a feasible solution, a two-phase decoding procedure is followed. In phase I, a path is



(a) An example of a chromosome for the network in Fig. 1.

(b) A feasible solution for the given chromosome.

**Fig. 2.** The PbR chromosome and its corresponding solution.



**Fig. 3.** A feasible solution that PbR fails to represent (for the network in Fig. 1).

generated based on the priorities and the maximum possible flow is sent through the generated path in phase II. After sending the flow on the network, the upper bound ( $u_{ij}$ ), supply and demand should be updated. If the supply/demand is not equal to 0, the next path should be generated. The above procedure repeats until all demands are satisfied. Figure 2b presents a feasible solution for the given chromosome in Fig. 2a.

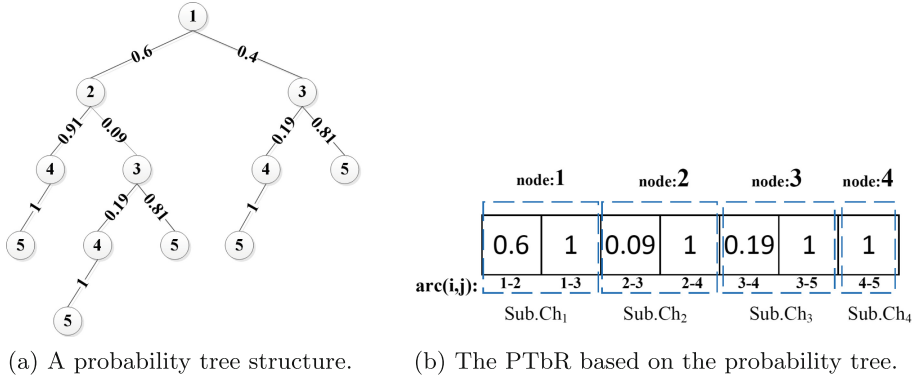
Although PbR has been commonly used in the network flow problems, it has some limitations in representing the full extent of the feasible space for MCFP. Figure 3 shows an example (for the network presented in Fig. 1) that PbR is unable to represent. Here the first path is generated as follows:  $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$ . Since in  $Path_1$  after node 1, node 2 is selected, it shows that node 2 has a higher priority than node 3. Hence, if arc (1, 2) is not saturated, PbR will not allow any flow to be sent through arc (1, 3), essentially blocking this possibility completely (Fig. 3,  $Path_2$ ). This means that PbR is unable to represent a potential feasible solution such that the flow would go through arc (1, 3) (as shown in Fig. 3). Another limitation for PbR is that each time a path is generated, we are supposed to send the maximum possible amount on the generated path. These limitations would restrict a search algorithm from reaching the full extent of the feasible space.

### 3 Proposed Method

Representation plays a critical role before applying an optimisation algorithm, and this applies to GA too. In this section we first propose a probabilistic tree-based representation (PTbR) scheme for solving MCFPs, which alleviates the deficiency of using PbR. Then we describe the GA employing PTbR for solving MCFP instances.

#### 3.1 Probabilistic Tree-Based Representation

To counteract the above-mentioned limitations of the PbR, we propose the PTbR scheme, where a probability tree is adopted to represent a potential MCFP



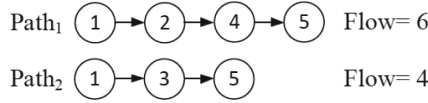
**Fig. 4.** Probability tree and its corresponding PTbR for the network in Fig. 1.

solution. Unlike the PbR scheme which is restricted to a small part of the feasible space, the PTbR is able to represent all possible feasible solutions. Figure 4a shows an example of the probability tree for the network presented in Fig. 1. Here, the probability of each successor node to be selected is defined on each branch.

The tree structure can be converted to a chromosome with several sub-chromosomes. Figure 4b shows the PTbR chromosome converted from the probability tree presented in Fig. 4a. The PTbR chromosome has  $n - 1$  sub-chromosomes (Sub.Ch) and the value of each gene is a random number between 0 and 1 which is then accumulated to 1 in each sub-chromosome. In order to obtain a feasible solution from PTbR, in phase I, a path is first constructed, and then a feasible flow is sent through the constructed path in phase II. For example, to obtain a feasible solution for the chromosome in Fig. 4b, we generate the first path from node  $i = 1$  (Sub.Ch <sub>$i=1$</sub> ). A random number is generated in  $[0,1]$  ( $rand = 0.2$ ), and since  $0 \leq rand = 0.2 \leq 0.6$ , we move through arc (1,2) and node 2 is selected. From node 2 (Sub.Ch <sub>$i=2$</sub> ) another random number is generated ( $0.09 \leq rand = 0.85 \leq 1$ ) and the selected successor node is 4. From node 4 the only available node is 5. Hence, the following path is generated:  $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$ .

In Phase II, we attempt to send a feasible flow through the generated path. First the capacity of the generated path is defined ( $U = \min\{u_{12} = 10, u_{24} = 7, u_{45} = 8\} = 7$ ). Then, there are three possible approaches to send a feasible flow on the generated path: (1) send a random flow between 1 and  $U$  (random(**R**)); (2) send a flow 1-by-1 (one-by-one (**O**)); (3) send the maximum possible amount of the flow on the generated path (maximum(**M**)), which is the same as PbR. In the above example, we follow the first approach (random(**R**)) and after calculating  $U = 7$ , we send a random flow in  $[1, 7]$  (e.g., flow = 6) and the network, supply and demand are updated. Since the demand has not been fully met (i.e., not equal to 0 yet), the above procedure is repeated.





**Fig. 5.** A feasible solution generated based on the PTbR chromosome in Fig. 4b.

Figure 5 shows a feasible solution for the chromosome presented in Fig. 4b. Note that in Fig. 5, after generating Path<sub>1</sub>, although arc (1,2) is not saturated, the second path picks node 3 as the successor of node 1, unlike the PbR. This example illustrates that PTbR allows all potential solutions to be generated probabilistically, instead of being restricted by using PbR.

### 3.2 Genetic Algorithm with PTbR

This section describes the GA employing the new representation scheme PTbR for solving MCFPs, i.e., PtGA. The key distinction between the PtGA and the PbR-based GA (PrGA) is that PrGA employs the PbR [8]. This PtGA can be described by the following procedure:

**Initialisation:** First a population with *pop\_size* individuals (chromosomes) is randomly generated. The process of creating a chromosome based on the PTbR is explained in Subsect. 3.1.

**Crossover and Mutation:** In order to explore the feasible region, crossover and mutation operators are applied to create the new offspring at each generation. For PtGA, a two-point crossover operation is applied, where two blocks (sub-chromosomes) of the selected chromosome (parents) are first randomly selected. Then, two parents swapping the selected sub-chromosomes to generate new offspring. To perform mutation for PtGA, first a random parent is selected and the randomly chosen sub-chromosome is regenerated to create a new offspring.

**Fitness Evaluation and Selection:** For each chromosome in the population, after finding a feasible solution ( $\mathbf{x}$ ) by applying the decoding procedure for PTbR, the value of cost function is evaluated using the following equation:  $Minimize : z(\mathbf{x}) = \sum_{i=1}^n \sum_{j=1}^n f(x_{ij})$ . After calculating the fitness values for all individuals in the population, the tournament selection procedure is applied to select individuals for the next generation.

**Termination Criteria:** The termination criteria for the PtGA are as follows: (1) no further fitness value improvement in the best individual of the population for  $\beta$  successive iterations; (2) the maximum number of function evaluations (NFEs) reached. If any of the above conditions is satisfied first, the algorithm stops and the best solution ( $\mathbf{x}^*$ ) and its corresponding cost function value are reported.

Note that for PrGA, it is common to employ a weight mapping crossover (WMX) and inversion mutation [8]. The termination criteria can be the same for both PrGA and PtGA.

## 4 Experimental Studies

This section first describes the MCFP instances and cost functions that have been adopted, followed by some discussion about the mathematical solver packages used in our experiments. We then describe the parameter settings, experimental comparisons and result analysis on the performances of PrGA, PtGA, and mathematical solvers in solving these MCFP instances.

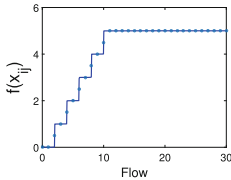
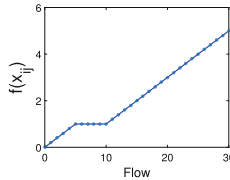
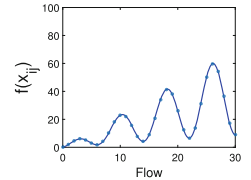
Since our focus is to solve nonlinear non-convex MCFP, we adopt a set of nonlinear non-convex cost functions which are commonly-used in the literature [9, 10, 14]. Michalewicz et al. [14] categorised the nonlinear cost functions as (1) piece-wise linear cost functions; (2) multimodal (nonlinear non-convex) cost functions; (3) smooth cost functions which are mostly used for Operations Research (OR) problems. In this paper we chose the nonlinear non-convex and arc-tangent approximation of the piece-wise linear cost functions from [9, 10, 14] to evaluate the performances of PrGA and PtGA. The formulation of these functions are as follows [9, 10, 14]:

$$F_1 : f(x_{ij}) = c_{ij} (\arctan(P_A(x_{ij} - S))/\pi + 0.5 + \arctan(P_A(x_{ij} - 2S))/\pi + 0.5 + \arctan(P_A(x_{ij} - 3S))/\pi + 0.5 + \arctan(P_A(x_{ij} - 4S))/\pi + 0.5 + \arctan(P_A(x_{ij} - 5S))/\pi + 0.5). \quad (5)$$

$$F_2 : f(x_{ij}) = c_{ij} ((x_{ij}/S)(\arctan(P_B x_{ij})/\pi + 0.5) + (1 - x_{ij}/S)(\arctan(P_B(x_{ij} - S))/\pi + 0.5) + (x_{ij}/S - 2)(\arctan(P_B(x_{ij} - 2S))/\pi + 0.5)). \quad (6)$$

$$F_3 : f(x_{ij}) = 100 \times c_{ij} (x_{ij} (\sin(\frac{5\pi x_{ij}}{4S}) + 1.3)). \quad (7)$$

Note that  $c_{ij}$  is non-negative coefficient,  $P_A$  and  $P_B$  are set to 1000 and  $S$  is set to 2 for  $F_1$ , and 5 for  $F_2$  and  $F_3$ , respectively [10]. All cost functions  $F_1$ ,  $F_2$  and  $F_3$  are illustrated in Fig. 6. A set of 35 single-source single-sink MCFP instances is randomly generated with different number of nodes ( $n = \{5, 10, 20, 40, 80, 120, 160\}$ ) and presented in Table 1 ( $No.$  denotes the instance number, and each instance has  $n$  nodes and  $m$  arcs). Note that, for each node size ( $n$ ), five different networks are randomly generated. The number of supply/demand for nodes  $1/n$  are set to  $q = 20/-20$  in the test instances up to 20 nodes and for all other test problems supply/demand are set to  $q = 30/-30$ .

(a) Cost function  $F_1$ (b) Cost function  $F_2$ (c) Cost function  $F_3$ 

**Fig. 6.** Shapes of different cost functions.

**Table 1.** A set of 35 randomly generated single-source single-sink MCFP instances.

<i>No.</i>	<i>n</i>	<i>m</i>	<i>No.</i>	<i>n</i>	<i>m</i>	<i>No.</i>	<i>n</i>	<i>m</i>	<i>No.</i>	<i>n</i>	<i>m</i>	<i>No.</i>	<i>n</i>	<i>m</i>	<i>No.</i>	<i>n</i>	<i>m</i>			
1	8	6	24	11	114	16	369	21	1484	26	3419	31	4882							
2	8	7	34	12	98	17	385	22	1406	27	3166	32	4718							
3	<b>5</b>	8	<b>8</b>	<b>10</b>	32	13	<b>20</b>	105	18	<b>40</b>	373	23	<b>80</b>	1560	28	<b>120</b>	3326	33	<b>160</b>	4986
4		9		9	27	14		99	19		406	24		1353	29		3212	34		4835
5		8		10	29	15		101	20		406	25		1526	30		2911	35		5130

This paper focuses on solving nonlinear non-convex MCFPs, which could be considered as mixed integer nonlinear programming (MINLP) problems. However, only very few mathematical solver packages exist for solving MINLP problems, such as CPLEX, Couennn, Baron, LINDOGlobal and AlphaECP [5, 12, 18]. Some of these solvers have serious limitations. For instance, CPLEX is only capable of solving quadratic optimisation problems, BARON cannot handle the trigonometric functions  $\sin(x)$ ,  $\cos(x)$ , while Couenne is not able to handle the *arctangent* function [5]. Among these solvers, AlphaECP and LINDOGlobal are able to handle general MINLPs [12, 18]. As a result, we choose to compare our PtGA and PrGA results with those of LINDOGlobal and AlphaECP.

### 4.1 Parameter Settings

Both PrGA and PtGA are implemented in MATLAB on a PC with Intel(R) Core(TM) i7-6500U 2.50 GHz processor with 8 GB RAM and run 30 times for each problem instance. In order to solve MCFP instances using mathematical solvers, AlphaECP is applied through a high level mathematical language general algebraic modelling system (GAMS) [11] and LINDOGlobal [12] is applied directly on all problem instances.

The parameter settings for the PrGA are as follows: maximum number of iterations ( $It_{max} = 200$ ), population size ( $pop\_size = \min\{n \times 10, 300\}$ ), crossover rate ( $P_c = 0.95$ ), mutation rate ( $P_m = 0.3$ ) and maximum number of function evaluations ( $NFEs = 100,000$ ). The parameter settings for the PtGA are  $It_{max} = 200$ ,  $pop\_size = \min\{n \times 5, 300\}$ ,  $P_c = 0.95$ ,  $P_m = 0.3$  and  $NFEs = 100,000$ . The  $pop\_size$  value depends on the number of nodes ( $n$ ) and increases for the larger networks and the  $P_m = 0.3$  value decreases linearly in each iteration. If the results are not improved in  $\beta = 30$  successive iterations for PrGA or PtGA, the algorithm is terminated. The run time limit for LINDOGlobal and AlphaECP is set to 3600 seconds (s). Other parameters for AlphaECP and LINDOGlobal are set as default settings.

### 4.2 Results and Analysis

As mentioned in the procedure of PTbR, after finding a path, there are three possible ways to send the flow over the generated path, i.e., send possible flow (1) randomly (**R**), (2) one-by-one (**O**), or (3) by a maximum possible amount (**M**).



**Table 4.** Results for cost function  $F_3$ .

No.	n	m	PtGA-R			PtGA-O			PtGA-M			PrGA		LINDOGlobal	AlphaECP	h		
			t	mean	std	t	mean	std	t	mean	std	t	mean	std	t'		OBJ	t'
1	8	5	<b>114.3819</b>	7.29E-14	11	<b>114.3819</b>	7.29E-14	3	<b>114.3819</b>	7.29E-14	9	<b>114.3819</b>	7.29E-14	1	<b>114.3819</b>	1	<b>114.3819</b>	0
2	8	6	<b>111.8675</b>	0.00E+00	14	<b>111.8675</b>	0.00E+00	4	<b>111.8675</b>	0.00E+00	9	<b>111.8675</b>	0.00E+00	1	<b>111.8675</b>	1	<b>111.8675</b>	0
3	8	7	<b>138.2576</b>	0.00E+00	16	<b>138.2576</b>	0.00E+00	3	138.285	0.00E+00	6	138.285	0.00E+00	1	<b>138.2576</b>	1	142.79	0
4	9	7	<b>107.3083</b>	0.00E+00	14	<b>107.3083</b>	0.00E+00	3	120.0322	5.83E-14	6	123.3197	2.92E-14	1	<b>107.3083</b>	1	123.32	0
5	8	6	<b>150.9943</b>	2.92E-14	15	<b>150.9943</b>	2.92E-14	3	<b>150.9943</b>	2.92E-14	7	<b>150.9943</b>	2.92E-14	1	<b>150.9943</b>	1	<b>150.9943</b>	0
6	24	66	106.7312	1.44E+00	108	107.7509	1.18E+00	43	118.2249	5.35E+01	41	135.3015	1.87E+00	3600	<b>104.5113</b>	332	141.64	-1
7	34	36	81.7436	3.64E-01	69	85.7071	3.81E+00	39	89.5549	7.08E+00	32	117.8976	1.06E+01	3600	<b>80.97226</b>	431	108.13	-1
8	32	49	89.4711	1.47E+00	65	88.5659	6.40E-01	36	90.3497	2.08E+00	33	131.963	6.02E+01	3600	<b>86.41261</b>	491	135.434	-1
9	27	45	112.2958	1.31E+00	74	115.9811	4.28E+00	37	114.295	1.58E+00	27	152.5404	1.28E+00	3600	<b>110.3257</b>	288	133.92	-1
10	29	49	90.1242	8.41E-01	78	90.2275	8.48E-01	39	90.4574	1.21E+00	35	104.8298	8.38E+00	3600	<b>88.05281</b>	333	135.81	-1
11	114	90	69.7972	1.25E+00	163	105.9186	8.93E-01	81	103.9751	3.76E+00	86	103.4472	4.32E+00	3600	<b>66.6795</b>	3600	102.375	0
12	98	111	79.5198	5.91E-01	194	116.2849	4.75E-01	79	116.3386	5.97E+00	78	122.6103	3.04E+00	3600	<b>79.3041</b>	3600	92.146	0
13	105	137	96.3334	2.15E+00	234	123.3836	2.84E+00	137	120.5739	1.11E+00	93	116.6983	3.49E+00	3600	<b>90.1959</b>	3600	137.339	0
14	99	92	<b>79.4689</b>	0.00E+00	213	103.5296	1.40E+00	86	99.9841	3.27E+00	80	115.5684	1.98E+00	3600	<b>79.4689</b>	3600	115.578	0
15	101	145	82.9519	1.56E+00	199	108.6625	1.11E+00	99	111.233	2.93E+00	75	115.2218	2.27E+00	3600	<b>78.4144</b>	3600	116.503	-1
16	369	299	<b>75.7446</b>	0.00E+00	384	139.4979	4.30E+00	223	105.1299	2.87E+00	223	108.7354	3.66E+00	3600	<b>77.1499</b>	3600	145.164	-1
17	385	253	<b>89.9066</b>	3.16E-01	366	108.3421	1.07E+00	241	111.1345	2.21E+00	294	119.3059	3.82E+00	3600	<b>90.1749</b>	3600	148.655	0
18	373	203	83.1843	3.89E+00	582	133.7859	3.33E+00	433	106.883	4.90E+00	215	93.0854	3.19E+00	3600	<b>79.8989</b>	3600	90.777	-1
19	406	282	<b>72.447</b>	5.14E-01	355	119.7419	4.45E+00	231	99.0947	3.44E+00	285	103.488	2.99E+00	3600	<b>73.0978</b>	3600	92.443	-1
20	406	288	65.1725	4.73E-01	359	127.004	3.93E+00	277	102.8105	6.24E+00	201	109.9314	3.55E+00	3600	<b>65.0667</b>	3600	117.312	0
21	1484	325	<b>83.9645</b>	1.85E+00	404	138.502	5.27E+00	296	109.589	1.40E+00	336	107.3701	2.04E+00	3600	<b>90.1854</b>	3600	124.549	-1
22	1406	238	<b>93.3544</b>	1.13E+00	332	128.5883	6.91E+00	246	109.0366	1.42E+00	347	108.2733	2.49E+00	3600	<b>93.4224</b>	3600	129.357	0
23	1560	296	<b>106.1091</b>	3.03E+00	304	143.037	2.91E+00	243	112.9184	1.57E+00	272	111.1676	1.20E+00	3600	<b>148.6104</b>	3600	107.493	0
24	1353	261	<b>63.7182</b>	4.32E-01	464	134.5088	7.03E+00	331	108.1399	2.83E+00	314	94.7708	5.93E+00	3600	<b>65.9662</b>	3600	132.307	-1
25	1526	365	<b>59.5713</b>	2.19E-14	293	134.002	6.85E+00	241	101.3695	7.98E+00	335	106.9928	5.50E+00	3600	<b>59.5713</b>	3600	125.508	0
26	3419	544	<b>86.9299</b>	1.93E+00	595	130.1538	2.76E+00	491	84.2045	2.36E+00	686	89.0996	3.92E+00	3600	<b>NF</b>	3600	91.267	-1
27	3166	464	<b>55.0274</b>	1.79E+00	475	129.6023	2.81E+00	437	83.2053	3.92E+00	544	87.5812	5.25E+00	3600	<b>66.009</b>	3600	92.576	0
28	3326	530	81.7884	8.00E-00	693	131.3889	3.27E+00	535	88.2298	5.87E+01	685	88.4319	5.24E-01	3600	<b>NF</b>	3600	<b>80.755</b>	0
29	3212	486	<b>84.6939</b>	5.19E+00	566	133.1519	2.59E+00	528	90.2997	5.19E-01	535	89.1701	4.72E-01	3600	<b>NF</b>	3600	136.381	-1
30	2911	481	<b>89.7215</b>	5.19E+00	525	136.358	4.51E+00	455	90.4635	1.55E+00	582	95.9065	6.85E+00	3600	<b>NF</b>	3600	102.4492	0
31	4882	885	<b>80.6756</b>	5.27E+00	994	257.169	5.66E+00	631	145.3273	6.83E+00	907	158.2899	1.14E+01	3600	<b>145.654</b>	3600	98.330	-1
32	4718	835	<b>82.8948</b>	9.12E-00	944	270.3601	8.35E+00	651	156.9115	8.70E+00	895	174.8795	7.43E+00	3600	<b>144.216</b>	3600	123.575	-1
33	4986	910	<b>80.4273</b>	5.40E+00	1066	266.059	6.02E+00	694	159.3669	5.94E+00	900	170.5026	1.20E+01	3600	<b>NF</b>	3600	117.75	-1
34	4835	992	<b>82.432</b>	3.75E+00	946	247.8246	4.70E+00	740	143.5167	7.49E+00	949	158.1368	9.00E+00	3600	<b>NF</b>	3600	102.685	-1
35	5130	988	<b>88.063</b>	4.01E+00	1040	272.3861	5.92E+00	856	162.7781	7.43E+00	917	181.6462	9.77E+00	3600	<b>173.679</b>	3600	117.999	-1

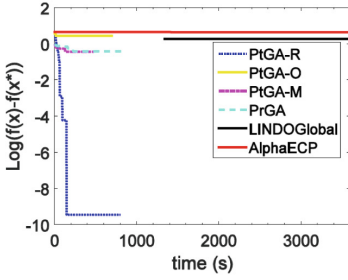
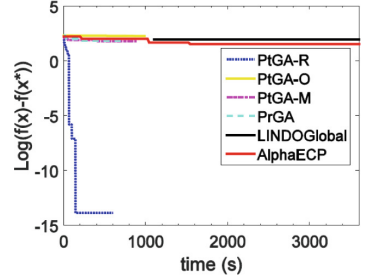
**Table 5.** The Friedman test’s results for PtGA-R, PtGA-O, PtGA-M and PrGA.

	p-value	Mean column ranks			
		PtGA-R	PtGA-O	PtGA-M	PrGA
$F_1$	0.000E+00	16.49	34.13	50.94	60.43
$F_2$	0.000E+00	18.99	59.27	32.69	51.05
$F_3$	0.000E+00	16.28	54.94	41.05	49.72

and AlphaECP (exact methods) denote the running time and the cost function value, respectively. ‘‘NF’’ denotes that the mathematical solver cannot find any feasible solution in the time limit of an hour (3600s). The best cost function value for each instance is presented in boldface.

To carry out a comprehensive comparison among PtGA-R, PtGA-O, PtGA-M, and PrGA, we use Friedman test [6]. For each function ( $F_1$ ,  $F_2$  and  $F_3$ ) we perform the Friedman test with the significance level set to 0.05, and the results are shown in Table 5. Since the  $p$ -values in all three functions are almost zero (less than 0.05), there are overall statistically significant differences between the mean ranks of the algorithms (PtGA-R, PtGA-O, PtGA-M and PrGA). The mean column rank values of the PtGA-R is less than those of the PtGA-O, PtGA-M and PrGA (Table 5) which indicates that PtGA-R’s performance is better than those of the other GA variants. It is clearly evident that the superior performance of the PrGA-R comes from utilising PTbR in its procedure and sending a random possible flow.

We also compare the performance of PtGA-R with LINDOGlobal and AlphaECP by applying a one-sample  $t$ -test with the significance level set to 0.05. After performing the one-sample  $t$ -test, if PtGA-R has statistically better or worse performance than that of the mathematical solvers, the parameter  $h$  is

(a)  $F_2$  on instance No.30.(b)  $F_3$  on instance No.35.**Fig. 7.** Convergence graphs for PtGA, PrGA, LINDOGlobal and AlphaECP.

set to 1 and  $-1$  respectively, otherwise  $h$  is set to 0. The last column of Tables 2, 3 and 4 presents the value of  $h$  for all instances.

For cost function  $F_1$ , Table 2 shows that PtGA-R has better performance on all instances with  $n = \{80, 120, 160\}$  compared with that of PtGA-O, PtGA-M, PrGA, LINDOGlobal and AlphaECP. Furthermore, LINDOGlobal fails to find any feasible solutions when the problem size is increased ( $n = \{80, 120, 160\}$ ). For  $F_2$ , Table 3 shows that on 28 out of 35 instances (80%), the PtGA-R has equal or better performance than the two mathematical solvers.

With regard to cost function  $F_3$ , Table 4 shows that even on instances 3 and 4 (small-sized instances), PrGA failed to find the optimal solutions due to the limitations of PbR in searching the feasible region, which is consistent with our analysis in Subsect. 2.1. In all large-sized instances ( $n = \{80, 120, 160\}$ ), the PtGA-R has similar or better performance than that of the mathematical solvers.

Figure 7 shows the convergence graphs of PtGA-R, PtGA-O, PtGA-M, PrGA and the mathematical solvers for large-sized instances on  $F_2$  and  $F_3$ . Since LINDOGlobal is not able to find any feasible solution for all large-sized problems on  $F_1$ , we are not able to provide the convergence graph for that cost function. As shown in Fig. 7, PtGA-R converges to a good solution faster than other GA variants as well as LINDOGlobal and AlphaECP. Based on Fig. 7, LINDOGlobal cannot find any feasible solution after about 1000s. Once a solution is found, mathematical solvers (specially LINDOGlobal) are not able to improve it.

## 5 Conclusion

This paper has proposed a new encoding scheme called probabilistic tree-based representation (PTbR) for more effective handling of MCFPs. We examine the commonly-used priority-based representation (PbR), and compare it with PTbR to demonstrate that PTbR is superior to PbR for solving MCFPs. To validate our analysis on these representation schemes, the PTbR-based GA (i.e., PtGA) and PbR-based GA (i.e., PrGA) are evaluated over a set of 35 single-source single-sink network instances with up to five thousand variables. The experimental

results demonstrate that PtGA with a random flow (i.e., PtGA-R) has better performance than PrGA on all problem instances. In addition, PtGA-R has also been shown to produce better solutions and have better efficiency than mathematical solvers such as LINDOGlobal and AlphaECP when considering the large-sized instances. For future research, one can focus on solving large-sized real-world MCFP using the proposed representation method.

## References

1. Abdelaziz, M.: Distribution network reconfiguration using a genetic algorithm with varying population size. *Electr. Power Syst. Res.* **142**, 9–11 (2017)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: *Network Flows: Theory, Algorithms, and Applications*, pp. 4–6. Prentice Hall, Upper Saddle River (1993)
3. Aiello, G., La Scalia, G., Enea, M.: A multi objective genetic algorithm for the facility layout problem based upon slicing structure encoding. *Expert Syst. Appl.* **39**(12), 10352–10358 (2012)
4. Amiri, A.S., Torabi, S.A., Ghodsi, R.: An iterative approach for a bi-level competitive supply chain network design problem under foresight competition and variable coverage. *Transp. Res. Part E: Logist. Transp. Rev.* **109**, 99–114 (2018)
5. Burer, S., Letchford, A.N.: Non-convex mixed-integer nonlinear programming: a survey. *Surv. Oper. Res. Manag. Sci.* **17**(2), 97–106 (2012)
6. Derrac, J., García, 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 Evol. Comput.* **1**(1), 3–18 (2011)
7. Fontes, D.B., Gonçalves, J.F.: Heuristic solutions for general concave minimum cost network flow problems. *Networks* **50**(1), 67–76 (2007)
8. Gen, M., Cheng, R., Lin, L.: *Network Models and Optimization: Multiobjective Genetic Algorithm Approach*. Springer, London (2008). <https://doi.org/10.1007/978-1-84800-181-7>
9. Klanšek, U.: Solving the nonlinear discrete transportation problem by minlp optimization. *Transport* **29**(1), 1–11 (2014)
10. Klanšek, U., Pšunder, M.: Solving the nonlinear transportation problem by global optimization. *Transport* **25**(3), 314–324 (2010)
11. Lastusilta, T., et al.: GAMS MINLP solver comparisons and some improvements to the AlphaECP algorithm. In: *Process Design and Systems Engineering Laboratory, Department of Chemical Engineering Division for Natural Sciences and Technology, Abo Akademi University, Abo, Finland* (2011)
12. Lin, Y., Schrage, L.: The global solver in the LINDO API. *Optim. Methods Softw.* **24**(4–5), 657–668 (2009)
13. Lotfi, M., Tavakkoli-Moghaddam, R.: A genetic algorithm using priority-based encoding with new operators for fixed charge transportation problems. *Appl. Soft Comput.* **13**(5), 2711–2726 (2013)
14. Michalewicz, Z., Vignaux, G.A., Hobbs, M.: A nonstandard genetic algorithm for the nonlinear transportation problem. *ORSA J. Comput.* **3**(4), 307–316 (1991)
15. Reça, J., Martínez, J., López-Luque, R.: A new efficient bounding strategy applied to the heuristic optimization of the water distribution networks design. In: *Congress on Numerical Methods in Engineering CMN* (2017)
16. Tari, F.G., Hashemi, Z.: A priority based genetic algorithm for nonlinear transportation costs problems. *Comput. Ind. Eng.* **96**, 86–95 (2016)

17. Vegh, L.A.: A strongly polynomial algorithm for a class of minimum-cost flow problems with separable convex objectives. *SIAM J. Comput.* **45**(5), 1729–1761 (2016)
18. Westerlund, T., Pörn, R.: Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optim. Eng.* **3**(3), 253–280 (2002)
19. Zhang, Y.H., Gong, Y.J., Gu, T.L., Li, Y., Zhang, J.: Flexible genetic algorithm: a simple and generic approach to node placement problems. *Appl. Soft Comput.* **52**, 457–470 (2017)





# Comparative Study of Different Memetic Algorithm Configurations for the Cyclic Bandwidth Sum Problem

Eduardo Rodriguez-Tello<sup>1</sup> , Valentina Narvaez-Teran<sup>1</sup> ,  
and Frédéric Lardeux<sup>2</sup> 

<sup>1</sup> CINVESTAV – Tamaulipas, Km. 5.5 Carretera Victoria-Soto La Marina,  
87130 Victoria, Tamaulipas, Mexico

`{ertello, mnarvaez}@tamps.cinvestav.mx`

<sup>2</sup> LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France  
`frederic.lardeux@univ-angers.fr`

**Abstract.** The Cyclic Bandwidth Sum Problem (CBSP) is an NP-Hard Graph Embedding Problem which aims to embed a simple, finite graph (the guest) into a cycle graph of the same order (the host) while minimizing the sum of cyclic distances in the host between guest's adjacent nodes. This paper presents preliminary results of our research on the design of a Memetic Algorithm (MA) able to solve the CBSP. A total of 24 MA versions, induced by all possible combinations of four selection schemes, two operators for recombination and three for mutation, were tested over a set of 25 representative graphs. Results compared with respect to the state-of-the-art top algorithm showed that all the tested MA versions were able to consistently improve its results and give us some insights on the suitability of the tested operators.

**Keywords:** Cyclic Bandwidth Sum Problem · Memetic algorithms  
Graph Embedding Problems

## 1 Introduction

Graph Embedding Problems (GEP) are combinatorial problems which aim to find the most suitable way to embed a guest graph  $G$  into a host graph  $H$  [3, 5]. An embedding is a labeling of the vertices of  $G$  by using the vertices of  $H$ . The Cyclic Bandwidth Sum Problem [2] can be formally defined as follows. Let  $G = (V, E)$  be a finite undirected (guest) graph of order  $n$  and  $C_n$  a cycle (host) graph with vertex set  $|V_H| = n$  and edge set  $E_H$ . Given an injection  $\varphi : V \rightarrow V_H$ , representing an embedding of  $G$  into  $C_n$ , the cyclic bandwidth sum (the cost) for  $G$  with respect to  $\varphi$  is defined as:

$$\text{Cbs}(G, \varphi) = \sum_{(u,v) \in E} |\varphi(u) - \varphi(v)|_n, \quad (1)$$

where  $|x|_n = \min\{|x|, n - |x|\}$  (with  $1 \leq |x| \leq n - 1$ ) is called the *cyclic distance*, and the label associated to vertex  $u$  is denoted  $\varphi(u)$ .

Then, the CBSP consists of finding the optimal embedding  $\varphi^*$ , such that  $\text{Cbs}(G, \varphi^*)$  is minimum, i.e.,  $\varphi^* = \arg \min_{\varphi \in \Phi} \{\text{Cbs}(G, \varphi)\}$  with  $\Phi$  denoting the set of all possible embeddings.

The CBSP is an NP-Hard problem originally studied by Yuang [16]. Most of the work reported in the literature has focused on theoretical research about calculating (or at least approximating) the optimal solution for some well-known graph topologies. Some of the topologies addressed by the reported exact formulas [2] are paths, cycles, wheels,  $k$ -th powers of cycles and complete bipartite graphs. For the Cartesian products of two graphs (when those graphs are paths, cycles or complete graphs) upper bounds have been reported in [9]. The relation of the CBSP with the Bandwidth Sum Problem<sup>1</sup> (BSP) was also studied [2]. Given the relevant applications of this problem on VLSI designs [1, 15], code design [7], simulation of network topologies for parallel computer systems [12], scheduling in broadcasting based networks [11], signal processing over networks [6] and compressed sensing in sensor networks [10], it has recently caught attention in the combinatorial optimization and operation research areas.

Theoretical formulations are useful to estimate optimal values, but they say little about how to algorithmically construct optimal embeddings, or at least near optimal solutions. This resulted in the development of two approximated algorithms devised to solve the CBSP: General Variable Neighborhood Search (GVNS) [14] and a greedy heuristic denominated as MACH [6].

GVNS algorithm applies Reduced Variable Neighborhood Search (RVNS) to improve its initial solution, which consist of a lexicographical embedding. The properly said GVNS phase includes six perturbation operators and two neighborhoods. When dealing with path, cycle, star and wheel topologies of order  $n \leq 200$  GVNS was able to achieve optimal results as well as solutions under the theoretical upper bounds for Cartesian products of order  $n \leq 64$  and graphs of the Harwell-Boeing collection of order  $n \leq 199$ .

MACH is a two phase greedy heuristic algorithm. In the first phase the guest graph  $G$  is partitioned into disjoint paths by a depth first search mechanism guided by the Jaccard index [8] as a similarity criterion between vertices. Since Jaccard index measures the similarity between vertices neighborhoods, vertices with common neighbors are likely to be included near each other in the same path. In the second phase a solution is incrementally built up by merging the paths. The longer path is added to the solution, then a greedy strategy is implemented to determine where in the partial solution the remaining paths should be inserted. It was experimentally shown that MACH consistently improves the solution quality achieved by GVNS, as well as the running time. Therefore, MACH is currently considered as the best-known algorithm to solve the CBSP.

Our approach consists in studying a combination of genetic and local search inspired operators implemented into a Memetic Algorithm to solve the general

---

<sup>1</sup> BSP is the problem of embedding a graph into a path while minimizing the sum of linear distances between embedded vertices.

case of the CBSP. We worked with four selection schemes, two recombination mechanisms, three mutation schemes and one survival strategy. The 24 possible combinations of operators (MA versions) were duly tested.

Our experiments over a set of 25 topologically diverse representative instances allowed us to obtain significantly improved results with respect to the state-of-the-art top algorithm. We also obtained some insights about the effectiveness of some of the tested operators for helping solving the CBSP.

The rest of this work is organized as follows. MA main routine and operator implementations are described in Sect. 2. Our experimental methodology and the results of the comparisons among the 24 implemented MA versions with respect to the literature results are shown and discussed in Sect. 3. Finally, the conclusions of this work and further research directions are presented in Sect. 4.

## 2 Memetic Algorithms for the CBSP

Algorithm 1 describes the main framework common to all our MA versions. Population  $P$  contains  $\mu$  individuals. At each generation we chose from  $P$  couples of individuals for recombination by crossover. Then, the resulting individuals are mutated and extra perturbations of their chromosomes are performed by inversion. Local search is applied only to the best individual  $P_{best}$  in the surviving population  $P$ , in order to accelerate the computational time expended in each generation. Furthermore, as it is described in Sect. 2.4, the mutation operators also incorporate certain local search operations.

Although  $o''$  is the individual added to the offspring population  $O$ , we also compare the fitness corresponding to previous states of its chromosome ( $o$  and  $o'$ ) with the best historically found solution  $g$ , in order to avoid losing any possible improvement, even if  $o$  and  $o'$  are not actually in  $O$ . The historically best found solution record  $g$  is kept independently of the populations  $P$  and  $O$ .

### 2.1 Solution Encoding and Initialization

The potential solutions were turned into chromosomes by the permutation encoding. An individual is represented as  $P_i = (\varphi_i, \rho_i, f_i)$  where  $\varphi_i$  and  $\rho_i$  are two representations of the same embedding:  $\varphi_i(u)$  stands for the label associated to vertex  $u$  (i.e., the vertex in the host graph associated to vertex  $u$ ).  $\rho_i(u')$  denotes the vertex in  $G$  having the label  $u'$  (i.e., the vertex hosted in vertex  $u'$ ); and  $f_i = f(\varphi_i, G)$  is the fitness of the individual assessed by the fitness function which corresponds to (1). Whenever a change occurs in  $\varphi_i$  it is reflected in  $\rho_i$  and vice-versa. All individuals in population  $P$  are initialized by the assignment of random permutations to their chromosomes. With exception of *insertion* mutation, all of our operators work primarily over  $\varphi_i$ .

### 2.2 Selection

We will denote  $S$  as a multiset containing the individuals for mating. Since we use the Cbs values as fitness values and CBSP is a minimization problem, the

---

**Algorithm 1.** Memetic Algorithm
 

---

```

1:  $P \leftarrow \text{initializePopulation}(P, \mu)$ 
2:  $O \leftarrow \emptyset$ 
3:  $t \leftarrow 1$ 
4:  $g \leftarrow P_{best}$ 
5: repeat
6:   for  $i \leftarrow 1$  to  $\mu$  do
7:      $P_a, P_b \leftarrow \text{selection}(P)$ 
8:      $o \leftarrow \text{crossover}(P_a, P_b, prob_c)$ 
9:      $o' \leftarrow \text{mutation}(o, prob_m)$ 
10:     $o'' \leftarrow \text{inversion}(o', prob_i)$ 
11:     $O \leftarrow O \cup o''$ 
12:     $g \leftarrow \text{fitter individual among current } g, o, o' \text{ and } o''$ 
13:  end for
14:   $P \leftarrow \text{survival}(P, O)$ 
15:   $O \leftarrow \emptyset$ 
16:   $P_{best} \leftarrow \text{localsearch}(P_{best}, tries)$ 
17:   $g \leftarrow \text{fitter individual among current } g \text{ and } P_{best}$ 
18: until stop criterion is met
19: return  $g$ 

```

---

individuals with lower Cbs values are actually the fittest ones. Therefore, in the case of *stochastic* and *roulette* selections we performed a min-max normalization of the fitness values. Then, for each individual its expected value was calculated based on its normalized fitness.

*Stochastic* selection is performed by adding to  $S$  as many copies of each individual as the integer part of its expected value indicates. Then, the floating point parts are used to probabilistically determine whether or not to add an additional copy.

In *roulette* selection the expected values serve as an indicator of the size of the section corresponding to each individual in the roulette. We pick  $2\mu$  individuals by spinning the roulette  $2\mu$  times. The higher the expected values, the bigger the section and the higher chances for the individual to be chosen.

*Random* selection is rather simple, it just picks  $2\mu$  individuals from  $P$ , with replacement. *Binary tournament* performs  $2\mu$  tournament rounds. At each round the individual with the lower Cbs value is chosen. So, when implementing *random* or *binary tournament* selections there is neither need for normalization nor for expected values.

### 2.3 Crossover

Two permutation specialized crossover operators were implemented: *cyclic* [13] and *order-based* crossover [4]. An offspring is created as follows. First, a couple of individuals from  $S$  is picked with replacement. Each couple can produce only one offspring. It is probabilistically decided if this individual is created by recombination, with probability  $prob_c$ , or if it is a copy of the fitter individual in the selected couple.

*Cyclic* crossover operates by computing the *cycles* between both parent chromosomes. The individual inherits, alternately, one cycle from one of the parents and one from the other. By doing this, the operator produces a new permutation in which the absolute positions of each of its genes is preserved with respect to one of the parents, and therefore implicit mutations are avoided.

*Order-based* crossover picks a random segment of genes from one parent individual and inherits it directly to the offspring. Then, the rest of genes of the offspring are assigned in the same order as they appeared in the other parent. This operator balances the preserving of absolute positions of the permutation elements and their relative order. It introduces implicit mutations, but within a limited scope.

## 2.4 Mutation

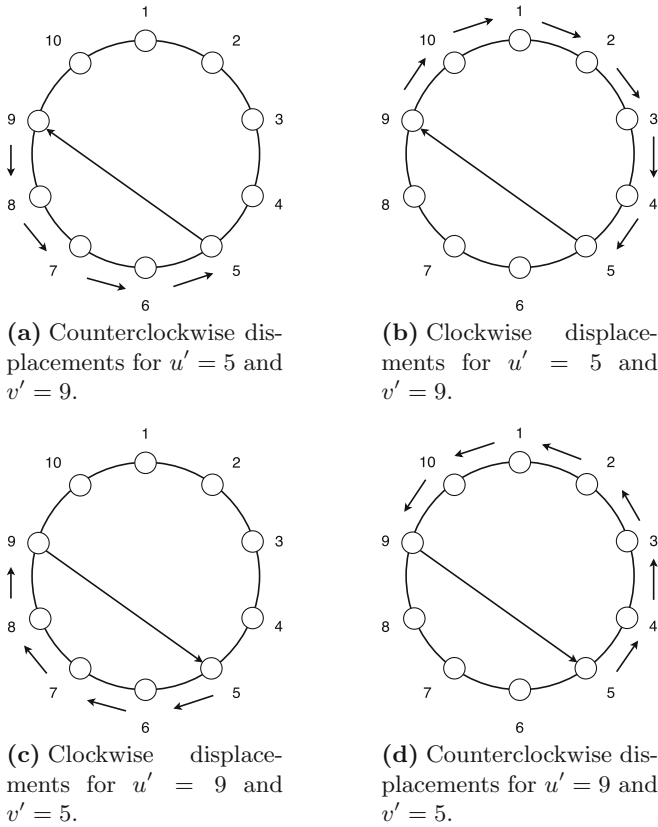
Keeping the population diverse is necessary to avoid premature convergence. Diversification is provided by mutation, introducing new genetic material into the population. Mutation works by probabilistically altering some of the genes of an individual. We tested three existing mutation schemes for permutations: *insertion*, *reduced 3-swap* and *cumulative swap*.

*Insertion* mutation operates over  $\rho_i$  (see Sect. 2.1). By manipulating  $\rho_i$ , *insertion* models the process of reallocating the guest vertex embedded at the host vertex  $u'$  to the vertex  $v'$ , while displacing the embedded vertices between  $u'$  and  $v'$ . Both host vertices  $u'$  and  $v'$  are randomly chosen. Given the cyclic nature of the embeddings for the CBSP, there are actually two sections of vertices that can be considered to be the *section in between*  $u'$  and  $v'$ : one section implies clockwise displacements and the other counterclockwise displacements. The insertion mutation will affect only the smaller section, i.e., the one with fewest vertices, which corresponds to the minimum length path between  $u'$  and  $v'$  in  $C_n$ . Figure 1 illustrates this by representing  $\rho_i$  as a cyclic permutation in order to reflect the cyclic nature of the embedding it encodes. As it can be inferred, any change in  $\rho_i$  must be properly reflected in  $\varphi_i$  by updating the labels, i.e., host vertices of the guest vertices embedded in the affected section.

*Reduced 3-swap* mutation picks three random vertices. The labels of those nodes are exchanged in every possible way, giving as a result five new solutions. The individual is then replaced by the best of those solutions, even if its fitness is worse than the current one. This can be seen as a subneighborhood from the 3-swap neighborhood, i.e., all solutions at Hamming distance equal to three from the current solution.

*Cumulative swap* performs  $n/2$  iterations (steps). At each iteration, with probability  $prob_c$  a pair of random vertices is picked. It is evaluated if the fitness of the individual would be improved by exchanging the labels of those vertices. If so, the labels are actually exchanged. Cumulative swap can be seen as a random up-hill walk of limited length.

One of the differences in the application of one or other mutation scheme is the role of the mutation probability  $prob_m$ . In the case of *reduced 3-swap* and *insertion*, the mutation probability acts at individual level, i.e., it is decided only

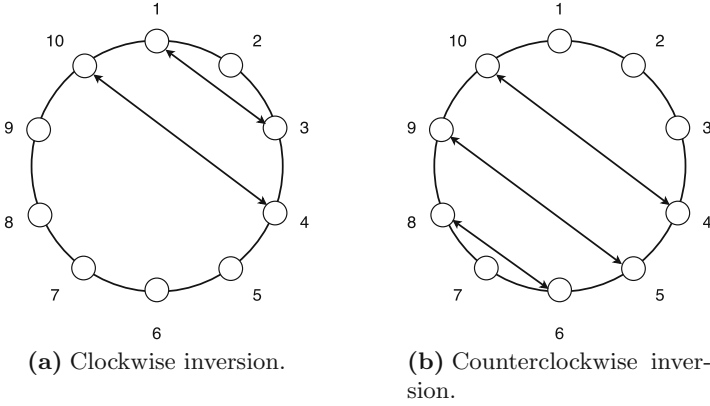


**Fig. 1.** *Insertion* mutation. Numbers represent the vertices of permutation  $\rho_i$ . Example in Fig. 1(a) corresponds to counterclockwise insertion when  $u' < v'$ , performing 5 steps. Also for  $u' < v'$ , clockwise insertion will perform 7 steps, as shown in Fig. 1(b), therefore counterclockwise insertion is preferred. For  $u' > v'$ , clockwise insertion will perform 5 steps, while 7 steps will be required by counterclockwise insertion.

once per generation if an individual will be mutated or not. Meanwhile, in *cumulative swap* little probabilistic mutations occur up to  $n/2$  times per individual. From this follows that, when using *insertion* or *reduced 3-swap* mutations some individuals will remain unchanged, approximately  $1 - \text{prob}_m \cdot \mu$ . The mutated individuals will present variable size mutations in the case of *insertion*, and uniform size mutations (exactly 3) in the case of *reduced 3-swap*. In *cumulative swap* it is likely that all individuals will mutate, but the amount of genes affected will vary within the population.

## 2.5 Inversion

The *inversion* phase is independent of the mutation one. In a similar way to the *reduced 3-swap* and *insertion* mutations, it is probabilistically applied at



**Fig. 2.** Inversion over  $\varphi_i$ , numbers represent the permutation labels. In Fig. 2(a) a clockwise inversion between vertices  $u = 10$  and  $v = 4$  would perform two exchanges of labels. Figure 2(b) shows the respective counterclockwise inversion which performs three exchanges of labels, therefore clockwise inversion is preferred.

individual level, so some individuals could remain unchanged. Given the nature of this operator, the number of changed genes in the affected individuals will be variable. *Inversion* operator consists in selecting two random vertices, and reversing the order of appearance of the labels in the section between them (inclusively). This is achieved by consecutive exchanges in the  $\varphi_i$  representation. In Fig. 2,  $\varphi_i$  is represented as a cyclic permutation to illustrate this process. Similarly to *insertion*, the cyclic feature of CBSP embeddings is considered, and *inversion* can operate clockwise or counterclockwise, preferring always the option implying the minimal number of exchanges.

### 2.6 Survival Strategy

The survival strategy applied was  $(\mu + \lambda)$ . All individuals in populations  $P$  and  $O$  are merged and sorted in nondecreasing order according with their fitness values. Then, the first  $\mu$  individuals are chosen to become the parent population  $P$  for the next generation.

### 2.7 Local Search

Local search is applied only to the best individual in the survivor population. The neighborhood employed was the one induced by the 2-swap operator, i.e., all solutions resulting from swapping the labels of two vertices in  $\varphi_i$ . It is visited in a random order, using the first-improvement move strategy. The local search phase ends when a local optimum is reached or after a maximal number of iterations was performed (*tries*).

**Table 1.** Input parameter values for the MA algorithms.

Parameter		Value	Parameter		Value
Population size	$\mu$	20	Inversion rate	$prob_i$	0.240
Crossover rate	$prob_c$	0.788	Local search iterations	$tries$	10
Mutation rate	$prob_m$	0.543	Evaluation function calls	$T$	4.0E+08

### 3 Experimental Results

We experimented with the full set of 24 MA versions corresponding to all the possible combinations of operators, with a maximal number ( $T$ ) of calls to the fitness function as stop criterion. A set of 25 topologically diverse and representative instances (see Table 3) belonging to three different types was used: Cartesian products, paths and cycles, and Harwell-Boeing graphs. All the MA versions were tested using a fixed set of parameter values (Table 1) obtained from the literature and from our a priori experiments using the *irace* R package for automatized algorithm tuning. Details on this matter are not included here due to the space limitations, but they are available online.<sup>2</sup>

For comparing the algorithms in terms of solution quality the overall relative root mean square error (O-RMSE) was computed for  $R = 31$  runs, with respect to the best-known solutions for the  $|\mathcal{T}| = 25$  tested instances, see (2). Those solutions were provided either by MACH or by any of our 24 memetic algorithms. The O-RMSE among all instances  $t \in \mathcal{T}$  was calculated as:

$$\text{O-RMSE} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} 100\% \sqrt{\left( \sum_{r=1}^R \left( \frac{\text{Cbs}_r(t) - \text{Cbs}^*(t)}{\text{Cbs}^*(t)} \right)^2 \right) / R}, \quad (2)$$

where  $\text{Cbs}_r(t)$  is the best solution quality achieved by the algorithm at execution  $r$ , and  $\text{Cbs}^*(t)$  is the best-known quality solution for instance  $t \in \mathcal{T}$ . An O-RMSE equal to 0% means the algorithm achieved the best known solution quality in all the  $R$  executions, and therefore it is the preferred value.

We also performed statistical significance analysis by the following methodology. The normality of data distributions was evaluated by the *Shapiro-Wilk* test. *Bartlett's* test was implemented to determine whether the variances of the normally distributed data were homogeneous or not. *ANOVA* test was applied in the case variance homogeneity was present and *Welch's t* parametric tests on the contrary. Meanwhile, *Kruskal-Wallis* test was implemented for non-normal data. In all cases the significance level considered was 0.05.

In order to identify the combination of operators corresponding to the different memetic algorithms we assigned keys to the tested operators, then those keys were used to construct a unique MA configuration identifier. The operator keys are *a)* for selection: *stochastic* (S1), *roulette* (S2), *random* (S3) and *binary*

<sup>2</sup> <http://www.tamps.cinvestav.mx/~ertello/cbsp-ma.php>.



**Table 2.** Results for the 24 MA tested versions.

#	Algorithm	Op. configuration	O-RMSE (%)	Avg. ex. time (s)	$G_{best}$ time (s)
1	MA-10	S2_C2_M1	5.297	87.434	19.886
2	MA-22	S4_C2_M1	5.637	87.133	20.623
3	MA-16	S3_C2_M1	5.854	86.427	19.954
4	MA-04	S1_C2_M1	5.945	87.569	19.728
5	MA-19	S4_C1_M1	6.030	86.911	17.741
6	MA-13	S3_C1_M1	6.609	86.230	17.942
7	MA-07	S2_C1_M1	6.693	87.238	18.808
8	MA-01	S1_C1_M1	6.715	87.339	18.587
9	MA-05	S1_C2_M2	10.022	85.434	21.414
10	MA-17	S3_C2_M2	10.065	84.391	23.983
11	MA-23	S4_C2_M2	10.237	85.064	22.840
12	MA-11	S2_C2_M2	10.583	85.353	22.958
13	MA-14	S3_C1_M2	10.626	84.556	22.129
14	MA-02	S1_C1_M2	11.092	85.573	22.103
15	MA-08	S2_C1_M2	11.383	85.523	21.973
16	MA-20	S4_C1_M2	11.389	85.223	21.235
17	MA-06	S1_C2_M3	12.749	97.562	24.883
18	MA-12	S2_C2_M3	12.872	97.463	24.396
19	MA-18	S3_C2_M3	13.182	96.630	23.779
20	MA-24	S4_C2_M3	13.449	97.210	25.685
21	MA-09	S2_C1_M3	14.116	97.191	23.832
22	MA-03	S1_C1_M3	14.285	97.305	23.219
23	MA-21	S4_C1_M3	15.067	96.952	22.853
24	MA-15	S3_C1_M3	15.077	96.377	24.367
25	MACH	N/A	21.050	2.09	N/A

*tournament* (S4); *b*) for crossover: *cyclic* (C1) and *order-based* (C2); and *c*) for mutation: *insertion* (M1), *reduced 3-swap* (M2) and *cumulative swap* (M3). Since all versions consider only  $(\mu + \lambda)$  as survival strategy there is no need to assign a key for it.

Table 2 presents our algorithms ranked according to their performance in terms of solution quality. MACH, which ranked last after all the MA, is also included as reference. Table 2 includes the rank of the algorithm (#), the configuration of genetic operators, the associated O-RMSE value, the average total running time (in seconds) and the average time in which the reported best found solution was reached by the algorithm. Since MACH is a constructive approach, only its average total time is reported.

The results in Table 2 suggest that the recombination and mutation schemes are more decisive than the selection, since the former operators induce the most remarkable grouping, indicated by the dashed lines. Despite algorithms including order-based crossover being better performing than their counterparts implementing cyclic crossover, it is the mutation operator the one having the higher influence over the final solution quality reached by the MA. Focusing on O-RMSE values, we found that the wider performance gap (of almost 5% O-RMSE) is observed between the algorithms implementing *insertion* mutation (M1) and the rest, while the gap between *cyclic* crossover (C1) or *order-based* crossover (C2) rarely surpasses 1%. From Table 2 it can be inferred that the top 3 MA configurations are quite similar in solution quality, total running time and time to find their best solution. The statistical significance analysis showed that, for the instance set being tested, our top 3 MA configurations are statistically indistinguishable from each other in terms of solution quality. This is not surprising since they differ only in the selection scheme. Moreover, the three of them are able to provide better solutions than MACH. Even the worst performing of our MA versions (MA-15) can provide better solutions than MACH. MA-15 has a O-RMSE value of 15.077%, meanwhile the O-RMSE of MACH surpasses 20%.

Although all the Memetic Algorithms take longer time than MACH, it is worth noting that MACH solution quality cannot be improved by employing a longer running time. It is also observable that all of our algorithms stopped finding improving solutions at an early stage of their total running time. Since there are some instances for which the optimal solutions or upper bounds were not always reached, this may be an indicator of premature convergence. While mutation and inversion are diversification mechanisms their effect may be diluted by the survival strategy. Once a locally optimal individual is reached, it will remain in the population in next generations and its genes are likely to keep proliferating in the population, until a fitter individual appears. Meanwhile, the less fit individuals will disappear from the population and diversity may be lost in preference of individuals becoming (probably) a locally optimal solution.

Table 3 presents the results of MA-10, the one with the best performance, compared with the state of the art. Only MACH is considered for the comparison, since it has been experimentally shown better than GVNS [6]. For each of the 25 instances in the set we present its number of vertices ( $|V|$ ), number of edges ( $|E|$ ), density ( $d = 2|E|/|V|(|V| - 1)$ ) and value of the optimum or upper bound (UB/Opt\*). Those values were assessed according to the graph topology: upper bound formula for the Cartesian products [9]; optimal value formula (marked by the symbol \*) for path, cycle, wheel and  $k$ -th power of cycle topologies [2, 9], and the general graph upper bound formula [9] for the Harwell-Boeing graphs.

Our best MA is compared to MACH [6], including the minimal of the solution cost values (*Best*) found among 31 executions, average and standard deviation of the those values (*Std*), and average time to reach the reported solutions. The last column (MA-10/MACH) corresponds to the result of the statistical significance test performed. Instances where MA-10 results present improvements with statistical significance with respect to those achieved by MACH are indicated by

**Table 3.** Performance comparison of our best performing MA (MA-10), with respect to the state-of-the-art method.

Graph	V	E	$d$	$UB/Opt^*$	MACH				MA-10 (S2_C2_M1)				MA-10/ MACH
					$Best$	$Avg$	$Std$	$T$	$Best$	$Avg$	$Std$	$T$	
p9p9	81	144	0.04	720	944	1254.77	183.07	0.00	<b>516</b>	585.68	96.65	3.51	+
c9c9	81	162	0.05	873	991	1283.65	131.95	0.01	<b>873</b>	961.52	85.73	6.30	+
p9c9	81	153	0.05	7434	794	794.00	0.00	0.00	<b>745</b>	805.81	73.38	5.62	*
p9k9	81	396	0.12	7362	<b>1728</b>	1728.00	0.00	0.01	<b>1728</b>	1728.00	0.00	1.13	*
c9k9	81	405	0.13	7434	<b>1809</b>	1809.00	0.00	0.01	<b>1809</b>	1809.00	0.00	0.68	*
k9k9	81	648	0.20	8370	9454	9533.32	43.63	0.02	<b>8280</b>	8605.81	270.05	21.87	+
path100	100	99	0.02	99*	<b>99</b>	99.00	0.00	0.00	<b>99</b>	99.00	0.00	7.48	*
cycle100	100	100	0.02	100*	<b>100</b>	100.00	0.00	0.00	<b>100</b>	144.65	56.29	5.13	-
wheel100	100	198	0.04	2600*	<b>2600</b>	2600.00	0.00	0.01	<b>2600</b>	2633.42	45.94	11.71	-
cPow100-10	100	1000	0.20	5500*	5598	5703.74	68.71	0.04	<b>5500</b>	5500.00	0.00	11.89	+
cPow100-2	100	200	0.04	300*	<b>300</b>	302.52	2.42	0.00	<b>300</b>	385.16	155.97	5.16	+
can_24	24	68	0.25	425	220	255.03	16.01	0.01	<b>182</b>	182.00	0.00	0.18	+
ibm32	32	90	0.18	743	493	540.35	22.94	0.01	<b>405</b>	411.84	8.18	1.84	+
bcsplr01	39	46	0.06	460	102	115.58	8.53	0.01	<b>98</b>	102.58	5.82	4.80	+
bcsstk01	48	176	0.16	2156	1157	1339.74	111.74	0.02	<b>936</b>	954.45	13.43	21.32	+
bcsplr02	49	59	0.05	737	158	176.23	20.03	0.02	<b>148</b>	151.94	5.93	10.53	+
curtis54	54	124	0.09	1705	448	633.61	89.46	0.03	<b>411</b>	422.90	20.66	8.54	+
will57	57	127	0.08	1841	408	436.55	45.42	0.04	<b>335</b>	345.29	21.55	0.60	+
impcol_b	59	281	0.16	4215	2462	2838.13	242.00	0.07	<b>1822</b>	1829.74	9.90	0.16	+
ash85	85	219	0.06	4708	1232	1422.16	142.17	0.14	<b>919</b>	1036.58	89.64	22.89	+
nos4	100	247	0.05	6237	1181	1397.48	222.87	0.07	<b>1031</b>	1031.00	0.00	6.03	+
bcsplr03	118	179	0.03	5325	766	926.90	76.74	0.25	<b>664</b>	713.19	53.72	14.67	+
can_292	292	1124	0.03	82333	23288	25703.48	1678.87	7.13	<b>15763</b>	18982.10	2148.92	75.81	+
bcsstk06	420	3720	0.04	391532	65017	84469.87	8027.79	30.83	<b>55140</b>	67875.65	10377.12	177.82	+
impcol_d	425	1267	0.01	134935	25677	35355.19	4596.68	13.48	<b>12232</b>	15932.90	3170.52	71.47	+

Note: The overall winner MA-10 scored 18 victories (+), 2 defeats (-), and 5 ties (\*).

the + symbol, meaning a *victory* for MA-10. The contrary case, a *defeat* for MA-10, is marked with the - symbol. Results with no statistical significant difference are counted as ties and marked with the \* symbol. The best known solution for each instance is highlighted in bold.

For most of the tested instances MA-10 is able to consistently produce significantly better solutions with respect to those furnished by MACH. Our only defeats correspond to graphs for which MACH is specially suitable to solve: highly regular topologies with low densities, such as cycles and paths. However, MA-10 shows dominance for regular topologies with growing densities (see Cartesian products) and more general graphs, such as Harwell-Boeing graphs. It is also noticeable that our algorithm reached solutions with CBS values under the theoretical upper bounds (or equal to the optimal know values) for all instances.

## 4 Conclusions and Future Work

A set of 24 different MA configurations for solving the CBSP was evaluated. The experiments presented revealed that the top three MA configurations, which are

statistically indistinguishable from each other, can provide significantly better results than MACH [6] for 18 out of 25 tested instances. Furthermore, the best MA version (MA-10) achieved optimal results for the 5 instances with known exact solution values. For the remaining 20 instances with unknown exact optimal values, MA-10 was able to establish 18 new upper bounds and to equal 2 other.

Confirming the presence of premature convergence in our MA, as well as identify its causes, are certainly interesting future research topics. Exploring other alternatives for the survival strategy, as well as using MACH as an initialization operator, could be promising directions to improve the performance our MA. It is also interesting to consider the implementation of automatic schemes allowing the algorithm to self-adapt its own operators, instead of defining them from the beginning of the search.

**Acknowledgments.** The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-Tamaulipas.

## References

1. Bhatt, S.N., Leighton, F.T.: A framework for solving VLSI graph layout problems. *J. Comput. Syst. Sci.* **28**(2), 300–343 (1984). [https://doi.org/10.1016/0022-0000\(84\)90071-0](https://doi.org/10.1016/0022-0000(84)90071-0)
2. Chen, Y., Yan, J.: A study on cyclic bandwidth sum. *J. Comb. Optim.* **14**(2), 295–308 (2007). <https://doi.org/10.1007/s10878-007-9051-y>
3. Chung, F.R.K.: Labelings of graphs (Chap. 7). In: Beineke, L.W., Wilson, R.J. (eds.) *Selected Topics in Graph Theory*, vol. 3, pp. 151–168. Academic Press, Cambridge (1988)
4. Davis, L.: Applying adaptive algorithms to epistatic domains. In: *Proceedings of the 9th IJCAI*, vol. 1, pp. 162–164. Morgan Kaufmann Publishers Inc., San Francisco (1985)
5. Diaz, J., Petit, J., Serna, M.: A survey of graph layout problems. *ACM Comput. Surv.* **34**(3), 313–356 (2002). <https://doi.org/10.1145/568522.568523>
6. Hamon, R., Borgnat, P., Flandrin, P., Robardet, C.: Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum. *J. Complex Netw.* **4**(4), 534–560 (2016). <https://doi.org/10.1093/comnet/cnw006>
7. Harper, L.: Optimal assignment of numbers to vertices. *J. SIAM* **12**(1), 131–135 (1964). <https://doi.org/10.1137/0112012>
8. Jaccard, P.: The distribution of the flora in the alpine zone. *New Phytol.* **11**(2), 37–50 (1912). <https://doi.org/10.1111/j.1469-8137.1912.tb05611.x>
9. Jianxiu, H.: Cyclic bandwidth sum of graphs. *Appl. Math. J. Chin. Univ.* **16**(2), 115–121 (2001). <https://doi.org/10.1007/s11766-001-0016-0>
10. Li, Y., Liang, Y.: Compressed sensing in multi-hop large-scale wireless sensor networks based on routing topology tomography. *IEEE Access* **6**, 27637–27650 (2018). <https://doi.org/10.1109/ACCESS.2018.2834550>
11. Liberatore, V.: Multicast scheduling for list requests. In: *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 1129–1137. IEEE (2002). <https://doi.org/10.1109/INFCOM.2002.1019361>

12. Monien, B., Sudborough, I.H.: Embedding one interconnection network in another. In: Tinhofer, G., Mayr, E., Noltemeier, H., Syslo, M.M. (eds.) *Computational Graph Theory*, vol. 7, pp. 257–282. Springer, Vienna (1990). [https://doi.org/10.1007/978-3-7091-9076-0\\_13](https://doi.org/10.1007/978-3-7091-9076-0_13)
13. Oliver, I., Smith, D., Holland, J.: A study of permutation crossover operators on the traveling salesman problem. In: *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Application*, pp. 224–230. L. Erlbaum Associates Inc., Hillsdale (1987)
14. Satsangi, D., Srivastava, K., Gursaran, S.: General variable neighbourhood search for cyclic bandwidth sum minimization problem. In: *Proceedings of the Students Conference on Engineering and Systems*, pp. 1–6. IEEE Press, March 2012. <https://doi.org/10.1109/SCES.2012.6199079>
15. Ullman, J.D.: *Computational Aspects of VLSI*. Computer Science Press, Rockville (1984)
16. Yuan, J.: Cyclic arrangement of graphs. In: *Graph Theory Notes of New York*, pp. 6–10. New York Academy of Sciences (1995)



# Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic

Renato Tinós<sup>1</sup>(✉), Keld Helsgaun<sup>2</sup>, and Darrell Whitley<sup>3</sup>

<sup>1</sup> Department of Computing and Mathematics, University of São Paulo, Ribeirão Preto, Brazil  
`rtinos@ffclrp.usp.br`

<sup>2</sup> Department of Computer Science, Roskilde University, Roskilde, Denmark  
`keld@ruc.dk`

<sup>3</sup> Department of Computer Science, Colorado State University, Fort Collins, USA  
`whitley@cs.colostate.edu`

**Abstract.** The Lin-Kernighan-Helsgaun (LKH) algorithm is one of the most successful search algorithms for the Traveling Salesman Problem (TSP). The core of LKH is a variable depth local search heuristic developed by Lin and Kernighan (LK). Several improvements have been incorporated to LKH along the years. The best results reported in the literature were obtained by an iterative local search version known as multi-trial LKH. In multi-trial LKH, solutions generated by soft restarts of the LK heuristic are recombined using Iterative Partial Transcription (IPT). We show that IPT can be classified as a partition crossover. Partition crossovers use the features common to the parents to decompose the evaluation function. Recently, a new generalized partition crossover, known as GPX2, was proposed for the TSP. We investigate the use of GPX2 in multi-trial LKH and compare it to multi-trial LKH using IPT. Results of experiments with 11 large instances of the TSP indicate that LKH with GPX2 outperforms LKH with IPT in most of the instances, but not in all of them.

**Keywords:** Traveling Salesman Problem · Recombination operator  
Heuristic search · Evolutionary combinatorial optimization

## 1 Introduction

The *Traveling Salesman Problem* (TSP) is one of the most investigated problems in Optimization [2]. Applications of the TSP can be found in the most diverse areas, such as Logistics, Bioinformatics, and Planning. Given a complete weighted graph  $G(V, E)$ , where  $V$  is a set of  $n$  vertices (cities) and  $E$  contains edges between every pair of vertices in  $V$ , the objective is to find the shortest Hamiltonian cycle. The evaluation of a solution (tour)  $\mathbf{x}$  is given by:

$$f(\mathbf{x}) = w_{x_n, x_1} + \sum_{i=1}^{n-1} w_{x_i, x_{i+1}} \quad (1)$$

where  $w_{x_i, x_j}$  is the weight of the edge between vertices  $v_{x_i}$  and  $v_{x_j}$  in  $V$ .

There are very good exact methods for the TSP, e.g., Concorde [2]. Concorde solves instances of the symmetric TSP with hundreds of cities in seconds. However, the TSP is NP-hard and, as a consequence, heuristic methods have been required for solving large TSP instances. One of the most successful heuristics for the TSP is the *Lin-Kernighan-Helsgaun* (LKH) algorithm [4, 5]. LKH holds the record for several large instances of the TSP, some of them with more than 100,000 vertices. The best results of LKH reported in the literature were obtained by an iterative local search version known as multi-trial LKH. In multi-trial LKH, solutions generated by soft restarts of the LK heuristic are recombined using an efficient crossover operator, called *Iterative Partial Transcription* (IPT).

Recently, a new *generalized partition crossover*, known as GPX2, was proposed for the TSP [14]. Partition crossovers are deterministic recombination operators that use the features common to the parents to decompose the evaluation function [17]. The main contributions of this work are two. First, we show that IPT [10] is a kind of partition crossover. Second, we investigate the use of GPX2 in multi-trial LKH and compare it with multi-trial LKH using IPT. Unlike previous works with generalized partition crossovers [3, 14], GPX2 is used here inside LKH. Before, generalized partition crossovers were used to recombine solutions generated by LKH but the offspring were not reinserted in LKH. Here, IPT is replaced by GPX2 inside LKH, which results in a different heuristic.

## 2 LKH Algorithm

LKH is an iterated local-search algorithm based on the Lin-Kernighan heuristic (LK) [9]. The local search performed by LK is based on  $k$ -opt moves. Given a tour  $\mathbf{x}$ , a  $k$ -opt move replaces  $k$  edges from  $\mathbf{x}$  in order to create a solution  $\mathbf{y}$  where  $f(\mathbf{y}) < f(\mathbf{x})$ . A  $k$ -opt move is incrementally obtained using basic moves, e.g., 2-opt, while the cumulative gain remains positive. Some heuristics (e.g., limiting the search to a subset of edges to the nearest neighbors of a node) are adopted in order to reduce the cost of the moves.

LK is an effective local search algorithm; implementations capable of finding solutions with typical cost 1–2% above the optimum cost were reported in the literature. A much more effective implementation of LK was reported in [4]. This implementation, called LKH, is able to find optimal solutions for large TSP instances with very high frequency [5]. Several improvements have been incorporated to LKH along the years. We present some of them in the following:

- **General  $k$ -opt moves:** In LK, moves are obtained by 2-opt or 3-opt moves followed by a sequence of 2-opt moves. Non-sequential moves are tried at the end if the sequential moves did not improve the original solution. LKH-1 [4] uses 5-opt sequential moves to create the sequence of basic moves. In LKH-2 [5], the basic moves are  $k$ -opt moves where  $k$  can be any integer greater than 1 and smaller than  $n$ . The moves are sequential, but non-sequential moves can also be tried during the search.

- **Partitioning:** Large instances of TSP are decomposed into smaller subproblems. Then, the solutions of the subproblems are used to improve the solutions of the original instance.
- **Candidate set criterion:** Instead of using the cost of an edge, the  $\alpha$ -measure is used to evaluate the quality of an edge. The  $\alpha$ -value of an edge  $e$  is computed as the increase of the cost of a minimum 1-tree when this tree is required to contain  $e$ . By restricting the search to a small number of neighbors of a node obtained according to a distance based on the  $\alpha$ -measure, the time complexity is reduced.
- **Multiple trials:** In each run, the local optimum obtained is perturbed in order to generate a new initial solution for the LK strategy. Each run  $r$  of the multi-trial LKH is composed of  $t$  trials (Fig. 1). The use of multiple trials allows the use of strategies that explore information from different solutions in order to create a new solution. Three of them are presented in the following.
- **Backbone-guided search:** Edges of solutions previously obtained in different trials compose a set of candidate edges for the current trial.
- **Recombination of solutions:** Local optima share many partial solutions. The TSP has a multi-funnel structure, where many edges are common to the optima located in the same funnel [11]. Recombination operators are generally used in population meta-heuristics. However, recombination can also be used to merge solutions generated in different runs of an algorithm, in different trials of an iterated local search, or generated by different algorithms. In LKH-2, tours obtained in different trials and runs are recombined using IPT. Figure 1 shows how recombination is used in multi-trial LKH.
- **Genetic Algorithm:** Instead of storing only the best current solution, a population of solutions obtained in different runs can be stored. When the population size is different from one, a simple genetic algorithm is executed. For each run, the best solution is stored in the population if its fitness is different from the other solutions in the population. After each run, the stored solutions are selected and recombined using a variant of the Edge Recombination Crossover (ERX) [18]. It is important to observe that IPT is still used as shown in Fig. 1; ERX is used only after the end of each run to recombine the solutions of the population.

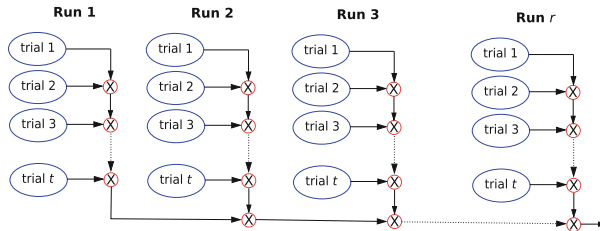


Fig. 1. Multi-trial LKH. The symbol  $\otimes$  indicates a recombination operation.



### 3 Partition Crossover

*Partition crossover* (PX) is a deterministic recombination operator that can be applied in problems where the cost function,  $f(\mathbf{x})$ , is written as the sum of  $m$  subfunctions  $f_i(\mathbf{x})$ , i.e.:

$$f(\mathbf{x}) = \sum_{i=1}^m f_i(\mathbf{x}) \quad (2)$$

where solution  $\mathbf{x}$  is given by an  $n$ -dimensional vector and  $m > 0$ . Suppose that an offspring  $\mathbf{z}$  is generated by recombining two solutions  $\mathbf{x}$  and  $\mathbf{y}$ . If partition crossover is employed, then we can write:

$$f(\mathbf{z}) = \sum_{i \in S^x} f_i(\mathbf{x}) + \sum_{i \in S^y} f_i(\mathbf{y}) \quad (3)$$

where  $|S^x| + |S^y| = n$ . The subsets  $S^x$  and  $S^y$  contain the indexes of the decision variables inherited respectively from parents  $\mathbf{x}$  and  $\mathbf{y}$ . The decomposition of the cost function is derived from two properties of partition crossover operators [12]: (i) the recombination is “respectful”, i.e., the offspring inherits all features that are common to both parents; (ii) the recombination “transmits alleles”, i.e., the offspring is composed only of features found in the parents. As a result of the properties of PX, the evaluation of the offspring is more correlated with the evaluation of the parents than in traditional recombination operators. Besides, if the parents are local optima with respect to a local search operator, then the offspring are guaranteed to be piecewise locally optimal under this local search operator. It was observed in different applications [13–16] that offspring are also very often true local optima when PX is employed.

The first step of PX is to remove all the features common to both parents. In the TSP, the features of a solution represented by  $\mathbf{x}$  are the edges between two consecutive cities in  $\mathbf{x}$ . Define the union graph  $G_u = G_x \cup G_y$ , where the graphs  $G_x$  and  $G_y$  represent the parent solutions. The graph  $G'_u$  is obtained by removing the common edges from  $G_u$ .

**Definition 1.** A **candidate component** is made up of one or more connected subgraphs of  $G'_u$ .

**Definition 2.** A **recombining component** is a candidate component such that: (1) it contains  $z$  vertices, where  $2x$  vertices are portals that connect to other recombining components by common edges, and the remaining  $z - 2x$  vertices only connect to vertices inside the recombining component; (2) exactly  $x$  vertices that are portals serve as “entry” points, and  $x$  vertices that are portals serve as “exit” points to other recombining components; (3) the two parent solutions must enter and exit the recombining component at exactly the same entry and exit vertices.

Inheriting one of the recombining components from one or another parent does not influence the evaluation of other recombining components. In other

words, the recombining components are subsets of features with independent evaluation. If  $p$  recombining components are found, there are  $2^p$  different ways of combining the components to create an offspring. PX selects the best partial solution (from one or another parent) for each recombining component. Thus, the best of  $2^p$  reachable offspring is found by PX.

**Definition 3.** A **partition crossover** is a recombination operator that: (1) finds recombining components in the graph obtained by removing the common edges from the union graph  $G_u$ ; (2) evaluates the cost of the partial solutions (for each parent) inside each recombining component; (3) generates the offspring by selecting the best partial solutions (from one or another parent) inside the recombining parents.

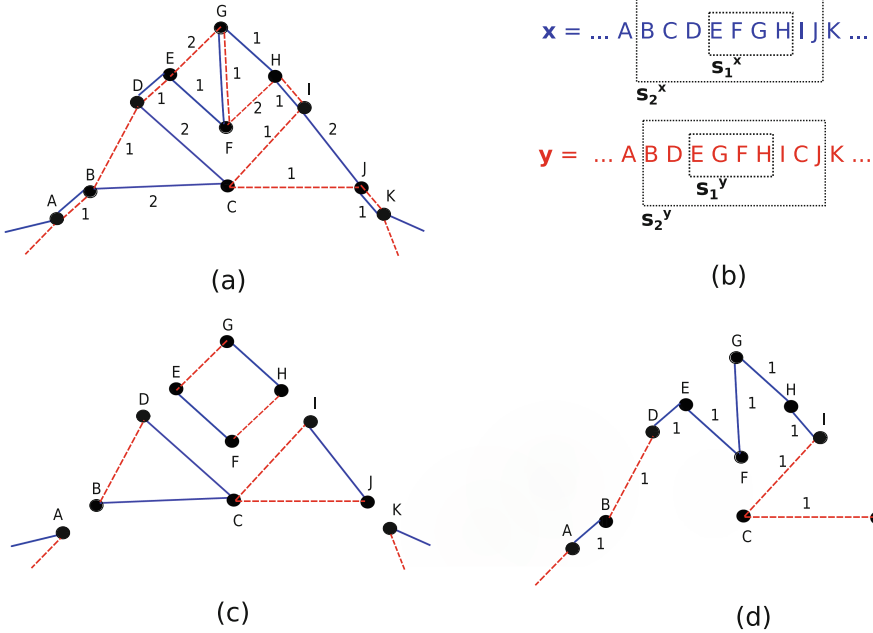
GPX2 is a PX developed for the symmetric TSP. According to the definition of PXs, IPT can also be classified as a PX operator (see next section). IPT and GPX2 differ in the way the recombining components are found.

### 3.1 IPT

IPT [10] works directly on the sequence representation of the tours. IPT searches for subchains in parents  $\mathbf{x}$  and  $\mathbf{y}$  with: (i) the same initial and final cities; (ii) composed of the same cities, but in different order. According to the PX terminology, the subset of cities in a subchain composes a recombining component. Each subchain can be independently evaluated. By selecting the best subchains, the reachable offspring with the best cost is found. The three main steps of IPT, written according to the definition of recombining components, are:

- **Removal of cities connected only to common edges:** if a city is connected to the same neighbors in  $\mathbf{x}$  and  $\mathbf{y}$ , then it can be removed from the tours, resulting in reduced sequences.
- **Finding recombining components in the reduced sequences:** suppose  $N_r$  is the size of the reduced sequences. Let  $v_s(v, \mathbf{x})$  be a vertex located  $s - 1$  positions from vertex  $v$  in  $\mathbf{x}$ . Start with  $s = 4$  (that is the minimum size of permutations that are different). For each vertex  $v \in \mathbf{x}$ , verify if  $v_s(v, \mathbf{x}) = v_s(v, \mathbf{y})$ , i.e., the subchains have the same initial and final cities. Subchains in both directions of  $\mathbf{y}$  must be tested. If the subchains in  $\mathbf{x}$  and  $\mathbf{y}$  are composed of the same cities, then the subset of indices in the subchains define a recombining component. Repeat, increasing  $s$  by 1, while  $s \leq N_r/2$ .
- **Creating the offspring:** for creating the offspring, select the best subchains in each recombining component and copy the cities connected only to common edges from one of the parents.

An example of IPT is presented in Fig. 2. In this example, IPT first finds a recombining component with 4 cities. The cost of subchain  $s_1^{\mathbf{x}}$  is smaller than the cost of  $s_1^{\mathbf{y}}$ . Thus, the offspring inherits  $s_1^{\mathbf{x}}$ . Then, it finds another recombining component with 5 cities. The cost of subchain  $s_2^{\mathbf{y}}$  is smaller than the cost of  $s_2^{\mathbf{x}}$ . Thus, the offspring inherits  $s_2^{\mathbf{y}}$ . The implementation of IPT in LKH-2 is very efficient. Despite of the fact that the worst case complexity is  $O(n^2)$ , the average time is linear in  $n$ .



**Fig. 2.** Examples of recombination by IPT and GPX2. Only the paths between cities  $A$  and  $K$  are shown (suppose that, for IPT,  $N_r > 20$ ). (a) Union graph composed of parents  $x$  (blue solid line) and  $y$  (red dashed line). (b) When IPT is applied, two subchains are identified for each parent. (c) When GPX2 is applied, candidate (connected) components are found after removing the common edges from the union graph. In this example, two candidate components are recombining components. (d) Offspring generated by IPT or GPX2. When compared to the parents, the offspring has better cost. (Color figure online)

### 3.2 GPX2

GPX1 [3] and GPX2 [14] work in the graph representation of the tours (Fig. 3). In GPX1, all candidate components linked to other parts of the graph  $G'_u$  by exactly two common edges are recombining components. Edges connecting a candidate component to other parts of the graph are *entries* for the tours in this candidate component. Thus, GPX1 finds only recombining components with two entries. The rest of the graph also composes a recombining component. Finally, the offspring is created by selecting, from one or another parent, the paths with the best cost inside each recombining component.

GPX2 presents 3 enhancements that allows to find much more recombining components than GPX1. Increasing linearly the number of recombining components,  $p$ , an exponentially larger number of reachable offspring are exploited. The enhancements are:

- **Exploring vertices of degree-4 as possible points for recombination:** GPX1 explores only common edges as possible connection points between

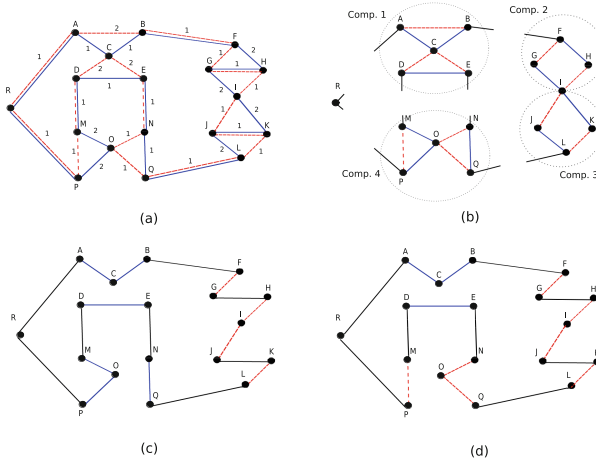
- recombining components. In GPX2, a “ghost node” is created for every degree-4 vertex in  $G_u$ . The original and ghost nodes are linked by a common edge with weight 0. Thus, by removing the common edges in the new union graph, some vertices of degree-4 in  $G_u$  become potential points for recombination. The number of recombining component is further increased by exploring both directions for tour  $y$  and by using an efficient data structure (Extended Edge Table) for storing the direct and reverse tours for parent  $y$ ;
- **Exploring candidate components with more than two entries:** All candidate components with 2 entries are recombining components. However, not all candidate components with more than 2 entries are recombining components. In order to test the candidate components, simplified graphs are built for the path of each parent inside the candidate component. If the simplified graphs for both parents are equal, then exchanging the paths still results in a Hamiltonian cycle for the offspring. Another test is executed for the case where a recombining component is nested inside a candidate component. If, after removing the already identified recombining components, the number of entries of the candidate component becomes 2, then the candidate component is a recombining component;
  - **Fusing candidate components:** Two candidate components that are not individual recombining components can be fused in order to create a recombining component. Two types of fusion are applied in GPX2. In fusion type 1, the fusion occurs between two candidate components that are neighbors. Then, the new candidate component is tested in order to verify if it is a recombining component or not. Cycles of fusion type 1 are repeated  $n_f$  times, obtaining each time larger candidate components. In fusion type 2, nested and intercalated candidate components are fused. Then, it is verified if the resulting component has 2 entries after removing the already identified recombining components. The procedure is repeated  $n_r$  times.

The time complexity for GPX2 is  $O(n)$  [14]. Examples of GPX2 are presented in Figs. 2 and 3. In Fig. 3, IPT finds 3 recombining components, while GPX2 finds 4 recombining components. As a consequence, IPT finds the best of  $2^3$  reachable offspring, while GPX2 finds the best of  $2^4$  reachable offspring. The tour found by GPX2 in this example is shorter than the tour found by IPT.

## 4 Results

In the experiments, LKH with IPT (LKH+IPT) is compared to LKH with GPX2 (LKH+GPX2). The version of LKH used is 2.0.8<sup>1</sup>. Here, LKH runs with the default parameters, except for the number of runs (10 or 50), number of trials (10 or 1000), and population size (equal to the number of runs). It is important to observe that, in LKH, the best results found in different runs are not independent

<sup>1</sup> In LKH version 2.0.8, tours may be recombined by GPX2 instead of IPT. The code for LKH version 2.0.8, that allows to reproduce the results presented in this paper, can be downloaded at <http://www.akira.ruc.dk/~keld/research/LKH/>.



**Fig. 3.** Examples of recombination by IPT and GPX2. (a) Union graph composed of parents  $x$  (blue solid line) and  $y$  (red dashed line). (b) GPX2 identifies 4 recombining components. Components 1 and 4 have 4 entries each. (c) IPT identifies 3 of the recombining components found by GPX2: 2, 3, and union of 1 and 4. (d) The offspring generated by IPT has cost 20. (e) The offspring generated by GPX2 has cost 18. (Color figure online)

(see Fig. 1). When GPX2 is used in LKH,  $n_f = 3$  and  $n_r = 1000$  (parameters used in fusion).

Experiments with 11 instances of four classes of the symmetric TSP are presented. The 2 instances of Class 1 are artificial instances used in the 8th DIMACS Implementation Challenge [8]. In E31k0, the locations of 31,623 cities are uniformly generated in a square of 1,000,000 by 1,000,000 units. In C10k0, the locations of 10,000 cities consist of clustered points in the same square. LKH currently holds the records for these instances [7]. The records for the remaining problems are reported in [1]. The 3 instances in Class 2 (pia3056, dke3097, and xqe3891) are from the VLSI TSP Collection. The 5 instances in Class 3 (tz6117, ym7663, ar9152, usa13509, usa115475) are formed using (Euclidean) distance between cities of different countries. The size  $n$  in the instances of classes 2 and 3 is given in the name of the instances. Finally, monalisa100K is an instance of the Art TSP Collection with  $n = 100,000$  vertices.

Due to the limitation of space, we only show the results for the experiments with 1000 trials and 50 runs (Table 2). Table 2 shows the percentage gap to the cost of the best solutions found in the literature [1, 7]; when the cost of the best solution found by an algorithm is equal to the best result reported in the literature, the number of runs needed for finding the best result is shown in parenthesis. Smaller results are better for the percentage gap, number of runs, and average running time. A summary of the comparison of the best cost found by the algorithms in all experiments is presented in Table 1.

We also tested two versions of LKH where both operators are used. In the “First IPT and then GPX2” version, GPX2 is applied after IPT. In the “First GPX2 and then IPT” version, IPT is applied after GPX2. For strategy “First IPT and then GPX2”, IPT is applied first to recombine the parents. If there is no improvement, GPX2 is then applied to recombine the parents. Otherwise, i.e., IPT generated an improvement, then GPX2 is applied to recombine the first parent with the offspring generated by IPT. The opposite occurs for strategy “First GPX2 and then IPT”, i.e., GPX2 is applied first. If an operator  $A$  is applied first and does not improve the best solution, but operator  $B$  does, this means that operator  $B$  found recombination opportunities missed by operator  $A$ . The results for those versions for the experiments with 1000 trials are shown in Table 3. Some observations can be made about the results.

In the experiments with versions “First IPT and then GPX2” and “First IPT and then GPX2”, GPX2 (applied after IPT) was able to find many recombination opportunities that improved the best solution in all experiments, except for instance ar9152 when the number of trials and runs is 10. For example, in the experiment with “First IPT and then GPX2” version applied to instance usa13509 when the number of trials and runs is 1000, GPX2 improved 713 times the best results (Table 3). IPT improved the best result 2358 times in this case. As IPT was applied first, it is clear that GPX2 found some recombination opportunities missed by IPT in this case. However, IPT (applied after GPX2) was not able to find recombination opportunities that improved the best solution, with exceptions for 2 instances in the experiments with 10 trials and 50 runs, and for 3 instances in the experiments with 1000 trials and 50 runs. The cases where IPT (applied after GPX2) was able to find recombination opportunities that improved the best solution can be explained by two main factors: the limit  $n_r$  used in fusion type 2 and the fact that there are different ways of finding the recombining components.

In general, GPX2 found more recombination opportunities that resulted in improvements of the best solution than IPT (see, for example, the number inside the parenthesis in Table 3). More efficient recombination generally results in better performance. When the number of trials and runs is 10, LKH+IPT found better solutions in experiments with 2 instances, while LKH+GPX2 found better solutions in experiments with 9 instances (Table 1). When the number of runs increased (and the number of trials was kept to 10), even better results were obtained by LKH+GPX2: it found better results for 10 out of 11 instances. Increasing the number of runs (from 10 to 50) resulted in more recombinations (Fig. 1); as a consequence, GPX2 found still more recombinations that resulted in improvements for the best solution.

LKH+GPX2 also resulted in better performance for the experiment with 1000 trials and 50 runs: LKH+GPX2 presented better performance in 8 out of 11 instances. However, LKH+IPT found better performance 3 times; for instance pia3056, LKH+IPT was able to find the literature best solution, while LKH+GPX2 was not. Thus, finding more recombination opportunities does not guarantee that LKH will perform better. A better solution obtained by recombination in a trial influences the solutions generated in the subsequent trials and

**Table 1.** Comparison between LKH+GPX2 and LKH+IPT (regarding the cost of the best solutions found by the algorithms). For each instance and experiment, the algorithm with better performance is indicated.

Problem	Experiment		
	10 trials, 10 runs	10 trials, 50 runs	1000 trials, 50 runs
pia3056	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>	LKH+IPT
dke3097	<b>LKH+GPX2</b>	LKH+IPT	<b>LKH+GPX2</b>
xqe3891	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>	LKH+IPT
tz6117	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>
ym7663	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>
ar9152	LKH+IPT	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>
C10k0	LKH+IPT	<b>LKH+GPX2</b>	LKH+IPT
usa13509	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>
E31k0	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>
monalisa100K	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>
usa115475	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>	<b>LKH+GPX2</b>

**Table 2.** Best cost gap (to the cost of the best results found in the literature) and average running time (in seconds) for LKH+GPX2 and LKH+IPT. The number of trials is 1000 and the number of runs is 50. The best results are in bold.

Problem	Best cost gap (%)		Average running time (s)	
	LKH+IPT	LKH+GPX2	LKH+IPT	LKH+GPX2
pia3056	<b>0 (run 5)</b>	0.0360	<b>56.01</b>	63.41
dke3097	0 (run 2)	<b>0 (run 1)</b>	<b>61.67</b>	75.83
xqe3891	<b>0 (run 2)</b>	0 (run 3)	<b>80.40</b>	104.65
tz6117	0 (run 29)	<b>0 (run 6)</b>	<b>188.36</b>	237.77
ym7663	0 (run 22)	<b>0 (run 4)</b>	<b>169.47</b>	194.24
ar9152	0.0140	<b>0.0130</b>	<b>723.72</b>	849.78
C10k0	<b>0 (run 16)</b>	0 (run 36)	<b>389.63</b>	406.23
usa13509	0 (run 34)	<b>0 (run 13)</b>	<b>331.73</b>	384.12
E31k0	0.0100	<b>0.0096</b>	1767.79	<b>1713.40</b>
monalisa100K	0.0220	<b>0.0110</b>	19901.08	<b>19061.51</b>
usa115475	0.0360	<b>0.0190</b>	13664.19	<b>13237.47</b>

runs. Recall that, in multi-trial LKH, the current best solution is employed to generate the soft restarts and the set of candidate edges. Thus, an initial solution that is not so good can be used by LK in order to generate a promising local optimum. For instances with 10,000 cities or more, LKH+IPT obtained better results for only one instance (C10k0). This is a clustered instance; in clustered

instances, most of the recombining components have two entries when two local optima are recombined. Recall that one of the most important properties of GPX2, when compared to IPT, is that it is able to find recombining components with more than two entries. If there are not many recombining components with more than two entries, e.g., in clustered instances, GPX2 and IPT generally have similar performance.

Despite the better results for the cost of the solutions, LKH+GPX2 generally resulted in higher mean running times (Table 2 shows the results for the experiments with 1000 trials). For the experiments with 1000 trials, LKH+IPT presented smaller mean time in 8 out of 11 instances (Table 2). LKH+GPX2 presented smaller mean time in the 3 largest instances: E31k0, monalisa100K, and usa115475.

**Table 3.** Number of times that both crossovers improved the best solution or only the second crossover improved the best solution. The number of trials is 1000 and the number of runs is 50. The results in parenthesis indicate the total number of improvements generated by the first crossover.

Problem	“First IPT, then GPX2”	“First GPX2, then IPT”
	GPX2 (total for IPT)	IPT (total for GPX2)
pia3056	21 (271)	0 (328)
dke3097	44 (342)	0 (383)
xqe3891	68 (361)	0 (417)
tz6117	153 (1256)	0 (1330)
ym7663	221 (1175)	0 (1321)
ar9152	65 (2094)	2 (2003)
C10k0	69 (2277)	0 (2381)
usa13509	713 (2358)	0 (2710)
E31k0	3003 (4036)	0 (5934)
monalisa100K	10997 (3374)	1 (12520)
usa115475	8206 (9981)	12 (13631)

## 5 Conclusions

Previous versions of multi-trial LKH use only IPT to recombine solutions generated in different trials and runs. Here, we investigated the use of GPX2 in LKH. The experimental results indicated that GPX2 finds some recombination opportunities that are missed by IPT. This impacts the quality of solutions generated by recombination. However, finding better offspring by recombination does not necessarily guarantee better performance for the iterative local search; solutions generated by recombination impacts the soft restarts of multi-trial LKH.

In the experiments with 10 trials and 10 runs, LKH+GPX2 obtained better results for 9 instances, while LKH+IPT obtained better results for 2 instances.



When the number of runs increased to 50 (and the number of trials was kept to 10), LKH+GPX2 obtained even better results: LKH+GPX2 obtained better results for 10 out of 11 instances. Finally, in the experiments with 50 runs and 1000 trials, LKH+GPX2 obtained better results for 8 instances, while LKH+IPT obtained better performance for 3 instances. Despite the better results for the cost of the solutions, LKH+IPT generally resulted in smaller mean running time. For the experiments with 50 runs and 1000 trials, LKH+GPX2 resulted in smaller mean time only for the three largest instances.

As a consequence of this work, solutions may be now recombined by GPX2 instead of IPT in LKH version 2.0.8 [7]. GPX2 can be easily adapted to asymmetric TSP. However, when used with LKH, no modifications are needed; the asymmetric instance is transformed into a symmetric instance twice the size. Optimizing the running time of LKH+GPX2 is a possible future work. There is room for improvement: for example, data structures that explore the implementation of LKH can be proposed. Another future work can be the investigation of LKH+GPX2 in different routing problems. LKH-3 [6] is a recent extension of LKH-2 that can be used in constrained TSP and other vehicle routing problems, e.g., multiple traveling repairman problem and vehicle routing problem with pickups and deliveries.

## References

1. Cook, W.: TSP test data (2009). <http://www.math.uwaterloo.ca/tsp/data/index.html>
2. Cook, W.: In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press, Princeton (2011)
3. Hains, D., Whitley, D., Howe, A.: Revisiting the big valley search space structure in the TSP. *J. Oper. Res. Soc.* **62**(2), 305–312 (2011)
4. Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* **126**(1), 106–130 (2000)
5. Helsgaun, K.: General k-opt submoves for the Lin-Kernighan TSP heuristic. *Math. Program. Comput.* **1**(2–3), 119–163 (2009)
6. Helsgaun, K.: An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Roskilde University, Technical report (2017)
7. Helsgaun, K.: LKH (2018). <http://www.akira.ruc.dk/~keld/research/LKH/>
8. Johnson, D., McGeoch, L., Glover, F., Rego, C.: 8th DIMACS implementation challenge: the traveling salesman problem (2013). <http://dimacs.rutgers.edu/Challenges/TSP/>
9. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling salesman problem. *Oper. Res.* **21**(2), 498–516 (1973)
10. Möbius, A., Freisleben, B., Merz, P., Schreiber, M.: Combinatorial optimization by iterative partial transcription. *Phys. Rev. E* **59**(4), 4667–4674 (1999)
11. Ochoa, G., Veerapen, N., Whitley, D., Burke, E.K.: The multi-funnel structure of TSP fitness landscapes: a visual exploration. In: Bonnefoy, S., Legrand, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) EA 2015. LNCS, vol. 9554, pp. 1–13. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-31471-6\\_1](https://doi.org/10.1007/978-3-319-31471-6_1)

12. Radcliffe, N., Surry, P.: Fitness variance of formae and performance predictions. In: Whitley, D., Vose, M. (eds.) *Foundations of Genetic Algorithms*, vol. 3, pp. 51–72. Morgan Kaufmann, Burlington (1995)
13. Tinós, R., Whitley, D., Chicano, F.: Partition crossover for pseudo-Boolean optimization. In: *Proceedings of FOGA XIII*, pp. 137–149 (2015)
14. Tinós, R., Whitley, D., Ochoa, G.: A new generalized partition crossover for the traveling salesman problem: tunneling between local optima. Submitted to *Evolutionary Computation* (2018)
15. Tinós, R., Zhao, L., Chicano, F., Whitley, D.: NK hybrid genetic algorithm for clustering. *IEEE Trans. Evol. Comput.*, 13 p. (2018). <https://doi.org/10.1109/TEVC.2018.2828643>
16. Veerapen, N., Ochoa, G., Tinós, R., Whitley, D.: Tunnelling crossover networks for the asymmetric TSP. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) *PPSN 2016. LNCS*, vol. 9921, pp. 994–1003. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_93](https://doi.org/10.1007/978-3-319-45823-6_93)
17. Whitley, D., Hains, D., Howe, A.: Tunneling between optima: partition crossover for the TSP. In: *Proceedings of GECCO 2009*, pp. 915–922 (2009)
18. Whitley, D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesmen: the genetic edge recombination operator. In: *Proceedings of ICGA 1989*, pp. 133–140 (1989)



# Escherization with a Distance Function Focusing on the Similarity of Local Structure

Yuichi Nagata<sup>(✉)</sup>

Tokushima University, 2-24, Shinkura-cho, Tokushima 770-8501, Japan  
nagata@is.tokushima-u.ac.jp

**Abstract.** The Escherization problem is that, given a goal figure, find a closed figure that is as close as possible to the goal figure and tiles the plane. In the Koizumi and Sugihara's formulation for the Escherization problem, the tile and goal shapes are represented as polygons whose similarity is evaluated by the Procrustes distance. In this paper, we incorporate a new distance function into their formulation, aiming at finding more satisfiable tile shapes. The proposed distance function successfully picks up tile shapes that are intuitively similar to the goal shape even when they are somewhat different from the goal shape in terms of the Procrustes distance. Due to the high computational cost for solving the formulated problem, we develop a tabu search algorithm to tackle this problem.

**Keywords:** Escher tiling · Tiling · Similarity measure

## 1 Introduction

A tiling refers to any pattern that covers the plane without any gaps or overlap. The Dutch artist M. C. Escher is famous for creating many artistic tilings, each of which consists of a few recognizable (especially one) figures such as animals. Such tiling is now called Escher tiling and it is a very intellectual task to design artistic Escher tilings while satisfying the constraints imposed to realize tiling.

As an attempt to automatically generate Escher tilings, Kaplan and Salesin [5] introduced the following optimization problem. Given a closed plane figure  $S$  (goal figure), find a closed figure  $T$  such that (i)  $T$  is as close as possible to  $S$ , and (ii) copies of  $T$  fit together to form a tiling of the plane. This problem is called the Escherization problem named after Escher. Koizumi and Sugihara [6] showed that when both tile and goal shapes are represented as polygons, the Escherization problem can be formulated as an eigenvalue problem.

Several enhancements to the Koizumi and Sugihara's formulation have been proposed. Imahori and Sakai [3] parameterized tile shapes (polygons) in a more flexible way, which creates a great deal of flexibility in the possible tile shapes (extended Koizumi and Sugihara's formulation). It requires, however, a considerable computational effort to solve the Escherization problem formulated with

this extension and they developed a local search algorithm for this problem. In the original Koizumi and Sugihara's formulation, the Procrustes distance [7] was introduced to measure the similarity between the tile and goal shapes. Imahori et al. [4], however, suggested that the Procrustes distance does not necessarily reflect an *intuitive* similarity between the two shapes. To handle this issue, they introduced weights to the Procrustes distance to emphasize the similarity with important parts of the goal figure. The idea of the weighted Procrustes distance, however, has not been incorporated into the extended Koizumi and Sugihara's formulation due to the heavy computational cost of calculating the weighted Procrustes distance.

In this paper, we propose another similarity measure (distance function), which captures the similarity of local structures between the tile and goal shapes, to successfully evaluate an intuitive similarity between them. We incorporate this similarity measure into the extended Koizumi and Sugihara's formulation and apply a tabu search algorithm [9] to the Escherization problem obtained.

## 2 Related Work

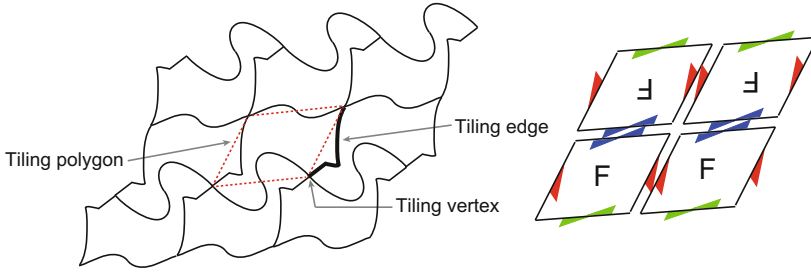
We first explain basic knowledge of tiling and then explain the Koizumi and Sugihara's formulation of the Escherization problem along with extended studies.

### 2.1 Isohedral Tilings

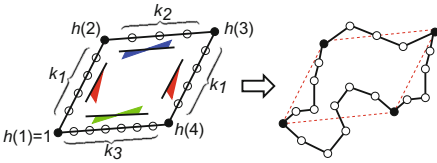
A *monohedral* tiling is one in which all the tiles are the same shape. If a monohedral tiling has a repeating structure, this tiling is called *isohedral*. There are 93 different types of isohedral tilings [1], which are individually referred to as IH1, IH2, ..., IH93. Figure 1 illustrates an example of an isohedral tiling belonging to IH47 with a few technical terms. A *tiling vertex* is a point where at least three tiles meet. A *tiling edge* is a boundary surface where exactly two tiles meet. A *tiling polygon* is the polygon formed by connecting the tiling vertices of a tile.

For each IH type, the nature of tile shapes can be represented by a template [8]. A template represents a tiling polygon from which all possible tile shapes are obtained by deforming the tiling edges and moving the tiling vertices under the constraints specified by the template. For example, Fig. 1 illustrates a template of IH47; this template means that the tiling polygon is a quadrilateral consisting of two opposite J edges that are parallel to one another and two independent S edges. There are four types of tiling edges (types J, S, U, and I) and it is convenient to express these types with colored arrowheads as illustrated in Fig. 1 (only types J and S are shown). These types are closely related to how the tiles are fitted to each other, and a template also gives information about the adjacency relationship between the tiles.

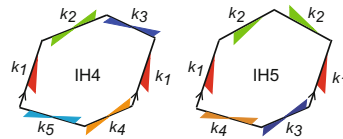
According to the adjacency relationship, four types of tiling edges can be deformed in the following ways (see also Fig. 1). A type J edge can be deformed in any arbitrary fashion, but the corresponding J edge must also be deformed into the same shape. A type S edge must be symmetric with respect to the



**Fig. 1.** Example of an isohedral tiling (left), and the template of IH47 (right) where J and S edges are indicated by single arrowheads and facing arrowheads, respectively.



**Fig. 2.** Template of IH47 for a specific assignment of the points to the tiling edges (left), and an example of a tile shape (right).



**Fig. 3.** Templates of IH4 and IH5. Two opposite J edges marked with  $\triangle$  are parallel to one another.

midpoint. A type U edge must be symmetric with respect to a line through the midpoint and orthogonal to it. A type I edge must be a straight line.

### 2.2 Koizumi and Sugiharas’s Formulation and Its Extension

Koizumi and Sugihara [6] modeled the tile shape as a polygon of  $n$  points. In this case, the template of IH47 is represented as shown in Fig. 2, where exactly one point must be placed at each of the tiling vertices (black circles) and the remaining points are placed on the tiling edges (white circles). This template represents possible arrangements of the  $n$  points; the  $n$  points can be moved as illustrated in Fig. 2. Koizumi and Sugihara originally placed the same number of points on every tiling edge. After that, Imahori et al. [3] extended this model to assign different numbers of points on the tiling edges (extended Koizumi and Sugihara’s formulation). We denote the numbers of points placed on the tiling edges as  $k_1, k_2, \dots$  as illustrated in Fig. 2.

Let the  $n$  points on the template be indexed clockwise by  $1, 2, \dots, n$ , starting from one of the tiling vertices. We represent the tile shape as a  $2n$ -dimensional vector  $\mathbf{u} = (x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n)^T$ , where  $(x_i, y_i)^T$  is the coordinates of the  $i$ th point in the  $xy$ -plane. We will also denote  $(x_i, y_i)^T$  as  $\hat{\mathbf{u}}_i$ . We refer to the tile shape (polygon) specified by  $\mathbf{u}$  as  $U$ . The values of vector  $\mathbf{u}$  are constrained so that the tile shape  $U$  is consistent with the template selected. For example, if we select IH47, the values of vector  $\mathbf{u}$  must satisfy the following equation:

$$\begin{cases} \hat{\mathbf{u}}_{h(1)+i} - \hat{\mathbf{u}}_{h(1)} = \hat{\mathbf{u}}_{h(4)-i} - \hat{\mathbf{u}}_{h(4)} & (i = 1, \dots, k_1 + 1) \\ \hat{\mathbf{u}}_{h(2)+i} - \hat{\mathbf{u}}_{h(2)} = -(\hat{\mathbf{u}}_{h(3)-i} - \hat{\mathbf{u}}_{h(3)}) & (i = 1, \dots, \lfloor \frac{k_2+1}{2} \rfloor) \\ \hat{\mathbf{u}}_{h(4)+i} - \hat{\mathbf{u}}_{h(4)} = -(\hat{\mathbf{u}}_{n+1-i} - \hat{\mathbf{u}}_{h(1)}) & (i = 1, \dots, \lfloor \frac{k_3+1}{2} \rfloor) \end{cases}, \quad (1)$$

where  $h(s)$  ( $s = 1, \dots, 4$ ) is the index of the  $s$ th tiling vertex as shown in Fig. 2.

Equation (1) is a homogeneous system of linear equations and is represented by

$$A\mathbf{u} = \mathbf{0}, \quad (2)$$

where  $A$  is a  $m' \times 2n$  matrix ( $m' < 2n$ ). Let  $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m$  be the orthonormal basis of  $\text{Ker}(A)$ . A general solution of Eq. (2) is then given by

$$\mathbf{u} = \xi_1 \mathbf{b}_1 + \xi_2 \mathbf{b}_2 + \dots + \xi_m \mathbf{b}_m = B\xi, \quad (3)$$

where  $B = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m)$  is a  $2n \times m$  matrix and  $\xi = (\xi_1, \xi_2, \dots, \xi_m)^\top$  is a parameter vector. In fact, tile shapes for every isohedral tilings can be parameterized in the form of Eq. (3), where the matrix  $B$  depends on the assignment of the  $n$  points to the tiling edges as well as isohedral type.

In the Koizumi and Sugihara's formulation, the goal figure is also represented as a polygon of  $n$  points and their coordinates are represented by a  $2n$ -dimensional vector  $\mathbf{w} = (x_1^w, x_2^w, \dots, x_n^w, y_1^w, y_2^w, \dots, y_n^w)^\top$ , where  $(x_i^w, y_i^w)^\top$  is the coordinates of the  $i$ th point of the goal polygon. We will also denote  $(x_i^w, y_i^w)^\top$  as  $\hat{\mathbf{w}}_i$ . We refer to the goal shape (polygon) specified by  $\mathbf{w}$  as  $W$ . To measure the similarity between the two polygons  $U$  and  $W$ , they employed the Procrustes distance [7]. Let us first, however, explain a more simple but essentially the same distance measure for the ease of understanding. We refer to this distance measure as the normal distance in this paper. The square of the normal distance between the two polygons  $U$  and  $W$  is defined by

$$d^2(U, W) = \|\mathbf{u} - \mathbf{w}\|^2 = \sum_{i=1}^n \|\hat{\mathbf{u}}_i - \hat{\mathbf{w}}_i\|^2, \quad (4)$$

where  $\|\cdot\|$  is the Euclidean norm.

When the normal distance is used, from Eqs. (3) and (4), the Escherization problem can be formulated as the following unconstrained optimization problem:

$$\text{minimize: } \|B\xi - \mathbf{w}\|^2. \quad (5)$$

This is a least-squares problem and the solution is given by  $\xi^* = (B^\top B)^{-1} B^\top \mathbf{w} = B^\top \mathbf{w}$  with the minimum value  $-\xi^{*\top} \xi^* + \mathbf{w}^\top \mathbf{w}$ . The optimal tile shape  $\mathbf{u}^*$  is then obtained by  $\mathbf{u}^* = B\xi^*$ .

When calculating the normal distance between the two polygons, we need to consider the  $n$  different numbering for the goal polygon  $W$  by shifting the first point for the numbering. Therefore, we define  $\mathbf{w}_j$  ( $j = 1, 2, \dots, n$ ) in the same way as  $\mathbf{w}$  by renumbering the index of the  $n$  points such that the  $j$ th point (in the original index) becomes the first point.

Let  $I$  be a set of the indices for the isohedral types and  $K_i$  a set of all possible configurations for the assignment of the  $n$  points to the tiling edges for an

isohedral type  $i$ . For example,  $K_{47} = \{(k_1, k_2) \mid 0 \leq k_1, 0 \leq k_2, 2k_1 + k_2 \leq n - 4\}$  whereas  $k_3$  is determined by  $k_3 = n - 4 - (2k_1 + k_2)$  (see Fig. 2). Because the matrix  $B$  depends on  $i \in I$  and  $k \in K_i$ , we denote it as  $B_{ik}$ . Let  $J = \{1, 2, \dots, n\}$  be a set of the indices of the first point for the  $n$  different numbering of the goal polygon. If we try to perform the exhaustive search, we need to compute

$$\min_{\xi \in R^m} \|B_{ik}\xi - \mathbf{w}_j\|^2 = -\xi_{ikj}^{*\top} \xi_{ikj}^* + \mathbf{w}^\top \mathbf{w}, \quad (6)$$

for all combinations of  $i \in I$ ,  $k \in K_i$ , and  $j \in J$ , where  $\xi_{ikj}^* = B_{ik}^\top \mathbf{w}_j$ .

For each  $i \in I$  and  $k \in K_i$ , it takes  $O(n^3)$  time to compute Eq. (6) for all values of  $j \in J$  because it takes  $O(n^3)$  time for computing  $B_{ik}$  and  $O(n^2)$  time for computing  $B_{ik}^\top \mathbf{w}_j$  (for each value of  $j$ ). However, the order of  $K_i$  reaches  $O(n^3)$  for IH5 and IH6 and  $O(n^4)$  for IH4, and it requires a considerable computational time to perform the exhaustive search. Figure 3 shows the templates of IH4 and IH5. To alleviate this problem, Imahori and Sakai [4] proposed a local search algorithm to search for only promising configurations in  $K_i$ , which has succeeded in finding better tile shapes than the original Koizumi and Sugihara's method.

Finally, we mention the difference between the Procrustes distance and the normal distance. For some isohedral types including IH47 and IH4, exactly the same result is obtained with either distance measure. For some isohedral types, however, the Procrustes distance must be used because the templates can only parameterize tile shapes facing in a specific direction. For example, the two adjacent J edges in the template of IH5 (see Fig. 3) are parameterized such that they make equal and opposite angles with the y-axis. The Procrustes distance calculates the normal distance after rotating  $U$  so that the normal distance between  $U$  and  $W$  is minimized. When the Procrustes distance is used, the Escherization problem can be reduced to an eigenvalue problem, which can be solved in  $O(n^2)$  time [4]. We use only the normal distance to explain the original Koizumi and Sugihara's formulation [6], the subsequent studies [2–4], and the proposed method for the ease of understanding and due to space limitations.

### 2.3 The Weighted Normal Distance

The normal distance Eq. (4) seems to be the most natural similarity measure between two polygons. Imahori et al. [4], however, suggested that the normal distance does not necessarily reflect intuitive similarity between two polygons. The main cause is that in many cases goal figures have important parts that characterize their shapes and they assigned *weights* to the points on the important parts of the goal polygon to emphasize the similarity with these parts.

Let  $k_i$  ( $i = 1, 2, \dots, n$ ) be a positive weight assigned to the  $i$ th point of the goal polygon  $W$ . The weighted normal distance is then defined by

$$d_w^2(U, W) = \sum_{i=1}^n k_i \|\hat{\mathbf{u}}_i - \hat{\mathbf{w}}_i\|^2 = \mathbf{u}^\top G \mathbf{u} - 2\mathbf{w}^\top G \mathbf{u} + \mathbf{w}^\top G \mathbf{w}, \quad (7)$$

where  $G$  is the  $2n \times 2n$  diagonal matrix whose diagonal elements are  $k_1, k_2, \dots, k_n, k_1, k_2, \dots, k_n$ . When the weighted normal distance is used, from Eqs. (3) and (7), the Escherization problem is formulated as follows:

$$\text{minimize: } \boldsymbol{\xi}^\top B^\top G B \boldsymbol{\xi} - 2\mathbf{w}^\top G B \boldsymbol{\xi} + \mathbf{w}^\top G \mathbf{w}. \tag{8}$$

### 3 Proposed Method

We propose a new distance function to evaluate intuitive similarity between two polygons and incorporate it into the extended Koizumi and Sugihara’s formulation. We try to solve the formulated problem using tabu search (TS) [9] to search for as many configurations  $k \in K_i$  as possible for each isohedral type  $i \in I$ .

#### 3.1 The Proposed Similarity Measure

As expressed by Eqs. (4) and (7), in order to shorten the (weighted) normal distance, the points of the tile polygon  $U$  must be close to the corresponding points of the goal polygon  $W$ . In contrast, we focus on the similarity of the relative positional relationship of adjacent points between two polygons. The proposed distance function is defined as follows:

$$d_a^2(U, W) = \sum_{i=1}^n k_i \|(\hat{\mathbf{u}}_{i+1} - \hat{\mathbf{u}}_i) - (\hat{\mathbf{w}}_{i+1} - \hat{\mathbf{w}}_i)\|^2, \tag{9}$$

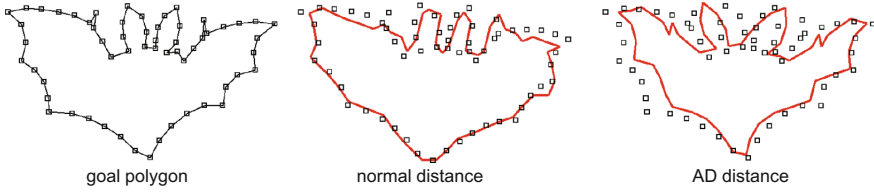
where  $n + 1$  represents 1 and  $k_i$  is the weight. We refer to the proposed distance as the (weighted) *adjacent difference (AD) distance*.

In the right side of Fig. 4, we can see a typical example of a tile polygon (red line) that is determined to be very similar to the goal polygon “bat” under the AD distance but is not so under the normal distance. The middle of the figure shows the opposite situation. Compared to the tile shape in the middle of the figure, the tile shape in the right side does not so much overlap with the goal polygon, but it seems to be intuitively more similar to the goal polygon than the former one. The reason is that local shapes of the contours of the wings and ears are well preserved in the right side figure even though overall shape is distorted (e.g., the vertical width of the wings is getting narrower). As exemplified in this example, even if the global structure is somewhat distorted, it would be better to actively preserve local structures of the goal shape to search for more satisfiable tile shapes. The AD distance is designed assuming such a situation.

It is also possible to assign weights to edges of the goal polygon  $W$  and  $k_i$  in Eq. (9) is the weight assigned to the edge between  $i$ th and  $(i + 1)$ th points. In fact, Eq. (9) can be expressed by the same matrix representation as the right side of Eq. (7), where  $G$  is a  $2n \times 2n$  symmetric tridiagonal matrix whose non-zero elements are defined as follows:

$$\begin{cases} g_{i,i} = g_{i+n,i+n} = k_i + k_{i+1} \\ g_{i,i+1} = g_{i+n,i+1+n} = -k_i \\ g_{i+1,i} = g_{i+1+n,i+n} = -k_i \end{cases}, \tag{10}$$





**Fig. 4.** Goal polygon “bat” and tile shapes that are very close to the goal polygon under the normal distance and the AD distance, respectively.

where  $2n + 1$  means 1. We should note that the matrix  $G$  depends on the first point for the numbering of the goal polygon  $j \in J$  (this also applies to the case where the weighted normal distance is used) because the indices of the weighted edges must be also shifted depending on the numbering. Therefore, we define  $G_j$  ( $j = 1, 2, \dots, n$ ) in the same way as Eq. (10), by renumbering the index such that the  $j$ th point (in the original index) becomes the first point.

### 3.2 The Extended Koizumi and Sugihara’s Formulation with the AD Distance

The weighted normal distance Eq. (7) and the AD distance Eq. (9) have the same matrix representation and the Escherization problem using the AD distance is also formulated by Eq. (8). When the AD distance is incorporated into the extended Koizumi and Sugihara’s formulation, we need to solve the following optimization problem:

$$\text{minimize: } \boldsymbol{\xi}^\top B_{ik}^\top G_j B_{ik} \boldsymbol{\xi} - 2\boldsymbol{w}_j^\top G_j B_{ik} \boldsymbol{\xi} + \boldsymbol{w}_j^\top G_j \boldsymbol{w}_j, \tag{11}$$

for all combinations of  $i \in I$ ,  $k \in K_i$ , and  $j \in J$ .

Let us consider Eq. (8) again instead of Eq. (11) for simplicity (indices  $i, k$ , and  $j$  are omitted). The solution  $\boldsymbol{\xi}^*$  to Eq. (8) is obtained by solving the equation  $B^\top G B \boldsymbol{\xi} = B^\top G \boldsymbol{w}$  (the minimum is  $-\boldsymbol{\xi}^{*\top} \boldsymbol{\xi}^* + \boldsymbol{w}^\top \boldsymbol{w}$ ). However, as explained later, the matrix  $B^\top G B$  is rank deficient (when the AD distance is used) and we find the solution in the following way, which is essentially the same as in [4]. First, a set of column vectors  $\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_m$  (see Eq. (3)) are linearly transformed into  $\boldsymbol{b}'_1, \boldsymbol{b}'_2, \dots, \boldsymbol{b}'_{m'}$ , such that  $\boldsymbol{b}'_i{}^\top G \boldsymbol{b}'_j = \delta_{ij}$  (the Kronecker delta function) for  $i, j \in \{1, 2, \dots, m'\}$  (as explained later,  $m' = m - 2$ ). Such a set of column vectors can be obtained in  $O(n^3)$  time by using the Gram-Schmidt orthogonalization process with an inner product defined as  $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\top G \boldsymbol{y}$ . Let a matrix  $B'$  be defined as  $B' = (\boldsymbol{b}'_1, \boldsymbol{b}'_2, \dots, \boldsymbol{b}'_{m'})$  and the tile shape  $U$  be parameterized by  $\boldsymbol{u} = B' \boldsymbol{\xi}$ . Because  $B'^\top G B'$  becomes an identity matrix, the solution to Eq. (8) ( $B$  is replaced with  $B'$  in this case) is obtained by  $\boldsymbol{\xi}^* = B'^\top G \boldsymbol{w}$ . Note that when the AD distance is used,  $m' = m - 2$  because the matrix  $G$  is rank deficient by 2. Therefore, the degree of freedom for parameterizing tile shapes is also reduced. Intuitively, this is because the value of the AD distance does not depends on the position of the center of gravity of  $U$  (it does not determined uniquely).

The matrix  $B'$  depends on  $i$  ( $\in I$ ) and  $k$  ( $\in K_i$ ). In addition, unlike in the case of the matrix  $B$ ,  $B'$  depends on  $j$  ( $\in J$ ). Therefore, we denote the matrix  $B'$  as  $B'_{ikj}$  when specifying the indices  $i$ ,  $k$ , and  $j$ . For each  $i \in I$  and  $k \in K_i$ , it takes  $O(n^4)$  time for solving the optimization problem Eq. (11) (i.e., computing  $\xi_{ikj}^* = B'_{ikj}{}^\top G_j \mathbf{w}_j$ ) for all values of  $j$  because it takes  $O(n^3)$  time for computing  $B'_{ikj}$ . Note that if the weight is not introduced,  $B'_{ikj}$  does not depend on  $j$  and it takes  $O(n^3)$  time in the same situation. Remember that when the normal distance is used, this computational cost is  $O(n^3)$  (see Sect. 2.2). Therefore, it is computationally more difficult to search for many configurations  $k \in K_i$  for each isohedral type  $i \in I$ , compared to the case of the normal distance. This also applies to the case where the weighted normal distance is used, and therefore the weighted normal distance was not incorporated into the extended Koizumi and Sugihara's formulation.

### 3.3 A Tabu Search Algorithm

It requires a considerable computation time to solve the optimization problem Eq. (11) for all possible combinations of  $i \in I$ ,  $k \in K_i$ , and  $j \in J$  and we propose a TS algorithm to search for only promising configurations among them. The basic idea is similar to the local search algorithm [4] developed for the extended Koizumi and Sugihara's formulation with the normal distance (Eq. (6)), but we propose a TS algorithm here to enhance the performance. Compared to the case of the normal distance, however, the required computational cost is significantly increased and we need to reduce the computational cost.

The TS algorithm is performed for each isohedral type  $i \in I$ . In the TS algorithm, the solution candidate represents a configuration  $(k, j)$  and its objective value is given by  $-\xi_{ikj}^*{}^\top \xi_{ikj}^* + \mathbf{w}^\top \mathbf{w}$  (the minimum of Eq. (11)). We define the neighborhood as a set of configurations  $(k', j')$  given by the combinations of  $j' \in \{j, j \pm 1\}$  and  $k'$  that is obtained by incrementing (or decrementing) the number of points assigned to two tiling edges in all possible ways. For example, if we select IH47 (see Fig. 2), for the current  $k = (k_1, k_2)$ , possible values of  $k'$  are  $(k_1 \pm 1, k_2 \mp 2, k_3)$ ,  $(k_1, k_2 \pm 1, k_3 \mp 1)$ ,  $(k_1 \pm 1, k_2, k_3 \mp 2)$  (actually  $k_3$  is omitted because  $k_3$  is obtained as  $n - 4 - 2k_1 - k_2$ ).

Algorithm 1 depicts the TS algorithm. We denote the current solution  $(k, j)$  and its neighborhood as  $x$  and  $\mathcal{N}(x)$ , respectively. Before starting the iterations, the current solution  $x$  and the current best solution  $x_{best}$  are initialized with a randomly generated solution (line 1). At each iteration, the best non-tabu solution  $x'$  (we define tabu solutions later) is selected from the neighborhood  $\mathcal{N}(x)$  (line 3). 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 as an exception. The current solution  $x$  and current best solution  $x_{best}$  (if necessary) are then updated by  $x'$  (line 4). Iterations are repeated until the number of iterations reaches a given maximum number  $iterMax$  (lines 2 and 5). Finally, the current best solution  $x_{best}$  is returned (line 7).

We explain how to define tabu solutions with an example where  $k$  is represented as  $(k_1, k_2, k_3, k_4)$ . Let the current solution be denoted as  $(k_1, k_2, k_3, k_4; j)$ .

If the current solution is replaced with the selected solution  $(k'_1, k_2, k'_3, k_4; j')$ , two pairs of  $(k'_1, j')$  and  $(k'_3, j')$  are stored in the tabu list during the subsequent  $T$  iterations. At each iteration, when a neighbor solution  $(k_1, k'_2, k'_3, k_4; j')$  is obtained from the current solution  $(k_1, k_2, k_3, k_4; j)$ , this solution is regarded as a tabu-solution if both  $(k'_2, j')$  and  $(k'_3, j')$  exists in the tabu list.

---

**Algorithm 1.** TABU-SEARCH(isohedral type  $i$ )
 

---

```

1: Set the solution  $x$  randomly, set  $x_{best} := x$ , and  $iter\_no := 0$ ;
2: while ( $iter\_no \leq iterMax$ ) do
3:   Select a best non-tabu solution  $x' \in \mathcal{N}(x)$  (aspiration criterion is considered);
4:   Update  $x := x'$  and  $x_{best} := x'$  (if  $x'$  is better than  $x_{best}$ );
5:   Set  $iter\_no := iter\_no + 1$ ;
6: end while
7: return  $x_{best}$ ;

```

---

We mention the main difference between the proposed TS and the local search used in [4]. In [4], the solution candidate was represented by only  $k$  and it was evaluated by testing the  $n$  different values for  $j$ . Its computational effort is  $O(n^3)$  when using the normal distance. On the other hand, however, it takes  $O(n^4)$  time in the same situation when the weighted AD distance is used because it takes  $O(n^3)$  time for each value of  $j$ . In our observation, the optimal value of  $j$  for the value of  $k$  tends to continuously change with the change of  $k$ . Therefore, it is reasonable to restrict the search range of  $j$  around the current value of  $j$  in the neighborhood as in the proposed TS algorithm. Therefore, we have decided to include the variable  $j$  in the solution  $x$ .

Although there are 93 isohedral types, it is enough to consider only 10 isohedral types (IH1, IH2, IH3, IH4, IH5, IH6, IH7, IH8, IH21, IH28) for the optimization because the remaining 83 types are approximately obtained by assigning no point to tiling edges in the 10 isohedral types. In our observation, good tile shapes are mostly obtained with IH4, IH5, and IH6 because other isohedral types do not have enough flexibility to represent tile shapes (e.g., most tiling edges have the same shape). We therefore consider only the three isohedral types IH4, IH5, IH6 for the optimization. In fact, the order of  $K_i$  is equal to or greater than  $O(n^3)$  only for these isohedral types.

Through preliminary experiments, the parameters of the TS algorithm were determined as follows:  $T = 50$  and  $iterMax = 100$ , where we set the value of  $iterMax$  to a small value since it was better to repeat the TS algorithm from different initial solutions rather than to continue one trial for a long time. We define one set of trials as 60 runs of the TS algorithm during which the top 20 tile shapes (including non-local minima) found are stored.

## 4 Experimental Results

We implemented the proposed TS algorithm in C++ and executed the program code on a Ubuntu 14.04 Linux PC with Intel Core i7-4790@3.60 GHz CPU.

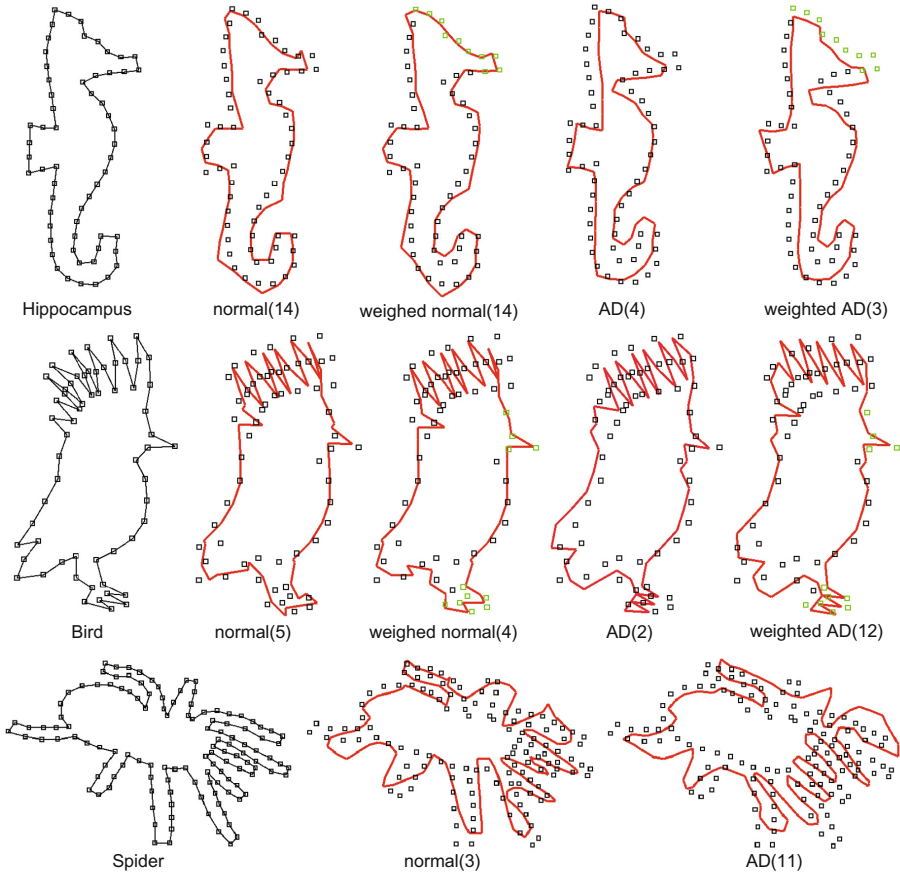
We applied one set of trials of the TS algorithm to the three goal figures hippocampus ( $n = 59$ ), bird ( $n = 60$ ), and spider ( $n = 126$ ) using each of the four distance functions (normal distance, weighted normal distance, AD distance, and weighted AD distance). The execution time of one set of TS algorithm were about 50s (normal and AD distances) and 140s (weighted normal and weighted AD distances) for the goal polygon bird and about 400s (normal and AD distances) for the goal polygon spider.

Figure 5 shows the three goal polygons followed by the intuitively best tile shapes obtained with the four distance functions. Note that the top tile shape in terms of the distance value do not necessarily the best one form an intuitive point of view, and we selected the intuitively best one among the top 20 tile shapes for each distance function, where the numbers in parentheses indicate the ranking in terms of distance values. The tile shapes are drawn with red lines on the points of the goal polygons (black points). When the (weighted) AD distance is used, the position of the center of gravity of the tile shape is not determined (see Sect. 3.2) and we put the tile such that the normal distance is minimized. When the weighted normal distance and the weighted AD distance are used, the weighted values were all set to four and the weighted points or the both ends of the weighted edges are drawn in green. In addition, Fig. 6 presents three tilings generated from the tile shapes shown on the right side of Fig. 5.

We first discuss the results for hippocampus. From the definition, when the normal distance is used, the resulting tile shape seems to be most overlapped with the goal polygon, but differences in some local structures are conspicuous. By using the weighted normal distance, the difference in the weighted part (head) is getting smaller. When using the AD distance, although the obtained tile shape does not so much overlap the goal polygon (compared to the normal distance case), local structures of the goal polygon are well maintained. The obtained tile shape seems to be intuitively quite similar to the goal polygon, except for the problem that the width of the head part is shortened and the neck is too thin. By introducing weights to the AD distance, the local shape of the head part is getting similar to that of the goal polygon and the aforementioned problem in the AD distance is somewhat improved.

Next, we discuss the results for bird. The tile shape obtained with the normal distance seems to be most overlapped with the goal polygon, but the difference in the foot part is conspicuous. Therefore, we assigned weights to the foot part as well as the beak part for the weighted normal distance. However, no particular improvement is found when the weighted normal distance is used. In contrast to the (weighted) normal distance, when using the AD distance, not only the overall structure but also the local structures (especially the foot part) are well maintained. By introducing weights to the AD distance, only the local shape of the beak parts is slightly improved.

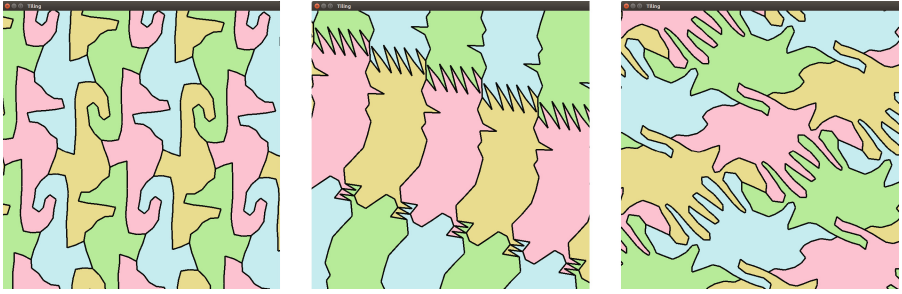
Next, we discuss the results for spider, which is a pretty challenging goal figure. Since it was difficult to determine appropriate weight points and edges, only the normal and AD distances were tested. We can see that the tile shape obtained with the AD distance seems to be intuitively more similar to the goal



**Fig. 5.** The goal polygons and tile shapes obtained with the four distance functions. (Color figure online)

figure than that with the normal distance. The main reason is that the shape of each leg is well preserved, although the positions of the bases of the legs are different from those of the goal figure.

We also mention the problem of the AD distance. Compared to the normal distance, tile shapes obtained with the AD distance are rich in variety and many undesirable tile shapes are also included in the top 20 tile shapes. The reason for this is that the value of the AD distance may be small even if overall tile shape is fairly distorted from the goal shape. In the present situation, a satisfiable tile shape is obtained when the global structure happens to be similar to that of the goal figure to some extent. In such a case, we can find a very satisfiable tile shape which cannot be obtained with the (weighted) normal distance.



**Fig. 6.** Tilings generated from the tile shapes on the right side of Fig. 5. (Color figure online)

## 5 Conclusion

We have proposed a new distance function (AD distance), which captures the similarity of local structures between the two shapes. When the AD distance is incorporated into the (extended) Koizumi and Sugihara’s formulation of the Escherization problem, tile shapes obtained actively preserve local structures of the goal shape even if the global structure is sacrificed. Experimental results showed that it is better to positively preserve local structures of the goal shape by allowing the global structure to deform (if the degree of deformation is not very large), in order to obtain intuitively satisfiable tile shapes. Due to the high computational cost of the exhaustive search for the formulated Escherization problem, we developed a TS algorithm for solving this problem, which made it possible to obtain satisfiable tile shapes in a reasonable computational time.

**Acknowledgements.** This work was supported by JSPS KAKENHI Grant Number 17K00342.

## References

1. Grünbaum, B., Shephard, G.: *Tilings and Patterns*. A Series of Books in the Mathematical Sciences (1989)
2. Imahori, S., Sakai, S.: A local-search based algorithm for the Escherization problem. In: *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 151–155 (2012)
3. Imahori, S., Sakai, S.: A local-search based algorithm for the Escher-like tiling problem. *IPSJ SIG Technical reports*, vol. 2013-AL-144, no.14 (2013 in Japanese)
4. Imahori, S., Kawade, S., Yamakata, Y.: Escher-like tilings with weights. In: Akiyama, J., Ito, H., Sakai, T. (eds.) *JCDCGG 2015*. LNCS, vol. 9943, pp. 132–142. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-48532-4\\_12](https://doi.org/10.1007/978-3-319-48532-4_12)
5. Kaplan, C.S., Salesin, D.H.: Escherization. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 499–510 (2000)
6. Koizumi, H., Sugihara, K.: Maximum eigenvalue problem for Escherization. *Graph. Comb.* **27**(3), 431–439 (2011)

7. Werman, M., Weinshall, D.: Similarity and affine invariant distances between 2D point sets. *IEEE Trans. Pattern Anal. Mach. Intell.* **17**(8), 810–814 (1995)
8. Kaplan, C.S.: *Introductory Tiling Theory for Computer Graphics*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, San Rafael (2009)
9. Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic Publishers, Dordrecht (1997)



# Evolutionary Search of Binary Orthogonal Arrays

Luca Mariot<sup>1</sup>(✉), Stjepan Picek<sup>2</sup>, Domagoj Jakobovic<sup>3</sup>, and Alberto Leporati<sup>1</sup>

<sup>1</sup> DISCo, Università degli Studi di Milano-Bicocca,  
Viale Sarca 336/14, 20126 Milano, Italy  
{luca.mariot,alberto.leporati}@unimib.it

<sup>2</sup> Cyber Security Research Group, Delft University of Technology,  
Mekelweg 2, Delft, The Netherlands  
S.Picek@tudelft.nl

<sup>3</sup> Faculty of Electrical Engineering and Computing,  
University of Zagreb, Unska 3, Zagreb, Croatia  
domagoj.jakobovic@fer.hr

**Abstract.** Orthogonal Arrays (OA) represent an interesting breed of combinatorial designs that finds applications in several domains such as statistics, coding theory, and cryptography. In this work, we address the problem of constructing binary OA through evolutionary algorithms, an approach which received little attention in the combinatorial designs literature. We focus on the representation of a feasible solution, which we encode as a set of Boolean functions whose truth tables are used as the columns of a binary matrix, and on the design of an appropriate fitness function and variation operators for this problem. We finally present experimental results obtained with genetic algorithms (GA) and genetic programming (GP) on optimizing such fitness function, and compare the performances of these two metaheuristics with respect to the size of the considered problem instances. The experimental results show that GP outperforms GA at handling this type of problem, as it converges to an optimal solution in all considered problem instances but one.

**Keywords:** Orthogonal arrays · Genetic algorithms  
Genetic programming · Boolean functions

## 1 Introduction

The field of *combinatorial designs* provides an interesting source of problems for heuristic optimization techniques. Depending on the size of the support set and the particular nature of the balancedness constraints, the two main research questions addressed in combinatorial design theory are the following:

1. *Existence:* Does a design with a particular set of parameters (i.e., support set, balancedness constraints) exist?



2. *Construction*: Once the existence question for a specified kind of design is positively answered, is there an efficient method to generate its instances?

Since the existence question of a design can always be cast as a combinatorial optimization problem [2], it follows that the use of heuristic techniques can contribute to the above research questions in a twofold way: first, by providing concrete examples of designs with specific parameters, hence answering the existence question in positive; second, once the existence question has been settled, by providing another method for efficiently constructing designs.

Despite this, the amount of literature devoted to the use of heuristic optimization techniques for constructing combinatorial designs is rather limited (see Chapter 6 of [2] for a concise survey). This is especially true for the case of *orthogonal arrays* (OA), which represent one of the most interesting breeds of combinatorial designs, due to their numerous applications in other research domains such as the *design of experiments*, *error-correcting codes* and *cryptography* [11]. Indeed, one can find only the papers by Safadi et al. [9] and Wang et al. [12] that deal with the construction of *mixed-level orthogonal arrays* (MOA), respectively through *genetic algorithms* (GA) and *simulated annealing* (SA). Nonetheless, MOA represent a very specific kind of OA, and to the best of our knowledge there are no works in the literature addressing the heuristic design of classic OA through evolutionary algorithms.

The aim of this paper is to begin filling this gap by considering the construction of orthogonal arrays through *evolutionary algorithms* (EAs), in particular *genetic algorithms* and *genetic programming* (GP). Beside its potential impact in other domains mentioned above, this research is also interesting from the evolutionary computing point of view. As a matter of fact, evolving OA through evolutionary heuristics requires to define suitable encodings and variation operators, which could find applications also in other optimization problems. Additionally, depending on the difficulty of converging to an optimal solution, designing OA could also represent an interesting benchmark problem for new evolutionary algorithms and optimization heuristics, as well as for more established ones.

Since the present work is the first one in this line of research, we consider in particular the modeling aspects of the optimization problem, focusing on the encodings for the feasible solutions and the design of variation operators to evolve them. For this reason, we begin by tackling the case of *binary* orthogonal arrays, since this allows us to represent the candidate solutions of our problem as sets of *Boolean functions*. More specifically, we take the *truth tables* of such Boolean functions as the columns of a binary matrix, which actually corresponds to the phenotype of a candidate solution. On the other hand, the genotype is either a set of binary strings for GA or a set of Boolean trees for GP.

In order to evaluate the candidate solutions evolved by GA and GP, we design a fitness function based on the *Minkowski distance* that measures the deviation of a binary matrix from being an orthogonal array having specified parameters, with the goal of minimizing it. In the case of GA, we also exploit a basic property of orthogonal arrays to design ad-hoc crossover and mutation operators, which ensure that the Boolean functions composing an individual are balanced, thus

reducing the resulting search space. For GP, we incorporate this property as an additional penalty factor in the fitness function, since there is no straightforward way to design GP variation operators that enforce the balancedness constraint at the tree level.

We compute the size of the search spaces respectively explored by GA and GP in terms of the number of variables of the Boolean functions and the columns of the binary matrices involved, showing that the resulting search spaces cannot be exhaustively enumerated already for Boolean functions of  $n = 4$  variables and  $k = 8$  columns.

The experimental results show that GP largely outperforms GA at evolving binary OA, even though the latter actually explores a smaller search space. As a matter of fact, GA is able to find orthogonal arrays defined by up to 8 Boolean functions of 4 variables, while GP arrives one step further by obtaining also orthogonal arrays composed of 16 functions of 5 variables. This performance difference is analogous to the findings reported in [5], where the authors observed that GP outperforms GA in the generation of cellular automata defining *orthogonal Latin squares*, which are a type of combinatorial designs closely connected with orthogonal arrays. Consequently, the present work brings additional empirical evidence that GP is a better metaheuristic at handling optimization problems related to combinatorial designs.

## 2 Basic Definitions

We begin by giving the basic definition of orthogonal arrays, following the notation used by Hedayat et al. [3]:

**Definition 1.** *Let  $S$  be a finite set of  $s$  symbols (called the support set) and let  $N, k, t, \lambda \in \mathbb{N}$  with  $0 \leq t \leq k$ . An  $N \times k$  matrix  $A$  with entries from  $S$  is an orthogonal array with  $s$  levels,  $k$  columns, strength  $t$ , and index  $\lambda$  (for short, an  $OA(N, k, s, t)$ ) if in each submatrix of  $N$  rows and  $t$  columns each  $t$ -uple over  $S$  occurs exactly  $\lambda$  times.*

Clearly, if  $A$  is an  $OA(N, k, s, t)$ , then it follows that  $\lambda = N/s^t$ .

This is the reason why the parameter  $\lambda$  is usually omitted from the specification of an OA.

A basic property of orthogonal arrays of strength  $t$  is that they satisfy the balancedness constraint also for smaller strengths, as shown in [3]:

**Theorem 1.** *Let  $A$  be an  $OA(N, k, s, t)$  with  $\lambda = N/s^t$ . Then,  $A$  is also an  $OA(N, k, s, t - i)$  with  $\lambda = N/s^{t-i}$  for all  $1 \leq i < t$ .*

An OA without repeated rows is called *simple*. If  $S = \{0, 1\}$  (i.e., the symbol set is the Boolean alphabet), then the OA is called *binary*.

Simple binary OA have an important application in defining the support of *correlation-immune Boolean functions*, which play an important role in the design of countermeasures for *side-channel attacks* [1].

Finally, we give a basic definition of Boolean functions and their truth tables:

**Definition 2.** A Boolean function of  $n$  variables is a mapping  $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ . Assuming that the vector of  $\mathbb{F}_2^n$  are lexicographically ordered, the truth table associated to  $f$  is the  $2^n$ -bit vector  $\Omega(f)$  defined as follows:

$$\Omega(f) = (f(0, 0, \dots, 0), f(0, 0, \dots, 1), \dots, f(1, 1, \dots, 1)). \quad (1)$$

In particular, a Boolean function  $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  is called *balanced* if the number of zeros in its truth table (and thus also the number of ones) equals  $2^{n-1}$ .

We can now formulate the combinatorial optimization problem which we will investigate in the rest of this work. We represent the columns of binary orthogonal arrays with the truth tables of a set of Boolean functions. This can be formally stated as follows:

*Problem 1.* Let  $n, k, t \in \mathbb{N}$ . Find  $k$  Boolean functions of  $n$  variables  $f_1, \dots, f_k : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  such that the matrix

$$A = [\Omega(f_1)^\top, \Omega(f_2)^\top, \dots, \Omega(f_k)^\top] \quad (2)$$

is an  $OA(2^n, k, 2, t)$ , with  $\lambda = 2^{n-t}$ .

In other words, solving Problem 1 requires finding a set of  $k$  Boolean functions of  $n$  variables whose truth tables, when put one next to the other, form the columns of an orthogonal array with  $N = 2^n$  rows,  $k$  columns, 2 levels, and strength  $t$ .

## 3 Specification of GA and GP

### 3.1 Solutions Encoding

Since Problem 1 requires finding a set of  $k$  Boolean functions whose truth tables form an  $OA(2^n, k, 2, t)$ , the encoding of the feasible solutions can be reduced to an appropriate representation of sets of Boolean functions which can be easily handled by evolutionary algorithms. Depending on the underlying heuristic (GA or GP), we adopted the following approaches:

1. *GA encoding:* The chromosome  $c$  of an individual is defined as follows:

$$c = (b_1, \dots, b_k),$$

where, for all  $i \in \{1, \dots, k\}$ ,  $b_i \in \mathbb{F}_2^{2^n}$  is a bitstring of length  $2^n$  that represents the truth table of the  $i$ -th Boolean function  $f_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  composing a feasible solution. The GA crossover and mutation operators are applied component-wise on each bitstring  $b_i$ .

2. *GP encoding:* The chromosome  $c$  in this case is defined as:

$$c = (T_1, \dots, T_k),$$

where, for all  $i \in \{1, \dots, k\}$ ,  $T_i$  is a *Boolean tree* which encodes a Boolean function of  $n$  variables, using a given set of Boolean operators. In particular,

the  $2^n$ -bit string representing the  $i$ -th column of the array is determined by evaluating  $T_i$  for all possible input combinations on the leaf nodes, and taking the corresponding outputs of the function as the values computed at the root node. Similar to the GA encoding case, the GP variation operators are applied component-wise for each tree in the chromosome of an individual (or in a pair of individuals, in the case of tree crossover).

### 3.2 Fitness Function

Once a suitable chromosome encoding has been designed, one needs to define a *fitness function* to determine how good the candidate solutions produced by an evolutionary algorithm are with respect to the optimal ones. In our case, an optimal solution is defined as a set of  $k$  Boolean functions whose truth tables form the columns of a binary orthogonal array. Hence, a preliminary idea could be to determine, for each possible subset of  $t$  columns of a candidate solution, how many  $t$ -uples are repeated more than  $\lambda$  times, and then minimize this deviation over all possible subsets of  $t$  columns.

Let us formalize the discussion above. Given a set of  $k$  Boolean functions  $f_1, \dots, f_k : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ , let  $A$  be the  $2^n \times k$  matrix formed by placing side by side the transpose of the truth tables  $\Omega(f_1), \dots, \Omega(f_k) \in \mathbb{F}_2^{2^n}$ . Additionally, let  $I = \{i_1, \dots, i_t\}$  be a subset of  $t$  indices, with  $1 \leq i_j \leq k$  for all  $j \in \{1, \dots, t\}$ , and let  $A_I$  denote the  $2^n \times t$  submatrix obtained by considering only the columns of  $A$  specified by the indices of  $I$ . For all binary  $t$ -uples  $x \in \mathbb{F}_2^t$ , let  $A_I[x]$  denote the number of occurrences of  $x$  in  $A_I$ , and define the  $\lambda$ -deviation of  $x$  as:

$$\delta(A_I, x) = |\lambda - A_I[x]|. \tag{3}$$

Then, given  $p \in \mathbb{N}$ , we define the  $p$ -deviation of  $A_I$  as:

$$\Delta(A_I)_p = \left( \sum_{x \in \mathbb{F}_2^t} \delta(A_I, x)^p \right)^{\frac{1}{p}}. \tag{4}$$

In particular, one may notice that Eq. (4) corresponds to the *Minkowski distance* (or  $L^p$  distance) between the vector  $\Lambda = (\lambda, \dots, \lambda)$  and the vector  $(A_I[(0, \dots, 0)], \dots, A_I[(1, \dots, 1)])$ .

We can now define the fitness function for our optimization problem, which amounts to the sum of the deviations of all possible  $N \times t$  submatrices of  $A$ :

$$fit_p(A) = \sum_{I \subseteq [k]: |I|=t} \Delta(A_I)_p. \tag{5}$$

Clearly, if  $A$  is an orthogonal array with the required parameters, then  $fit_p(A) = 0$ . As a consequence, the optimization objective is to *minimize*  $fit_p$ .

### 3.3 Variation Operators

Recall from Theorem 1 that any OA of strength  $t$  is also an OA for all strengths  $i < t$ . Considering the extreme case where  $i = 1$ , this means that for each column of the array we must see every symbol of the support set equally often. Since in our problem we are considering binary OA where the number of rows equals  $N = 2^n$ , it follows that each column of an optimal solution must be composed of  $2^{n-1}$  zeros and  $2^{n-1}$  ones or, equivalently, that the corresponding Boolean function of  $n$  variables must be balanced.

We can exploit this fact to reduce the size of the search space of feasible solutions explored by our GA. In fact, since we are interested only in sets of  $k$  balanced Boolean functions, we can adopt variation operators that preserve their balancedness. To this end, we employ a slightly modified version of the crossover operator originally proposed by Millan et al. [6]. In particular, let  $p_1$  and  $p_2$  be two balanced bitstrings. Then, we generate a balanced offspring chromosome  $c$  using the following procedure:

BALANCED-CROSSOVER( $p_1, p_2$ )

**Initialization:** Set two counters  $cnt_0$  and  $cnt_1$  to zero.

**Loop:** Until all positions in the offspring chromosome  $c$  have been filled:

1. Sample a random position  $i \in \{1, \dots, 2^n\}$  (without replacement)
2. If one of the two counters is equal to  $2^{n-1}$ , then set  $c[i]$  to the opposite value (i.e., 1 if  $cnt_0 = 2^{n-1}$  or 0 if  $cnt_1 = 2^{n-1}$ )
3. Otherwise, randomly choose between  $p_1[i]$  and  $p_2[i]$  and copy the corresponding value in  $c[i]$ , increasing the relevant counter.

**Output:** Return  $c$

As one can observe, our crossover operator uses two counters to keep track of the number of zeros and ones in the child chromosome during its generation. Until these two counters are less than half of the chromosome length, a random position is sampled and the gene to be copied is randomly selected from one of the two parents. Then, when one of the two counters reaches the  $2^{n-1}$  threshold, all remaining positions in the child are filled with the opposite value. This ensures that the child chromosome is also balanced.

Regarding the mutation operator, we opted for a simple swap-based operator. More precisely, each column composing an individual is mutated with a small probability by swapping two bits in it, so that the balancedness of the corresponding Boolean function is preserved. In particular, the swap is performed between two random positions holding different values, in order to produce a mutated individual which differs from the original one.

On the contrary, for GP there is no straightforward way to design crossover and mutation operators which ensure that the resulting trees map to the balanced Boolean functions. Hence, in this case we chose to employ classic GP variation operators, specifically simple tree crossover, uniform crossover, size fair, one-point, and context preserving crossover [8] (selected at random) and subtree mutation. Additionally, we considered the balancedness constraint at the fitness

function level, using a penalty factor. In particular, let  $\delta_{0,1}(i) = |\#0 - \#1|$  be the absolute value of the difference between the number of ones and the number of zeros in the  $i$ -th column of a binary array  $A$ . Then, the new fitness function minimized by GP equals:

$$fit_p(A) = \sum_{I \subseteq [k]: |I|=t} \Delta(A_I)_p + \sum_{i=1}^k \delta_{0,1}(i). \quad (6)$$

## 4 Analysis of the Search Space

We now give some basic combinatorial remarks that allow us to compute the sizes of the solution spaces. By taking into account the bare statement of Problem 1, one can see that the number of feasible solutions depends only on the number of columns  $k$  composing the array and on the number of variables  $n$  of the Boolean functions whose truth tables represent those columns. The number of Boolean functions of  $n$  variables is  $2^{2^n}$ , since it equals the number of bitstrings of length  $2^n$ , which are in one-to-one correspondence with the truth tables of such functions. Hence, it follows that the number of ways one can choose a set of  $k$  Boolean functions of  $n$  variables is given by

$$\mathcal{F}_{n,k} = \binom{2^{2^n}}{k}, \quad (7)$$

which corresponds to the size of the search space  $\mathcal{F}_{n,k}$  induced by Problem 1. Indeed  $\mathcal{F}_{n,k}$  is actually a *subset* of the search space explored by our GP algorithm. This is due to the fact that different Boolean trees evolved by GP can be semantically equivalent (i.e. evaluate to the same truth table, such as  $x$  and  $NOT(NOT(x))$ ).

On the contrary, the search space explored by our GA coincides the set of binary  $2^n \times k$  matrices whose columns are balanced, or equivalently to the space of all subsets of  $k$  balanced Boolean functions of  $n$  variables. The number of balanced Boolean functions of  $n$  variables is

$$\mathcal{BAL}_n = \binom{2^n}{2^{n-1}}, \quad (8)$$

since it is equal to the number of bitstrings of length  $2^n$  that include  $2^{n-1}$  ones. Thus, the number of combinations of  $k$  balanced  $n$ -variable Boolean functions is

$$\mathcal{G}_{n,k} = \binom{\mathcal{BAL}_n}{k} = \binom{\binom{2^n}{2^{n-1}}}{k}, \quad (9)$$

which gives the size of the search space explored by GA.

A natural question that arises is up to which values of the parameters  $n$  and  $k$  the two sets  $\mathcal{F}_{n,k}$  and  $\mathcal{G}_{n,k}$  are amenable to exhaustive search. Table 1 reports the corresponding sizes for increasing values of  $n$ , along with the dimensions of the spaces of all Boolean functions and balanced functions of  $n$  variables.

**Table 1.** Search space sizes with respect to  $n$  and  $k$ .

$n$	$N$	$k$	$\mathcal{B}_n$	$\mathcal{B}\mathcal{A}\mathcal{L}_n$	$\mathcal{F}_{n,k}$	$\mathcal{G}_{n,k}$
2	4	2	16	6	120	15
3	8	4	256	70	$1.7 \cdot 10^8$	916 895
4	16	8	65 536	12 870	$8.4 \cdot 10^{33}$	$1.8 \cdot 10^{28}$
5	32	16	$4.2 \cdot 10^9$	$6.0 \cdot 10^8$	$6.4 \cdot 10^{140}$	$1.3 \cdot 10^{127}$

From Table 1, one can see that the sizes of the two search spaces grow very quickly with respect to the number of variables of the Boolean functions involved, and that exhaustive enumeration is already unfeasible for  $n \geq 4$  variables.

## 5 Experiments

### 5.1 Problem Instances

Table 2 reports the problem instances on which we run our GA and GP heuristics. In particular, each row of the table reports the number of variables  $n$  of the involved Boolean functions, the number of rows  $N = 2^n$  of the OA, the number of columns  $k$ , the strength  $t$ , and the index  $\lambda$ . In the rest of this section, we refer to a problem instance by  $(N, k, t, \lambda)$ . We selected these instances from the orthogonal array library published by Sloane [10]. We chose these particular parameters combinations since they contain both instances that can be exhaustively enumerated (those with  $n = 3$ , which we used for tuning our algorithms) and the smallest instances that are not amenable to exhaustive search.

**Table 2.** OA parameters/problem instances

	$I_1$	$I_2$	$I_3$	$I_4$	$I_5$	$I_6$	$I_7$	$I_8$	$I_9$	$I_{10}$
$n$	3	3	3	3	3	4	4	5	5	6
$N$	8	8	8	8	16	16	16	32	32	64
$k$	4	4	5	7	8	8	15	16	31	32
$t$	2	3	2	2	2	3	2	3	2	3
$\lambda$	2	1	2	2	4	2	4	4	8	8

### 5.2 Evolutionary Algorithms Parameters

As mentioned in Sect. 3.1, the GP encoding uses elementary Boolean operators to build one or more trees, each representing an independent Boolean function, whereas the corresponding Boolean variables are used as terminals. The function set in our experiments comprise the binary operators *AND*, *OR*, *XOR*, *XNOR*,

and the unary operator *NOT*. Additionally, we include the function *IF*, which takes three arguments and returns the second one if the first one evaluates to true, and the third one otherwise. The maximum tree depth is varied depending on the number of Boolean variables, which determines the number of rows of the target orthogonal array.

Regarding the population size, we set it to 500 individuals for GP and 50 for GA. The reason for this difference is that after performing some preliminary experiments, we observed that using larger population size in GA did not improve its performance. For the selection process, we employed a steady-state selection with a 3-tournament operator for both GA and GP, that in each iteration randomly selects three individuals for the tournament and eliminates the worst one. A new individual is created immediately by crossing over the remaining two from the tournament, which then undergoes mutation respectively with probability 0.5 in GP and 0.2 in GA.

Concerning the fitness function, after some preliminary tuning tests we observed that using the Minkowski distance with  $p = 2$  yielded the best results, hence we adopted  $fit_2$  for all subsequent experiments. Likewise, we set the termination condition for both GA and GP to 500 000 fitness evaluations after observing from a preliminary round of experiments that optimal solutions are mostly found before reaching this number of evaluations. Finally, each experiment is repeated 30 times.

### 5.3 Results

Table 3 presents the results for genetic algorithms and genetic programming in the form of success rate (in percentages) of finding an optimal solution, i.e., an orthogonal array with given properties. We denote by  $GP_d$  a GP experiment where the maximum tree depth is  $d$ . It can be observed that GP outperforms by far GA at converging to an optimal solution. As a matter of fact, GA is able to generate OA only up to 16 rows and 8 columns, with the (16, 8, 3, 2) problem instance having a very low success rate. On the contrary, GP was able to find an optimal solution at least once in all instances but one (the last row with orthogonal array of 64 rows and 32 columns). Similar to GA, one can see greatly differing success rates depending on the size of the problem instance. We varied the maximum tree depth parameter to determine the conditions under which GP is able to produce an optimal solution. It can be seen that having the maximum tree depth equal to the number of variables  $n$  is enough to obtain an orthogonal array. Reasonably, the problem becomes much harder to solve also for GP when the number of variables and the number of trees (i.e., array columns) grow.

Table 4 shows the basic statistical indicators for the fitness of the best individuals found by GA and GP for every considered problem instance, as well as the average time needed to either obtain an optimal solution, or terminate the run after 500 000 evaluations. In the GA case, we did not experiment with the (64, 32, 3, 8) combination, since as remarked above GA could not even converge on the smaller instances with 32 rows. These results are based on GP experiments with the largest maximum tree depth in every configuration.



**Table 3.** GP and GA success rates for different problem sizes. Success rates are rounded to the nearest integer.

Exp.	Heuristic				
	GA	GP <sub>2</sub>	GP <sub>3</sub>	GP <sub>4</sub>	GP <sub>5</sub>
(8, 4, 2, 2)	100	100	100	-	-
(8, 4, 3, 1)	100	100	100	-	-
(8, 5, 2, 2)	100	100	100	-	-
(8, 7, 2, 2)	87	0	100	-	-
(16, 8, 2, 4)	27	100	100	100	-
(16, 8, 3, 2)	3	0	100	97	-
(16, 15, 2, 4)	0	0	90	93	-
(32, 16, 3, 4)	0	-	6	10	-
(32, 31, 2, 8)	0	-	0	2	-
(64, 32, 3, 8)	-	-	0	0	0

**Table 4.** Statistical indicators for GA and GP (largest max tree depth for GP).

Exp.	GA					GP				
	min	avg	std	max	time (s)	min	avg	std	max	time (s)
(8, 4, 2, 2)	0	0	0	0	<1	0	0	0	0	<1
(8, 4, 3, 1)	0	0	0	0	<1	0	0	0	0	<1
(8, 5, 2, 2)	0	0	0	0	<1	0	0	0	0	<1
(8, 7, 2, 2)	0	0.533	1.38	4	7	0	0	0	0	1
(16, 8, 2, 4)	0	2.333	1.75	6	38	0	0	0	0	1
(16, 8, 3, 2)	0	39.96	10.9	57.41	110	0	0.565	3.09	16.97	13
(16, 15, 2, 4)	52	65.4	6.41	80	147	0	0.533	2.03	8	48
(32, 16, 3, 4)	1 174	1 266	43.4	1 349	1 995	0	83.72	41.5	135.8	1 212
(32, 31, 2, 8)	654	684	14.5	714	1 125	0	32	13.9	64	692
(64, 32, 3, 8)	-	-	-	-	-	18 812	19 159	116	19 355	15 308

The data are consistent with those reported in Table 3, indicating that GP has far better performances than GA on all problem instances, under the same number of fitness evaluations.

## 6 Conclusions and Perspectives

In this paper, we considered how evolutionary algorithms can be used to evolve binary orthogonal arrays. To that end, we formulated the combinatorial optimization problem as the search of a set of  $k$  Boolean functions of  $n$  variables

whose truth tables must be the columns of a binary OA with specified parameters. We chose GA and GP as heuristics to solve this problem, each working on a specific solution encoding for the candidate solutions of the problem.

The results show genetic programming greatly outperforming genetic algorithms. Interestingly, for all instances but one (the largest), GP is able to find at least one successful solution. On the other hand, GA managed to solve at least once only 6 instances out of the 10 considered, with lower success rates than GP. This experimental finding is interesting, as it contrasts with the fact that GA actually explored a smaller search space than that of GP, since the former evolved only sets of balanced Boolean functions. This observation is somewhat analogous to what has been reported by Mariot et al. in [5], where GP also outperformed GA at evolving orthogonal Latin squares based on cellular automata, even though also in that case GA was exploring the smaller search space of *pairwise-balanced* Boolean functions. Considering also our findings, this seems to indicate that GP is a better optimization heuristic at handling problems related to combinatorial designs. Moreover, since there is no other work in the literature concerning the heuristic construction of binary OA, the results that we obtained in our experiments could represent a first baseline of comparison for future research in this domain, for example by investigating the performances of other evolutionary optimization methods like discrete PSO [4] or Cartesian GP [7], which already proved useful to evolve balanced Boolean functions.

More generally, an interesting question pertains the comparison between our evolutionary approach and other non-heuristic methods to construct OA already known in the relevant literature. In particular, one can observe that most of the existing constructions of OA are based on *algebraic methods*, which usually leverages on finite fields and coding theory (see for example [3]). However, it is necessary to remark that a straightforward comparison between our heuristics and these algebraic methods is not possible, due to the great differences that the two approaches adopt to generate OA. Indeed, our heuristic approach casts the problem in terms of optimization: starting from a population of candidate solutions which most likely do not contain an optimal solution, evolve them until an OA is obtained. On the contrary, algebraic methods usually work by constructing new OA starting from previously existing ones, which have already been obtained through other constructions and/or exhaustive search. This the case, for example, of the *juxtaposition construction* or the *X<sub>4</sub> construction* surveyed in [3]. As a consequence, barely looking at the size of the OA produced by our GA and GP would bring to the conclusion that our approach is no match for the more established algebraic constructions, since the latter manage to create significantly bigger OA (as one can see for example in Sloane [10]). However, an important aspect to remark is that most of the known algebraic constructions arise from *linear error-correcting codes*, thus yielding *linear orthogonal arrays*. This basically means that each row in the OA is a linear combination of the remaining rows. Hence, these methods do not provide for a great diversity, and the OA produced by them are actually a subset of all possible OA. On the contrary, our optimization approach does not assume any linearity constraint on the optimal solutions, hence it can generate a wider variety of OA.

Considering the above remarks, it appears natural that the main direction for future research is to improve the performances of GA and GP over larger problem instances, in order to obtain binary OA of higher dimensions. To this end, one could adopt a different fitness function that yields a smoother fitness landscape over the search space of candidate solutions. A possible idea to accomplish this would be to define fitness functions based on Theorem 3.30 in [3], which shows that a binary matrix is a binary OA of strength  $t$  if and only if its *Walsh-Hadamard transform* vanishes for all subsets of rows having at most  $i \leq t$  nonzero entries.

A second direction would be to start from an OA with a certain number  $k$  of columns, and then incrementally add columns which still satisfy the balancedness constraints with the previous ones, thus yielding a larger OA. In this case, GA and GP would work on a single Boolean function at a time, possibly making convergence easier on larger problem instances.

**Acknowledgments.** This work has been supported in part by Croatian Science Foundation under the project IP-2014-09-4882.

## References

1. Carlet, C., Guilley, S.: Correlation-immune boolean functions for easing counter measures to side-channel attacks. *Algebraic Curves Finite Fields: Cryptograph. Other Appl.* **16**, 41–70 (2014)
2. Colbourn, C.J., Dinitz, J.H.: *Handbook of Combinatorial Designs*. CRC Press, Boca Raton (2006)
3. Hedayat, A.S., Sloane, N.J.A., Stufken, J.: *Orthogonal Arrays: Theory and Applications*. Springer, Heidelberg (2012). <https://doi.org/10.1007/978-1-4612-1478-6>
4. Mariot, L., Leporati, A.: Heuristic search by particle swarm optimization of boolean functions for cryptographic applications. In: *Genetic and Evolutionary Computation Conference, Companion Material Proceedings, GECCO 2015, Madrid, Spain, 11–15 July 2015*, pp. 1425–1426 (2015)
5. Mariot, L., Picek, S., Jakobovic, D., Leporati, A.: Evolutionary algorithms for the design of orthogonal latin squares based on cellular automata. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, 15–19 July 2017*, pp. 306–313 (2017)
6. Millan, W., Clark, A., 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). <https://doi.org/10.1007/BFb0054148>
7. Picek, S., Jakobovic, D., Miller, J.F., Batina, L., Cupic, M.: Cryptographic boolean functions: one output, many design criteria. *Appl. Soft Comput.* **40**, 635–653 (2016)
8. Poli, R., Langdon, W.B., McPhee, N.F.: *A Field Guide to Genetic Programming* (2008). <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk>. (With contributions by J.R. Koza)
9. Safadi, R., Wang, R.: The use of genetic algorithms in the construction of mixed multilevel orthogonal arrays. Technical report, Olin Corp Cheshire CT Olin Research Center (1992)

10. Sloane, N.J.: A library of orthogonal arrays. Fixed-level arrays with more than three levels: OA 16(4.2) (2007)
11. Stinson, D.R.: *Combinatorial Designs: Constructions and Analysis*. Springer, Heidelberg (2007). <https://doi.org/10.1007/b97564>
12. Wang, R., Safadi, R.: Generating mixed multilevel orthogonal arrays by simulated annealing. In: Page, C., LePage, R. (eds.) *Computing Science and Statistics*, pp. 557–560. Springer, New York (1992). [https://doi.org/10.1007/978-1-4612-2856-1\\_100](https://doi.org/10.1007/978-1-4612-2856-1_100)



# Heavy-Tailed Mutation Operators in Single-Objective Combinatorial Optimization

Tobias Friedrich<sup>1,2</sup>, Andreas Göbel<sup>1(✉)</sup>, Francesco Quinzan<sup>1</sup>,  
and Markus Wagner<sup>2</sup>

<sup>1</sup> Hasso Plattner Institute, Potsdam, Germany  
{andreas.gobel, francesco.quinzan}@hpi.de  
<sup>2</sup> University of Adelaide, Adelaide, Australia

**Abstract.** A core feature of evolutionary algorithms is their mutation operator. Recently, much attention has been devoted to the study of mutation operators with dynamic and non-uniform mutation rates. Following up on this line of work, we propose a new mutation operator and analyze its performance on the (1+1) Evolutionary Algorithm (EA). Our analyses show that this mutation operator competes with pre-existing ones, when used by the (1+1) EA on classes of problems for which results on the other mutation operators are available. We present a “jump” function for which the performance of the (1+1) EA using any static uniform mutation and any restart strategy can be worse than the performance of the (1+1) EA using our mutation operator with no restarts. We show that the (1+1) EA using our mutation operator finds a (1/3)-approximation ratio on any non-negative submodular function in polynomial time. This performance matches that of combinatorial local search algorithms specifically designed to solve this problem.

Finally, we evaluate experimentally the performance of the (1+1) EA using our operator, on real-world graphs of different origins with up to  $\sim 37\,000$  vertices and  $\sim 1.6$  million edges. In comparison with uniform mutation and a recently proposed dynamic scheme our operator comes out on top on these instances.

**Keywords:** Mutation operators · Minimum vertex cover problem  
Submodular functions maximization

## 1 Introduction

One of the simplest and most studied *evolutionary algorithm* is the (1+1) EA [4, 17, 20] (see Algorithm 1). A key procedure of the (1+1) EA that affects its performance is the *mutation operator*, i.e., the operator that determines at each step how the potential new solution is generated. In the past several years there

---

A full version of this paper is available at <http://arxiv.org/abs/1805.10902>.

has been a huge effort, both from a theoretical and an experimental points of view, towards understanding how this parameter influences the performance of the (1+1) EA and towards deciding which is the optimal way of choosing this parameter (e.g., see [5, 6]).

The most common mutation operator on  $n$ -bit strings is the static *uniform mutation* operator. This operator,  $\text{unif}_p$ , flips each bit of the current solution independently with probability  $p(n)$ . This probability,  $p(n)$ , is called *static mutation rate* and remains the same throughout the run of the algorithm. The most common choice for  $p(n)$  is  $1/n$ ; thus, mutated solutions differ in expectation in 1-bit from their predecessors. Witt [21] shows that this choice of  $p(n)$  is optimal for all pseudo-Boolean linear functions. Doerr et al. [2] further observe that changing  $p(n)$  by a constant factor can lead to large variations of the overall run-time of the (1+1) EA. They also show the existence of functions for which this choice of  $p(n)$  is not optimal.

Static mutation rates are not the only ones studied in literature. Jansen et al. [13] propose a mutation rate which at time step  $t$  flips each bit independently with probability  $2^{(t-1) \bmod (\lceil \log_2 n \rceil - 1)}/n$ . Doerr et al. [3] observe that this mutation rate is equivalent to a mutation rate of the form  $\alpha/n$ , with  $\alpha$  drawn uniformly at random (u.a.r.) from the set  $\{2^{(t-1) \bmod (\lceil \log_2 n \rceil - 1)} \mid t \in \{1, \dots, \lceil \log_2 n \rceil\}\}$ .

Doerr et al. [3] notice that the choice of  $p(n) = 1/n$  is a result of over-tailoring the mutation rates to commonly studied simple unimodal problems. They propose a non-static mutation operator  $\text{fmut}_\beta$ , which chooses a mutation rate  $\alpha \leq 1/2$  from a power-law distribution at every step of the algorithm. Their analysis shows that for a family of “jump” functions introduced below, the run-time of the (1+1) EA yields a polynomial speed-up over the optimal time when using  $\text{fmut}_\beta$ .

Recently, Friedrich et al. [10] propose a new mutation operator. Their operator  $\text{cMut}(p)$  chooses at each step with constant probability  $p$  to flip 1-bit of the solution chosen uniformly at random. With the remaining probability  $1 - p$ , the operator chooses  $k \in \{2, \dots, n\}$  uniformly at random and flips  $k$  bits of the solution chosen uniformly at random. This operator performs well in optimizing pseudo-Boolean functions, as well as combinatorial problems such as the minimum vertex cover and the maximum cut. Experiments suggest that this operator outperforms the mutation operator of Doerr et al. [3] when run on functions that exhibit large deceptive basins of attraction, i.e., local optima whose hamming distance from the global optimum is in  $\Theta(n)$ .

Inspired by the recent results of Doerr et al. [3] and Friedrich et al. [10] we propose the mutation operator  $\text{pmut}_\beta$  that mutates  $n$ -bit string solutions as follows. At each step,  $\text{pmut}_\beta$  chooses  $k \in \{1, \dots, n\}$  from a power-law distribution. Then  $k$  bits of the current solution are chosen uniformly at random and then flipped. During a run of the (1+1) EA using  $\text{pmut}_\beta$ , the majority of mutations consist of flipping a small number of bits, but occasionally a large number, of up to  $n$  bit flips can be performed. In comparison to the mutations of  $\text{fmut}_\beta$ , the mutations of  $\text{pmut}_\beta$  have a considerably higher likelihood of performing larger than  $(n/2)$ -bit jumps. A visualization of these probabilities is shown in Fig. 1. Our results can be summarized as follows.

**Run-Time Comparison on Artificial Landscapes.** In Sect. 3.1 we show that the (1+1) EA using  $\text{pmut}_\beta$  manages to find the optimum of any function within exponential time. When run on the OneMax function, the (1+1) EA with  $\text{pmut}_\beta$  finds the optimum solution in expected polynomial time.

In Sect. 3.2 we consider the problem of maximizing the  $n$ -dimensional jump function, first introduced by Droste et al. [4].

$$\text{Jump}(m, n)(x) = \begin{cases} m + |x|_1 & \text{if } |x|_1 \leq n - m \text{ or } |x|_1 = n; \\ n - |x|_1 & \text{otherwise;} \end{cases}$$

We show that for any value of the parameters  $m, n$  with  $m$  constant or  $n - m$ , the expected run time of the (1+1) EA using  $\text{pmut}_\beta$  remains polynomial. This is not the case for the (1+1) EA using  $\text{unif}_p$ , for which Droste et al. [4] showed a run time of  $\Theta(n^m + n \log n)$  in expectation. Doerr et al. [3] are able to derive polynomial bounds for the expected run-time of the (1+1) EA using their mutation operator  $\text{fmut}_\beta$ , but in their results limit the jump parameter to  $m \leq n/2$ .

**Optimization of Submodular Functions and Experiments.** In Sect. 5 we examine the performance of the (1+1) EA with  $\text{pmut}_\beta$  on submodular functions. Submodular functions arise in the analysis of various optimization problems. Examples include: maximum facility location problems [1], maximum cut and maximum directed cut [11], restricted SAT instances [12]. Submodular functions are also found in AI in connection with probabilistic fault diagnosis problems [14, 15].

Submodular functions exhibit additional properties in some cases, such as *symmetry* and *monotonicity*. These properties can be exploited to derive run time bounds for local randomized search heuristics such as the (1+1) EA. In particular, Friedrich and Neumann [9] give run time bounds for the (1+1) EA and GSEMO on this problem, assuming either monotonicity or symmetry.

We show (Sect. 5.1) that the (1+1) EA with  $\text{pmut}_\beta$  on any non-negative, submodular function gives a 1/3-approximation within polynomial time. This result matches the performance of the local search heuristic of Feige et al. [7] designed to target non-negative, submodular functions in particular. An example of a natural non-negative submodular function that is neither symmetric nor monotone is the utility function of a player in a combinatorial auction (see e.g. [16]).

In Sect. 5.2 we apply our general upper bound to the maximum directed cut problem. Unlike the results of Friedrich et al. [10] we consider graphs with weighted edges, and our run-time bound does not depend on the maximum outdegree.

Finally, we evaluate the performance of the (1+1) EA on the maximum directed cut problem using  $\text{pmut}_\beta$  experimentally, on real-world graphs of different origins, and with up to  $\sim 37\,000$  vertices and  $\sim 1.6$  million edges. Our experiments show that  $\text{pmut}_\beta$  outperforms  $\text{unif}_p$  and  $\text{fmut}_\beta$  on those instances.

---

**Algorithm 1.** General framework for the (1+1) EA

---

```

Choose initial solution  $x \in \{0, 1\}^n$  uniformly at random;
while convergence criterion not met do
   $y \leftarrow \mathbf{Mutation}(x)$  for given mutation operator;
  if  $f(y) \geq f(x)$  then
     $x \leftarrow y$ ;
return  $x$ ;

```

---

## 2 Algorithms and Setting

### 2.1 The (1+1) Evolutionary Algorithm and Mutation Rates

In this paper we look at the run time of the simple (1+1) Evolutionary Algorithm under various configurations. This algorithm requires a bit-string of fixed length  $n$  as input. An offspring is then generated by the *mutation operator*, an operator that resembles asexual reproduction. The fitness of the solution is then computed and the less desirable result is discarded. This algorithm is *elitist* in the sense that the solution quality never decreases throughout the process. Pseudo-code for the (1+1) EA is given in Algorithm 1.

In the (1+1) EA the offspring generated in each iteration depends on the mutation operator. The standard choice for the  $\mathbf{Mutation}(\cdot)$  is to flip each bit of an input string  $x = (x_1, \dots, x_n)$  independently with probability  $1/n$ . In a slightly more general setting, the mutation operator  $\mathbf{unif}_p(\cdot)$  flips each bit of  $x$  independently with probability  $p/n$ , where  $p \in [0, n/2]$ . We refer to the parameter  $p$  as *mutation rate*.

Uniform mutations can be further generalized, by sampling the mutation rate  $p \in [0, n/2]$  at each step according to a given probability distribution. We assume this distribution to be fixed throughout the optimization process. Among this class of mutation rates, is the *power-law* mutation  $\mathbf{fmut}_\beta$  of Doerr et al. [3].  $\mathbf{fmut}_\beta$  chooses the mutation rate according to a power-law distribution on  $[0, 1/2]$  with exponent  $\beta$ . More formally, denote with  $X$  the r.v. (random variable) that returns the mutation rate at a given step. The power-law operator  $\mathbf{fmut}_\beta$  uses a probability distribution  $D_{n/2}^\beta$  s.t.  $\Pr(X = k) = H_{n/2}^\beta k^{-\beta}$ , where  $H_\ell^\beta = \sum_{j=1}^\ell \frac{1}{j^\beta}$ . The  $H_\ell^\beta$ s are known in the literature as generalized harmonic numbers. Interestingly, generalized harmonic numbers can be approximated with the Riemann Zeta function as  $\lim_{\ell \rightarrow +\infty} H_\ell^\beta = \zeta(\beta)$ , with  $\zeta(\cdot)$  the Riemann Zeta function. In particular, harmonic numbers  $H_{n/2}^\beta$  are always upper-bounded by a constant, for increasing problem size and for a fixed  $\beta > 1$ .

### 2.2 Non-uniform Mutation Rates

In this paper we consider an alternative approach to the non-uniform mutation operators described above. For a given probability distribution  $P = [1, \dots, n] \rightarrow \mathbb{R}$  the proposed mutation operator samples an element



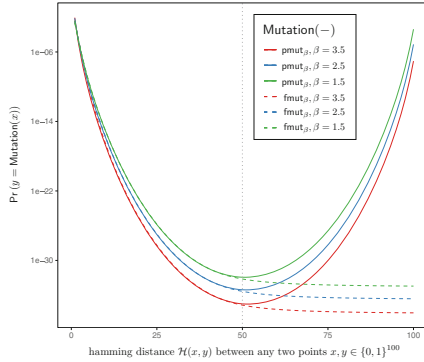
---

**Algorithm 2.** The mutation operator  $\text{pmut}_\beta(x)$

---

$y \leftarrow x$ ;  
 choose  $k \in [1, \dots, n]$  according to distribution  $D_n^\beta$ ;  
 flip  $k$ -bits of  $y$  chosen uniformly at random;  
**return**  $y$ ;

---



**Fig. 1.** A visualization of the probability  $\Pr(y = \text{Mutation}(x))$ , for any two points  $x, y \in \{0, 1\}^n$  w.r.t. the Hamming distance  $\mathcal{H}(x, y)$ , for problem size  $n = 100$  and for  $\beta = 1.5, 2.5, 3.5$ . We consider the case  $\text{Mutation} = \text{pmut}_\beta$  and  $\text{Mutation} = \text{fmut}_\beta$ . Note that the  $y$ -axis follows a logarithmic scale.

$k \in [1, \dots, n]$  according to the distribution  $P$ , and flips *exactly*  $k$ -many bits in an input string  $x = (x_1, \dots, x_n)$ , chosen uniformly at random among all possibilities. This framework depends on the distribution  $P$ , which we always assume fixed throughout the optimization process.

Based on the results of Doerr et al. [3], we study a specialization of our non-uniform framework that uses a distribution of the form  $P = D_n^\beta$ . We refer to this operator as  $\text{pmut}_\beta$ , and pseudocode is given in Algorithm 2. This operator uses a power-law distribution on the probability of performing exactly  $k$ -bit flips in one iteration. That is, for  $x \in \{0, 1\}^n$  and all  $k \in \{1, \dots, n\}$ ,

$$\Pr(\mathcal{H}(x, \text{pmut}_\beta(x)) = k) = (H_n^\beta)^{-1} k^{-\beta} \tag{1}$$

We remark that with this operator, for any two points  $x, y \in \{0, 1\}^n$ , the probability  $\Pr(y = \text{pmut}_\beta(x))$  only depends on their hamming distance  $\mathcal{H}(x, y)$ .

Although both operators,  $\text{fmut}_\beta$  and  $\text{pmut}_\beta$ , are defined in terms of a power-law distribution their behavior differs. A visualization of this can be seen in Fig. 1. We note that, for any choice of the constant  $\beta > 1$  and all  $x \in \{0, 1\}^n$ ,  $\Pr(\mathcal{H}(x, \text{fmut}_\beta(x)) = 0) > 0$ , while  $\Pr(\mathcal{H}(x, \text{pmut}_\beta(x)) = 0) = 0$ . We discuss the advantages and disadvantages of these two operators in Sects. 3 and 4.

### 3 Artificial Landscapes

#### 3.1 General Bounds and the OneMax Function

In this section we derive a general upper-bound on the run time of the (1+1) EA using the mutation operator  $\text{pmut}_\beta$  on any fitness function  $f: \{0, 1\}^n \rightarrow \mathbb{R}$ . It is well-known that the (1+1) EA using uniform mutation on any such fitness function has expected run time at most  $n^n$ . This upper-bound is tight, in the sense that there exists a function  $f$  s.t. the expected run time of the (1+1) EA using uniform mutation to find the global optimum of  $f$  is  $\Omega(n^n)$ . For a discussion on these bounds see Droste et al. [4]. Doerr et al. [3] prove that on any fitness function  $f: \{0, 1\}^n \rightarrow \mathbb{R}$  the (1+1) EA using the mutation operator  $\text{fmut}_\beta$  has run time at most  $\mathcal{O}\left(H_{n/2}^\beta 2^n n^\beta\right)$ . Similarly, we derive a general upper-bound on the run time of the (1+1) EA using mutation  $\text{pmut}_\beta$ .

**Lemma 1.** *On any fitness function  $f: \{0, 1\}^n \rightarrow \mathbb{R}$  the (1+1) EA with mutation  $\text{pmut}_\beta$  finds the optimum solution after expected  $\mathcal{O}\left(H_n^\beta e^{n/e} n^\beta\right)$  fitness evaluations, with the constant implicit in the asymptotic notation independent of  $\beta$ .*

We consider the OneMax function, defined as  $\text{OneMax}(x_1, \dots, x_n) = |x|_1 = \sum_{j=1}^n x_j$ . This simple linear function of unication returns the number of ones in a pseudo-Boolean input string. The (1+1) EA with mutation operators  $\text{unif}_p$  and  $\text{fmut}_\beta$  finds the global optimum after  $\mathcal{O}(n \log n)$  fitness evaluations (see [3, 4, 17]). It can be easily shown that the (1+1) EA with mutation operator  $\text{pmut}_\beta$  achieves similar performance on this instance.

**Lemma 2.** *The (1+1) EA with mutation  $\text{pmut}_\beta$  finds the global optimum of the OneMax after expected  $\mathcal{O}\left(H_n^\beta n \log n\right)$  fitness evaluations, for all  $\beta > 1$  and with the constant implicit in the asymptotic notation independent of  $\beta$ .*

Lemma 2 can be proved using the fitness level method outlined in Wegener [20]. The (1+1) EA with mutation  $\text{pmut}_\beta$  performs a single chosen bit-flip with probability at least  $(H_n^\beta n)^{-1}$  and the expected time for such an event to occur is  $H_n^\beta n$ .

#### 3.2 A Comparison with Static Uniform Mutations

Recall the definition of the *jump* function from the introduction. For  $1 < m < n$  this function exhibits a single *local* maximum and a single *global* maximum. The first parameter of  $\text{Jump}(m, n)$  determines the hamming distance between the local and the global optimum, while the second parameter denotes the size of the input. We present a general upper-bound on the run time of the (1+1) EA on  $\text{Jump}(m, n)$  with mutation operator  $\text{pmut}_\beta$ . Then, following the footsteps of Doerr et al. [3], we compare the performance of  $\text{pmut}_\beta$  with static mutation operators on jump functions for all  $m \leq n/2$ .

**Lemma 3.** Consider a jump function  $f = \text{Jump}(m, n)$  and denote with  $T_{\text{pmut}_\beta}(f)$  the expected run time of the (1+1) EA using the mutation  $\text{pmut}_\beta$  on the function  $f$ .  $T_{\text{pmut}_\beta}(f) = H_n^\beta \binom{n}{m} \mathcal{O}(m^\beta)$ , where the constant implicit in the asymptotic notation is independent of  $m$  and  $\beta$ .

Note that the upper-bound on the run time given in Lemma 3 yields polynomial run time on all functions  $\text{Jump}(m, n)$  with  $m$  constant for increasing problem size and also with  $n - m$  constant for increasing problem size.

Following the analysis of Doerr et al. [3], we can compare the run time of the (1+1) EA with mutation  $\text{pmut}_\beta$  with the (1+1) EA with uniform mutations, on the jump function  $\text{Jump}(m, n)$  for  $m \leq n/2$ .

**Corollary 4.** Consider a jump function  $f = \text{Jump}(m, n)$  with  $m \leq n/2$  and denote with  $T_{\text{pmut}_\beta}(f)$  the run time of the (1+1) EA using the mutation  $\text{pmut}_\beta$  on the function  $f$ . Similarly, denote with  $T_{\text{OPT}}(f)$  the run time of the (1+1) EA using the best possible static uniform mutation on the function  $f$ . Then it holds  $T_{\text{pmut}_\beta}(f) \leq cm^{\beta-0.5} H_n^\beta T_{\text{OPT}}(f)$ , for a constant  $c$  independent of  $m$  and  $\beta$ .

The result above holds because Doerr et al. [3] prove that the best possible optimization time for a static mutation rate a function  $f = \text{Jump}(m, n)$  with  $m \leq n/2$  is lower-bounded as  $1/2 n^m / m^m (n/(n - m))^{n-m} \leq T_{\text{OPT}}(f)$ .

## 4 An Application to the Minimum Vertex Cover Problem

In this section, we study the *minimum vertex cover* problem (MVC): Given a graph  $G = (V, E)$  with  $n$  vertices, find a minimal subset  $U \subseteq V$  such that each edge in  $E$  is incident to at least one vertex in  $U$ . Following Friedrich et al. [8], we approach MVC by minimizing the functions  $(u(x), |x|_1)$  in lexicographical order, where  $u(x)$  is the number of uncovered edges.

**Lemma 5.** On any graph  $G = (V, E)$ , the (1+1) EA with mutation  $\text{pmut}_\beta$  finds a not necessarily minimum vertex cover after expected  $\mathcal{O}(H_n^\beta n \log n)$  fitness evaluations.

This lemma follows from Friedrich et al. [8, Theorems 1 and 2] and (1) for  $k = 1$ .

The (1 + 1) EA using  $\text{unif}_p$  as a mutation operator, when solving MVC on complete bipartite graphs, does not find the global optimum within polynomial time. Consider the complete bipartite graph  $G = (V, E)$  with partitions  $V_1, V_2$  of size  $m$  and  $n - m$  respectively, where  $0 < m < n/2$ . The expected run time of the (1 + 1) EA using  $\text{unif}_p$  on this instance is at least  $\Omega(mn^{m-1} + n \log n)$ . For  $m \leq n/3$  the (1 + 1) EA using mutation  $\text{fmut}_\beta$  finds the global optimum of MVC after at most  $\mathcal{O}\left(H_{n/2}^\beta n^\beta 2^m\right)$  fitness evaluations in expectation and for  $m \geq n/3$  after at most  $\mathcal{O}\left(H_{n/2}^\beta n^\beta 2^n\right)$  fitness evaluations in expectation. For a discussion on these run time bounds see Friedrich et al. [8] and Doerr et al. [3].

**Theorem 6.** On any complete bipartite graph  $G = (V, E)$ , the (1+1) EA using mutation  $\text{pmut}_\beta$  finds a solution to the MVC after expected  $\mathcal{O}\left(H_n^\beta (n \log n + n^\beta)\right)$  fitness evaluations.

## 5 Maximizing Submodular Functions

### 5.1 A General Upper-Bound

Consider a finite set  $V$  and a function  $f: 2^V \rightarrow \mathbb{R}$ . We say that  $f$  is *submodular* if for all  $U, W \subseteq V$ ,  $f(U) + f(W) \geq f(U \cup W) + f(U \cap W)$ . We consider the problem of maximizing a non-negative submodular function, with the (1+1) EA using the mutation operator  $\text{pmut}_\beta$ . This problem is APX-complete. That is, this problem is NP-hard and does not admit a polynomial time approximation scheme (PTAS), unless  $\text{P} = \text{NP}$ .

We prove that the (1+1) EA with mutation  $\text{pmut}_\beta$  is a  $(1/3 - \varepsilon/n)$ -approximation algorithm for the problem of maximizing a submodular function. In our analysis we assume neither monotonicity nor symmetry. We approach this problem by searching for  $(1 + \alpha)$ -local optima, which we define below.

**Definition 7.** *Let  $f: 2^V \rightarrow \mathbb{R}_{\geq 0}$  be any submodular function. A set  $S \subseteq V$  is a  $(1 + \alpha)$ -local optimum if it holds  $(1 + \alpha)f(S) \geq f(S \setminus \{u\})$  for all  $u \in S$ , and  $(1 + \alpha)f(S) \geq f(S \cup \{v\})$  for all  $v \in V \setminus S$ , for a constant  $\alpha > 0$ .*

The definition given above is useful in the analysis because it can be proved that either  $(1 + \alpha)$ -local optima or their complement always yield a good approximation of the global maximum.

**Theorem 8.** *Consider a non-negative submodular function  $f: 2^V \rightarrow \mathbb{R}_{\geq 0}$  over a set of cardinality  $|V| = n$  and let  $S$  be a  $(1 + \varepsilon/n^2)$ -local optimum. Then either  $S$  or  $V \setminus S$  is a  $(1/3 - \varepsilon/n)$ -approximation of the global maximum.*

We remark that Theorem 8 as we present it is implicit in the proof of Theorem 3.4 in Feige et al. [7]. Also, it is possible to construct examples of submodular functions that exhibit  $(1 + \varepsilon/n^2)$ -local optima with arbitrarily bad approximation ratios. Thus,  $(1 + \varepsilon/n^2)$ -local optima alone do not yield any approximation guarantee with respect to the global maximum, unless the valuation oracle is symmetric.

We can use Theorem 8 to estimate the run time of the (1+1) EA using mutation  $\text{pmut}_\beta$  to maximize a given submodular function. Intuitively, it is always possible to find a  $(1 + \varepsilon/n^2)$ -local optimum in polynomial time using single bit-flips. It is then possible to compare the approximate local solution  $S$  with its complement  $V \setminus S$  by flipping all bits in one iteration.

**Theorem 9.** *Let  $f: 2^V \rightarrow \mathbb{R}_{\geq 0}$  be a non-negative submodular function over a set of cardinality  $|V| = n$ . Then the (1+1) EA with mutation  $\text{pmut}_\beta$  finds a  $(1/3 - \varepsilon/n)$ -approximation of the global maximum after expected  $\mathcal{O}\left(\frac{1}{\varepsilon}n^3 \log\left(\frac{n}{\varepsilon}\right) + n^\beta\right)$  fitness evaluations.*

### 5.2 An Application to the Maximum Directed Cut Problem

Let  $G = (V, E)$  be a graph, together with a weight function  $w: E \mapsto \mathbb{R}_{\geq 0}$  on the edges. We assume the weights to be non-negative. We consider the problem

of finding a subset  $U \subseteq V$  of nodes such that the sum of the weights on the outer edges of  $U$  is maximal. This problem is the maximum directed cut problem (Max-Di-Cut) and is known to be NP-complete. In contrast to Friedrich and Neumann [9], our analysis considers both directed and undirected graphs, although it might be possible to obtain improved bounds on undirected graphs. Furthermore, unlike Friedrich et al. [10] our run-time bound does not depend on the size of the maximum cut in  $G$ .

We first define the cut function.

**Definition 10.** *Let  $G = (V, E)$  be a graph together with a non-negative weight function  $w: E \rightarrow \mathbb{R}_{\geq 0}$ . For each subset of nodes  $U \subseteq V$ , consider the set  $\Delta(U) = \{(e_1, e_2) \in E: e_1 \in U \text{ and } e_2 \notin U\}$  of all edges leaving  $U$ . We define the cut function  $f: 2^V \rightarrow \mathbb{R}_{\geq 0}$  as  $f(U) = \sum_{e \in \Delta(U)} w(e)$ .*

Since we require the weights to be non-negative, the cut function is also non-negative. For any graph  $G = (V, E)$  the corresponding cut function is always sub-modular and, in general, non-monotone (see e.g. [7, 9]). If a graph  $G$  is directed, then the corresponding cut function needs not be symmetric. Using Theorem 9, we derive the following upper-bound on the run time.

**Corollary 11.** *Let  $G = (V, E)$  be a graph of order  $n$  together with a non-negative weight function  $w: E \rightarrow \mathbb{R}_{\geq 0}$ . Then the  $(1+1)$  EA with mutation  $\text{pmut}_\beta$  is a  $(1/3 - \varepsilon/n)$ -approximation algorithm for the Max-Di-Cut on  $G$ . Its expected optimization time is  $\mathcal{O}\left(\frac{1}{\varepsilon}n^3 \log\left(\frac{n}{\varepsilon}\right) + n^\beta\right)$ .*

### 5.3 Experiments on Large Real Graphs

For our experimental investigations, we select the 123 large instances used by Wagner et al. [19]. The number of vertices ranges from about 400 to over 6 million and the number of edges ranges from about 1000 to over 56 million. All 123 instances are available online [18].

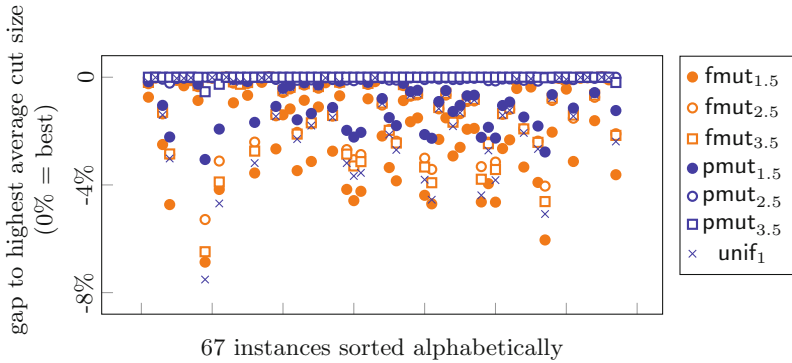
The instances vary widely in their origin. For example, we include 14 collaboration networks (ca-\*, from various sources such as Citeseer and also Hollywood productions), 14 web graphs (web-\*, showing the state of various subsets of the internet at particular points in time), five infrastructure networks (inf-\*), six interaction networks (ia-\*, e.g. about email exchange), 21 general social networks (soc-\*, e.g., Delicious, LastFM, Youtube) and 44 subnets of Facebook (socfb-\*, mostly from different American universities). We take these graphs and run Algorithm 1 with different mutation operators:  $\text{fmut}_\beta$  and  $\text{pmut}_\beta$  with  $\beta \in \{1.5, 2.5, 3.5\}$  and  $\text{unif}_1$ . The solution representation is based on vertices and we initialize uniformly at random. Each edge has a weight of 1.

We perform 100 independent runs (100 000 evaluations each) with an overall computation budget of 72 h per mutation-instance pair. Out of the initial 123 instances 67 finish their 100 repetitions per instance within this time limit.<sup>1</sup>

<sup>1</sup> Source categories of the 67 instances: 2x bio-\*, 6x ca-\*, 5x ia-\*, 2x inf-\*, 1x soc-\*, 40x socfb-\*, 4x tech-\*, 7x web-\*. The largest graph is socfb-Texas84 with 36 364 vertices and 1 590 651 edges.

**Table 1.** Average ranks (based on mean cut size) at  $t = 10\,000$  and  $t = 100\,000$  iterations (lower is better). Our  $\text{pmut}_\beta$  approaches perform best at both budgets.  $\text{unif}_1$  or  $\text{fmut}_{1.5}$  have the worst average rank. The colors correspond to the average rank of a scheme (colder colors are better).

mutation	t=10k	t=100k
$\text{fmut}_{1.5}$	3.4	6.8
$\text{fmut}_{2.5}$	4.9	5.1
$\text{fmut}_{3.5}$	5.8	4.6
$\text{pmut}_{1.5}$	1.6	3.1
$\text{pmut}_{2.5}$	2.2	1.9
$\text{pmut}_{3.5}$	3.3	1.1
$\text{unif}_1$	6.8	4.9



**Fig. 2.** Distance of average cut size to best average of the seven approaches.

We will report on these 67 in the following, and we will use the average cut size achieved in the 100 runs as the basis for our analyses.

First, we rank the seven approaches based on the average cut size achieved in 100 independent runs (best rank is 1, worst rank is 7). Table 1 shows the average rank achieved by the seven different mutation approaches across the 68 instances. It is obvious that  $\text{unif}_1$  is among the worst.  $\text{pmut}_\beta$  clearly performs best, however, while  $\text{pmut}_\beta$  with  $\beta = 1.5$  performs best at 10 000 iterations,  $\text{pmut}_\beta$  with  $\beta = 3.5$  performs best when the budget is 100 000 iterations.

Across the 67 instances, the achieved cut sizes vary significantly (see Fig. 2 and Table 2). For example, the average gap between the worst and the best approach is 46% at 10 000 iterations and it still is 8.1% at 100 000 iterations. Also, when we compare the best  $\text{fmut}_\beta$  and  $\text{pmut}_\beta$  configurations (as per Table 2), then we can see that (i)  $\text{pmut}_\beta$  is better or equal to  $\text{fmut}_\beta$ , and (ii) the performance advantage of  $\text{pmut}_\beta$  over  $\text{fmut}_\beta$  is 2.2% and 1.3% on average, with a maximum of 4.8% and 6.4% (i.e., for 10 000 and 100 000 evaluations).

**Table 2.** Summary of cut-size differences. “Total” refers to the gap between the best and worst performing mutation out of all seven. The two highlighted pairs compare the best  $\text{fmut}_\beta$  and  $\text{pmut}_\beta$  values listed in Table 1.

	$t = 10k$		$t = 100k$	
	Total	$\text{pmut}_{1.5}$ vs $\text{fmut}_{1.5}$	Total	$\text{pmut}_{3.5}$ vs $\text{fmut}_{3.5}$
Min gap	0.3%	0.3%	0.0%	0.0%
Mean gap	12.2%	2.2%	2.1%	1.3%
Max gap	46.0%	4.8%	8.1%	6.4%

## 6 Discussion

In the pursuit of optimizers for complex landscapes that arise in industrial problems, we have identified a new mutation operator. This operator allows for good performance of the classical (1+1) EA when optimizing not only simple artificial test functions, but the whole class of non-negative submodular functions. As submodular functions find applications in a variety of natural settings, it is interesting to consider the potential utility of our operator as a building block for optimizers of more complex landscapes, where submodularity can be identified in parts of these landscapes.

**Acknowledgements.** The authors would like to thank Martin Krejca for giving his advice on one of the proofs, and Karen Seidel for proof-reading the paper.

## References

1. Ageev, A.A., Sviridenko, M.: An 0.828-approximation algorithm for the uncapacitated facility location problem. *Discrete Appl. Math.* **93**(2–3), 149–156 (1999)
2. Doerr, B., Jansen, T., Sudholt, D., Winzen, C., Zarges, C.: Mutation rate matters even when optimizing monotonic functions. *Evol. Comput.* **21**(1), 1–27 (2013)
3. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: GECCO, pp. 777–784 (2017)
4. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. *Theoret. Comput. Sci.* **276**(1–2), 51–81 (2002)
5. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* **3**(2), 124–141 (1999)
6. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computation. Natural Computing Series. Springer, Heidelberg (2003). <https://doi.org/10.1007/978-3-662-05094-1>
7. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. *SIAM J. Comput.* **40**(4), 1133–1153 (2011)
8. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. *Evol. Comput.* **18**(4), 617–633 (2010)
9. Friedrich, T., Neumann, F.: Maximizing submodular functions under matroid constraints by evolutionary algorithms. *Evol. Comput.* **23**(4), 543–558 (2015)

10. Friedrich, T., Quinzan, F., Wagner, M.: Escaping large deceptive basins of attraction with heavy mutation operators. In: GECCO (2018, accepted). <https://hpi.de/friedrich/docs/paper/GECCO18.pdf>
11. 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)
12. Håstad, J.: Some optimal inapproximability results. *J. ACM* **48**(4), 798–859 (2001)
13. Jansen, T., Wegener, I.: Real royal road functions-where crossover provably is essential. *Discrete Appl. Math.* **149**(1–3), 111–125 (2005)
14. Krause, A., Guestrin, C.: Near-optimal observation selection using submodular functions. In: AAAI, pp. 1650–1654 (2007)
15. Lee, J., Mirrokni, V.S., Nagarajan, V., Sviridenko, M.: Non-monotone submodular maximization under matroid and knapsack constraints. In: STOC, pp. 323–332 (2009)
16. Lehmann, B., Lehmann, D.J., Nisan, N.: Combinatorial auctions with decreasing marginal utilities. *Games Econ. Behav.* **55**(2), 270–296 (2006)
17. Mühlenbein, H.: How genetic algorithms really work: mutation and hillclimbing. In: PPSN, pp. 15–26 (1992)
18. Rossi, R.A., Ahmed, N.K.: The Network Data Repository with Interactive Graph Analytics and Visualization (Website) (2015). <http://networkrepository.com>
19. Wagner, M., Friedrich, T., Lindauer, M.: Improving local search in a minimum vertex cover solver for classes of networks. In: CEC, pp. 1704–1711 (2017)
20. Wegener, I.: Theoretical aspects of evolutionary algorithms. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 64–78. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-48224-5\\_6](https://doi.org/10.1007/3-540-48224-5_6)
21. 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). [https://doi.org/10.1007/978-3-540-31856-9\\_4](https://doi.org/10.1007/978-3-540-31856-9_4)





# Heuristics in Permutation GOMEA for Solving the Permutation Flowshop Scheduling Problem

G. H. Aalvanger<sup>1</sup>, N. H. Luong<sup>2</sup>, P. A. N. Bosman<sup>2</sup>, and D. Thierens<sup>1</sup>(✉)

<sup>1</sup> Institute of Information and Computing Sciences,  
Universiteit Utrecht, Utrecht, The Netherlands  
d.thierens@uu.nl

<sup>2</sup> Centre for Mathematics and Computer Science (CWI),  
Amsterdam, The Netherlands

**Abstract.** The recently introduced permutation Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) has shown to be an effective Model Based Evolutionary Algorithm (MBEA) for permutation problems. So far, permutation GOMEA has only been used in the context of Black-Box Optimization (BBO). This paper first shows that permutation GOMEA can be improved by incorporating a constructive heuristic to seed the initial population. Secondly, the paper shows that hybridizing with job swapping neighborhood search does not lead to consistent improvement. The seeded permutation GOMEA is compared to a state-of-the-art algorithm (VNS4) for solving the Permutation Flowshop Scheduling Problem (PFSP). Both unstructured and structured instances are used in the benchmarks. The results show that permutation GOMEA often outperforms the VNS4 algorithm for the PFSP with the total flowtime criterion.

## 1 Introduction

Recently, Bosman et al. [2] introduced the permutation Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA), a model-based evolutionary algorithm which is able to solve permutation problems from a Black-Box Optimization (BBO) perspective. Permutation GOMEA has been tested on the Permutation Flowshop Scheduling Problem (PFSP) with the total flowtime (TFT) criterion. In these tests, permutation GOMEA outperformed GM-EDA [3] another permutation model-based evolutionary algorithm. In order to improve permutation GOMEA further, we should shift from a BBO perspective to a White-Box perspective. In this paper we study the effect of seeding the initial population with solutions from a constructive heuristic, and we look at hybridizing permutation GOMEA with neighborhood search heuristics.

Section 2 briefly introduces permutation GOMEA. After this we explain the PFSP and benchmark instances and performance measures in Sect. 3. Constructive heuristics for the PFSP are given in Sect. 4.1, along with some experiments

on the effectiveness of these heuristics. In Sect. 4.3 we do the same for improvement heuristics for the PFSP. Finally, we compare the permutation GOMEA - seeded with a constructive heuristic - with VNS4, a state-of-the-art algorithm for solving the PFSP in Sect. 5. Section 6 concludes this paper.

## 2 Permutation GOMEA

### 2.1 Solution and Model Encoding

Permutation GOMEA encodes solutions using a random-key encoding [2]. A permutation of  $n$  variables is encoded as  $r = (r_1, \dots, r_n)$ , where each random key  $r_i \in [0, 1]$ . The position of variable  $i$  in the permutation is equal to the position of  $r_i$  when  $r$  is sorted in ascending order. Multiple random key encodings can encode the same permutation. For example,  $r_1 = (0.34, 0.56, 0.21)$  and  $r_2 = (0.72, 0.93, 0.12)$  both encode  $x = (3, 1, 2)$ .

### 2.2 Model Building

The model used in permutation GOMEA is a linkage tree that models dependencies between problem variables in a hierarchical manner [9]. The root of the linkage tree is a set with all variables. Each node is recursively split up, ending in leaves containing only a single variable. Variables grouped in a node are assumed to be dependent, so optimal mixing can improve solutions effectively.

In permutation GOMEA, the linkage tree is built in each generation anew, by merging nodes starting at the bottom of the tree. The two sets  $i$  and  $j$  are merged which have the strongest dependency  $\delta(I, J)$ . For two variables  $i$  and  $j$ , the dependency is composed of two factors:  $\delta(i, j) = \delta_1(i, j) \cdot \delta_2(i, j)$ . The first dependency factor is based on relative-ordering information in the population and is calculated using the entropy of the probability that variable  $i$  is before variable  $j$  in the population:

$$\delta_1(i, j) = 1 - Entropy(p_{i,j}). \quad (1)$$

The second dependency factor uses the average squared distance in random key values of variable  $i$  and  $j$ :

$$\delta_2(i, j) = 1 - \frac{1}{n} \sum_{k=0}^{n-1} (r_i^k - r_j^k)^2. \quad (2)$$

This results in a symmetric dependency measure between two variables, where high values indicate a high dependency. We can extend the dependency measure to calculate the dependency between two sets, by taking the average pairwise dependency of the variables in the sets:

$$\delta(I, J) = \frac{1}{|I| \cdot |J|} \sum_{i \in I} \sum_{j \in J} \delta(i, j). \quad (3)$$

### 2.3 Optimal Mixing

To generate new solutions, permutation GOMEA uses Gene-pool Optimal Mixing (GOM) [9]. For each solution, permutation GOMEA takes every set in the linkage tree as a crossover mask. The values of the masked variables are then substituted by values from a random donor solution. For example, solution  $r_1 = (0.2, 0.3, 0.6, 0.5)$  is changed using crossover mask  $(x_1, x_2, x_4)$  and donor  $r_2 = (0.9, 0.5, 0.1, 0.7)$  to  $r'_1 = (\mathbf{0.9}, \mathbf{0.5}, 0.6, \mathbf{0.7})$ . If such a change is not strictly improving a solution, the substitution is reverted. Thanks to the random keys encoding, optimal mixing always results in a feasible permutation.

If a solution is not improved using any crossover mask, permutation GOMEA will ‘force’ improvements using the best known solution so far. In this Forced Improvement (FI) phase, permutation GOMEA repeats optimal mixing but the best known solution is used as donor, instead of a random one. In order to improve convergence changes are accepted when they do not decrease the quality of the solution. FI is also entered if the best overall solution has not changed for  $10 + 10 \cdot \log n$  generations (denoted with variable  $NIS = true$  in Algorithm 1).

With a probability of 0.1, permutation GOMEA will ‘scale’ the random keys before substitution. Here, the values to substitute are scaled to a new interval. For example, scaling random keys  $(0.9, 0.5, 0.7)$  to the interval  $[0.3, 0.5]$  results in  $(0.5, 0.3, 0.4)$ . Scaling allows permutation GOMEA to move a group of variables closer together in the permutation. Also, the random key diversity is improved in the population. Random key diversity is also ensured by re-encoding. After the GOM phase of permutation GOMEA, each random key gets a new value, while retaining the order of the random keys.

### 2.4 Population Sizing Scheme

When implemented, permutation GOMEA would look like the pseudocode in Algorithm 1. However, one needs to specify the population size before running the algorithm. Therefore, permutation GOMEA incorporates an exponential population sizing scheme [2]. In this scheme, a population is started with size  $n_{base}$ . Every four times this population is evaluated, a population with size  $2 \cdot n_{base}$  is evaluated once. This pattern recurses, so population  $i$  is evaluated four times as often as population  $i + 1$ . Using such a scheme, no population size has to be estimated. When a population is converged, no evaluations are performed anymore for that population, allowing permutation GOMEA to evaluate more in the other populations.

## 3 Permutation Flowshop Scheduling Benchmark

The PFSP is concerned with finding the optimal solution for scheduling  $J$  jobs on  $M$  machines. Each job requires  $M$  operations, which should be performed sequentially, starting on machine 1 and finishing on machine  $M$  (the *Flowshop* property). Operations cannot be interrupted, but a job can be delayed when its

**Result:** A good/optimal solution with respect to fitness function  $f$

```

Pop ← rand_Pop(n) ;
while ¬termination_criterion do
    LT ← build_LT(Pop) ; // Model-building
    foreach receiver ∈ Pop do
        receiver* ← receiver;
        improved ← False;
        foreach set ∈ LinkageTree do // Gene-pool Optimal Mixing
            donor ← Random(Pop);
            child ← Donate_rescale(receiver*, set, donor, Rand(0, 1) < 0.1);
            if f(child) ≥ f(receiver*) then
                receiver* ← child;
            if f(child) > f(receiver*) then
                improved ← True;
        if ¬improved ∨ NIS then // Forced Improvement
            foreach set ∈ LinkageTree do
                child ←
                    Donate_rescale(receiver*, set, best_solution, Rand(0, 1) < 0.1);
                if f(child) ≥ f(receiver*) then
                    receiver* ← child;
                    break
            receiver = Reencode(receiver*) // Re-encoding
return best solution from Pop
    
```

**Algorithm 1.** GOMEA outline

operations are not performed immediately after each other. Any solution can be seen as a permutation of jobs, since each machine has to process the jobs in the same order (the *Permutation* property). In three field notation, the PFSP is denoted by  $F|prmu|\gamma$ , where  $\gamma$  refers to the objective function that is used for optimizing the schedule. Here, we consider the total flowtime (TFT) criterion, which is defined as the sum of completion times of all jobs:

$$TFT(\pi) = \sum_{i=1}^J c(\pi_i, M). \quad (4)$$

The completion times of all jobs can be calculated using the equations in (5) in  $\mathcal{O}(J \cdot M)$  time. For the TFT criterion, the PFSP is NP-hard when  $M > 1$ .

$$\begin{aligned}
 c(\pi_1, 1) &= p(\pi_1, 1) \\
 c(\pi_1, j) &= c(\pi_1, j-1) + p(\pi_1, j) \quad \text{for } j = 2 \dots M \\
 c(\pi_i, 1) &= c(\pi_{i-1}, 1) + p(\pi_i, 1) \quad \text{for } i = 2 \dots J \\
 c(\pi_i, 1) &= \max\{c(\pi_{i-1}, j), c(\pi_i, j-1)\} + p(\pi_i, j), \\
 &\quad \text{for } i = 2 \dots J; \text{ for } j = 2 \dots M.
 \end{aligned} \quad (5)$$

Here,  $p(\pi_i, j)$  is the processing time of job  $\pi_i$  on machine  $j$ . The completion time of job  $\pi_i$  on machine  $j$  (i.e.,  $c(\pi_i, j)$ ) is the duration from when job  $\pi_i$  is started on the first machine until job  $\pi_i$  is finished on machine  $j$ .

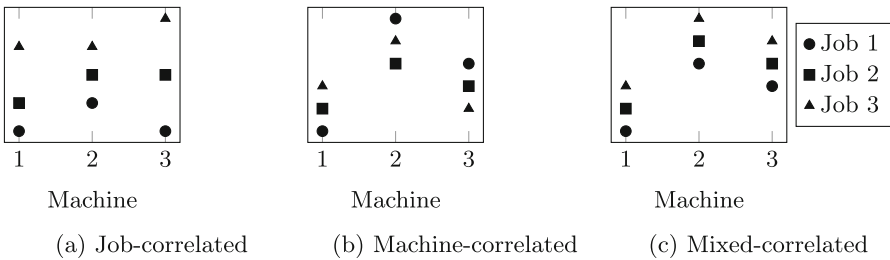
### 3.1 Problem Instances

#### Taillard Instances

For the PFSP, the most often used benchmark set is developed by Taillard [7]. This benchmark set can be divided in 12 ( $J \times M$ ) sets with 10 instances each (See Table 1). The instances are a selection of the hardest randomly generated instances. Here, instances for which simple metaheuristics do not often find the same solution or where the solution is far from a lower bound are considered to be hard.

#### Structured Instances

Aalvanger [1] introduced a new set of benchmarks for testing algorithms on structured instances. The benchmark set contains the three types of structured instances as described by Watson [10]: Job-correlated (JC), Machine-correlated (MC) and Mixed-correlated (MXC) instances (see Fig. 1). In job-correlated instances, processing times are dependent on the job and not on the machines. Therefore the processing times of operations in one job are related. In machine-correlated instances the structure goes the other way around. Here, processing times on one machine are related, while processing times within one job are unrelated. Mixed-correlated instances are equal to Machine-correlated instances, but here the relative ranks of processing times within each machine are job-dependent.



**Fig. 1.** Job processing time for three types of structured PFSP instances.

For each of the three correlation types, four ( $J \times 20$ ) sets are generated (See underlined in Table 1). For each instance size, 1100 instances are generated, with varying values for correlation:  $\alpha \in \{0.0, 0.1 \dots 1.0\}$ . For  $\alpha = 0.0$ , instances reflect the way Taillard instances are generated, higher values introduce more correlation. For  $\alpha = 1.0$ , every task in a job/machine has the same processing time.

### 3.2 Comparing Results

To compare algorithms for PFSP, the Relative Percentage Deviation (RPD) is often used. The RPD describes the relative distance to the best known upper

**Table 1.** Sizes of the Taillard PFSP instances, for underlined sizes structured instances are available.

	$J = 20$	$J = 50$	$J = 100$	$J = 200$	$J = 500$
$M = 5$	$20 \times 5$	$50 \times 5$	$100 \times 5$		
$M = 10$	$20 \times 10$	$50 \times 10$	$100 \times 10$	$200 \times 10$	
$M = 20$	<u><math>20 \times 20</math></u>	$50 \times 20$	<u><math>100 \times 20</math></u>	<u><math>200 \times 20</math></u>	$500 \times 20$

bound ( $UB$ ) of an instance and the result of the algorithm  $RES$ . The RPD is calculated by

$$RPD(RES) = \frac{100 \cdot (RES - UB)}{UB}. \tag{6}$$

RPD values are best used when the upper bound is very close to the optimal solution. An RPD value of 0.0 then means that the optimal solution has been found. Over a set of runs, the average or median RPD is often reported (ARPD/MRPD). In our results, we also report the average over the MRPDs of multiple instances (AMRPD).

We use the Mann-Whitney-U test to check for a significant difference between two algorithms. Unless reported otherwise, we use sample sizes of 20 per instance to find MRPD values. AMRPD values are found over 10 instances with the same size. For significance tests we use a significance level of  $p < 0.05$ .

## 4 Heuristics for the PFSP

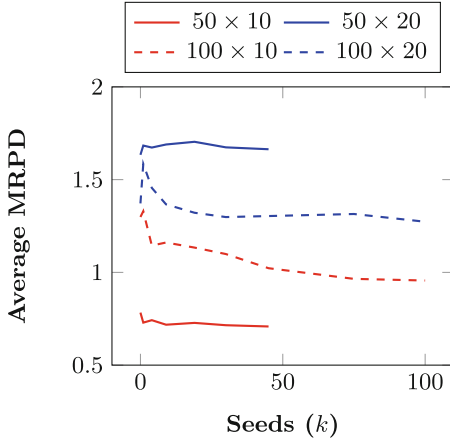
### 4.1 Constructive Heuristics

For the TFT criterion, Liu and Reeves have introduced the LR( $x$ ) heuristic [6], which can generate up to  $J$  schedules, depending on the parameter  $x$ . LR( $x$ ) builds a schedule from the front to the back, using the following three steps:

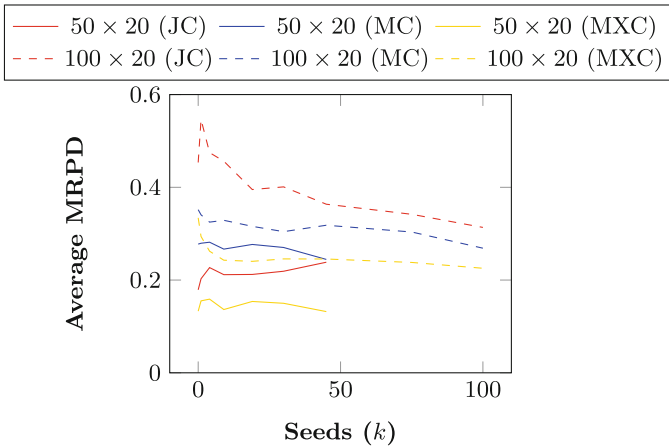
1. Sort all jobs according to the index function.
2. Create  $x$  partial schedules with the top- $x$  jobs scheduled first. Extend the partial schedules by iteratively adding the best job according to the re-evaluated index function.
3. Select the best schedule generated in step 2).

The index function for adding job  $i$  after the last job  $k$  in the partial schedule consists of two components:

1. A *weighted total machine idle time*, penalizing the time the machines wait between job  $k$  and job  $i$ . Idle time on the first machines is punished more than idle time on the last machines.
2. The *artificial total flow time*, is the sum of the completion time of job  $i$  plus the completion time of an artificial job representing the unscheduled jobs.



(a) Taillard instances

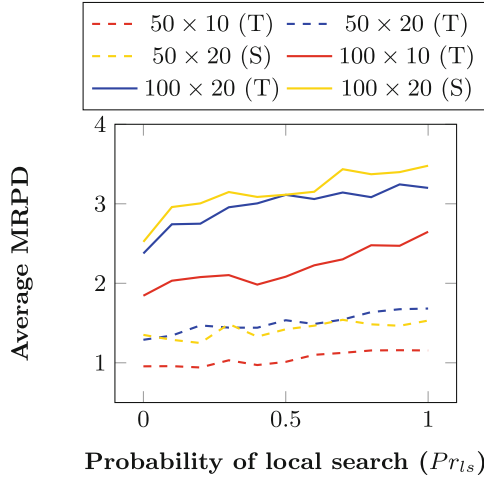


(b) Structured instances

**Fig. 2.** Seeding with the LR heuristics: amount of seeds vs. solution quality after 50,000,000 fitness evaluations.

## 4.2 Constructive Heuristics Seeding: Results

For the LR heuristic we have tested the effect of seeding solutions in the initial populations of permutation GOMEA. Figure 2 shows that for most instances - especially the larger ones - more seeds result in better solutions. This holds for both structured and unstructured instances. An interesting observation is the effect of single-solution seeding. Here, the dominant new solution can misguide optimal mixing, leading to worse solutions.



**Fig. 3.** Hybrid GOMEA performance with respect to the probability of local search for Taillard (T) and structured (S) instances ( $\alpha = 0.3$ )

### 4.3 Improvement Heuristics

For the PFSP with the TFT criterion, various improvement heuristics exist. Each of these improvement heuristics are based on two fundamental permutation neighborhoods: job insertion and job swap. The swap heuristic takes two jobs and swaps them in a permutation. The insertion heuristic takes one job and puts it in another place in the permutation. Both heuristics have a neighbor-space that is quadratic in the amount of jobs and take  $\mathcal{O}(J \cdot M)$  time to compute the fitness of a neighbor. In permutation GOMEA an improvement heuristic is most effectively applied when a solution has changed in the GOM phase. For permutation GOMEA solving the PFSP with the TFT criterion, the swap heuristic was shown to have the most potential, especially on instances with a few machines (for more details see [1]). Figure 3 shows for structured (mixed-correlation) and unstructured instances how permutation GOMEA performs when this improvement heuristic is applied with some probability  $Pr_{ls}$ . Clearly, the use of the neighborhood search does not improve the effectiveness of permutation GOMEA within the given computational time budget. Apparently the extensive search already executed by the Gene-pool Optimal Mixing process does not benefit anymore from the classical swap neighborhood exploration.

## 5 Permutation GOMEA vs. VNS4 Iterated Local Search

The previous section showed that permutation GOMEA can best be enhanced by seeding the initial population with solutions constructed with the LR heuristic. Adding local search to improve each solution after the gene-pool mixing process does not result in consistent improvements on all instances, and is therefore



**Table 2.** Quality of pGOMEA and VNS4 on Taillard instances.

	Best	pGOM	VNS4	Best	pGOM	VNS4	Best	pGOM	VNS4
50 × 5	64803	<b>0.47</b>	0.54	87207	1.26	<b>1.23</b>	125831	<b>0.78</b>	1.06
	68062	<b>0.63</b>	0.63	82820	<b>0.79</b>	1.42	119259	<b>0.56</b>	0.94
	63162	<b>0.92</b>	1.02	79987	<b>0.84</b>	1.07	116459	<b>0.71</b>	1.11
	68226	0.85	<b>0.84</b>	86581	<b>0.71</b>	0.99	120712	<b>0.92</b>	0.96
	69392	<b>0.65</b>	<b>0.65</b>	86450	<b>0.48</b>	1.15	118184	1.25	<b>1.16</b>
	66841	<b>0.65</b>	0.71	86637	<b>0.97</b>	1.00	120703	<b>0.82</b>	1.04
	66253	<b>0.60</b>	0.69	88866	<b>0.63</b>	1.07	122962	<b>1.01</b>	1.03
	64359	<b>0.52</b>	0.77	86824	1.18	<b>1.09</b>	122489	<b>1.02</b>	1.03
	62981	0.69	<b>0.62</b>	85526	<b>1.07</b>	1.24	121872	<b>0.99</b>	1.06
68853	0.93	<b>0.84</b>	88077	<b>0.62</b>	1.11	124064	<b>0.90</b>	1.06	
100 × 5	253713	<b>0.80</b>	0.94	299431	<b>1.02</b>	1.43	367267	1.63	<b>1.57</b>
	242777	<b>1.04</b>	1.12	274593	<b>1.55</b>	1.77	374032	<b>1.41</b>	1.50
	238180	<b>0.57</b>	0.94	288630	<b>1.15</b>	1.54	371417	<b>1.47</b>	1.56
	227889	<b>0.82</b>	1.01	302105	<b>1.55</b>	1.59	373822	<b>1.54</b>	1.75
	240589	<b>0.77</b>	0.95	285340	<b>1.09</b>	1.19	370459	1.58	<b>1.50</b>
	232936	<b>0.82</b>	1.12	270817	<b>1.14</b>	1.56	372768	<b>1.60</b>	1.66
	240669	<b>0.85</b>	0.90	280649	<b>1.21</b>	1.30	374483	<b>1.30</b>	1.71
	231428	<b>1.00</b>	1.06	291665	<b>0.81</b>	1.52	385456	<b>1.39</b>	1.56
	248481	1.00	<b>0.93</b>	302624	<b>1.20</b>	1.44	376063	<b>1.43</b>	1.58
243360	<b>0.87</b>	0.88	292230	<b>1.02</b>	1.58	379899	<b>1.45</b>	1.68	

not applied in this section. To see how well permutation GOMEA performs in comparison with a well tested Iterated Local Search heuristic for the PFSP, we compare it with VNS4, a Variable Neighborhood Search algorithm which uses an optimal form of combining the insertion heuristic and swap heuristic in order to solve the PFSP with the TFT criterion [4]. VNS4 was the most successful algorithm in a study of six different ways to combine the two most used neighborhoods in the literature used for the permutation flowshop scheduling problem with total flowtime criterion, namely job interchange and job insertion. VNS4 turned out to be the most effective of the six variable neighborhood search algorithms. VNS4 was also compared to a state-of-the-art evolutionary approach which it outperformed on most of the benchmark instances.

VNS4 is started from a solution generated by the LR constructive heuristic. First, VNS4 fully explores the job interchange neighborhood until no further improvement is possible. Then, a single iteration of the job insertion neighborhood search is executed. If this iteration improves the current solution, the algorithm resumes the interchange neighborhood search. When a local optimum common to both neighborhoods has been reached within the computational time limit, VNS4 executes a random walk to escape from the region of attraction of this local optimum. The random walk consists of  $k$  random job insertion moves. Iterated Local Search is sensitive to the length of the perturbation size. Experimental results show that VNS4’s performance degrades when the perturbation

**Table 3.** Quality of pGOMEA and VNS4 on structured instances.

		Job correlated			Machine Correlated			Mixed correlated		
		Best	pGOM	VNS4	Best	pGOM	VNS4	Best	pGOM	VNS4
$\alpha = 0.2$	354483	<b>0.42</b>	0.46	376876	0.28	<b>0.18</b>	392925	<b>0.35</b>	0.48	
	352436	<b>0.38</b>	0.58	379781	<b>0.31</b>	0.42	375810	<b>0.26</b>	0.66	
	354530	<b>0.34</b>	0.43	395682	<b>0.18</b>	0.42	400652	<b>0.14</b>	0.36	
	341043	<b>0.26</b>	0.52	389305	<b>0.36</b>	0.58	376562	<b>0.31</b>	0.67	
	354807	<b>0.31</b>	0.69	389688	<b>0.25</b>	0.59	368258	<b>0.29</b>	<b>0.29</b>	
	370144	<b>0.33</b>	0.54	360776	<b>0.15</b>	0.54	388375	<b>0.33</b>	0.63	
	351676	<b>0.32</b>	0.62	389651	<b>0.33</b>	0.59	379751	0.62	<b>0.38</b>	
	372308	<b>0.31</b>	0.39	377689	0.38	<b>0.35</b>	372021	<b>0.18</b>	0.29	
	362577	<b>0.43</b>	0.48	385017	<b>0.29</b>	0.40	364454	<b>0.29</b>	0.29	
	360703	<b>0.39</b>	0.57	389271	<b>0.25</b>	0.35	349791	0.28	0.52	
$\alpha = 0.4$	338303	<b>0.27</b>	0.39	432943	0.25	<b>0.20</b>	409404	<b>0.15</b>	0.23	
	331602	<b>0.23</b>	0.51	404417	0.20	<b>0.19</b>	408221	<b>0.11</b>	<b>0.11</b>	
	337449	<b>0.14</b>	0.23	411893	0.10	<b>0.09</b>	381029	<b>0.07</b>	0.15	
	343512	<b>0.15</b>	0.49	410707	0.26	<b>0.16</b>	367183	<b>0.18</b>	0.30	
	321656	<b>0.28</b>	0.41	437327	0.20	<b>0.12</b>	392645	<b>0.12</b>	0.33	
	348395	<b>0.23</b>	0.53	418578	0.23	<b>0.16</b>	428664	<b>0.06</b>	0.10	
	350807	<b>0.26</b>	0.28	402707	0.14	<b>0.13</b>	400714	0.15	<b>0.12</b>	
	336867	<b>0.11</b>	0.47	419103	<b>0.14</b>	<b>0.14</b>	378965	<b>0.16</b>	0.28	
	334469	<b>0.16</b>	0.44	440173	0.21	<b>0.15</b>	411678	<b>0.06</b>	0.15	
	341423	<b>0.35</b>	0.78	417045	0.13	<b>0.08</b>	392508	<b>0.10</b>	0.22	
$\alpha = 0.6$	319476	<b>0.17</b>	0.32	491858	<b>0.11</b>	0.12	395660	<b>0.03</b>	0.06	
	335018	<b>0.13</b>	0.30	407094	0.14	<b>0.05</b>	454738	<b>0.02</b>	0.02	
	311021	<b>0.06</b>	0.22	450581	<b>0.04</b>	0.02	365546	<b>0.02</b>	<b>0.02</b>	
	301670	<b>0.16</b>	0.37	391007	0.14	<b>0.06</b>	400760	<b>0.00</b>	0.00	
	303487	<b>0.10</b>	0.21	449246	0.13	<b>0.03</b>	447317	<b>0.01</b>	0.03	
	299529	<b>0.12</b>	0.22	479164	<b>0.12</b>	0.14	412359	<b>0.02</b>	0.05	
	320155	<b>0.14</b>	0.17	469897	0.41	<b>0.10</b>	450060	<b>0.02</b>	<b>0.02</b>	
	290573	<b>0.15</b>	0.27	471761	0.18	<b>0.06</b>	484204	<b>0.01</b>	0.02	
	305220	<b>0.12</b>	0.26	463159	0.14	<b>0.10</b>	414879	<b>0.02</b>	<b>0.02</b>	
	328872	<b>0.15</b>	0.27	453227	0.10	<b>0.06</b>	480739	<b>0.02</b>	<b>0.02</b>	

size is less than 14 or greater than 18 random job insertion moves [4]. The results with  $14 \leq k \leq 18$  produce very similar results, but  $k = 14$  has the lowest RPD median, so this value is shown here in the Tables with experimental results.

Table 2 shows the MRPD values on Taillard problem instances for VNS4 and permutation GOMEA when both algorithms are run for  $400 \cdot J \cdot M$  milliseconds. This stopping criterion is the same as used in recent works of [5, 8, 11] which were all included in the comparison in [4].

The best solution in the Table is marked bold and if the other solution performs significantly worse, its cell is marked grey. The results show that in most cases permutation GOMEA outperforms VNS4 significantly, in a number of cases there is no statistically significant difference, and in only a few instances VNS4 outperforms permutation GOMEA.

Secondly, we have tested permutation GOMEA and VNS4 on multiple structured instances with size  $100 \times 20$ . For these problems we have run the algorithms for  $400 \cdot (1 - \alpha) \cdot J \cdot M$  seconds, as structure makes the problems easier. Table 3 shows the results for three types of structured instances and three  $\alpha$  values.

The results show for job-correlated instances that permutation GOMEA always outperforms the VNS4 algorithm. The type of structure apparently suits permutation GOMEA best, while VNS4 cannot benefit from an easier fitness landscape. The machine-correlated instances with a high amount of structure ( $\alpha \geq 0.4$ ) are however easier for VNS4. When machine and job correlation are mixed, the PFSP is best solved using permutation GOMEA. Permutation GOMEA finds solutions with MRPD values lower than 0.5, showing that structured instances are easier than the standard Taillard instances.

An interesting question is why permutation GOMEA does not outperform VNS4 for the machine-correlated instances with a high amount of structure? Apparently, permutation GOMEA does not fully capture the structure in the machine-related instances. The most likely explanation is that this structure is not represented well enough in the distance measure used to build the linkage tree. Further research into the relation between the structure in specific problem instances and the type of structure searched for by GOMEA using different distance measures is needed to answer this question.

## 6 Conclusions

Previous work has shown how the Gene-pool Optimal Mixing Evolutionary Algorithm can be applied to permutation problems like the PFSP by representing solutions with the random-key encoding. Each generation GOMEA builds a linkage tree in order to capture structure in the set of solutions. This linkage tree can also be looked upon as an adaptive neighborhood learned by GOMEA to explore new solutions. In this paper we have investigated how the use of constructive heuristics and neighborhood search might improve on the Black-Box approach of permutation GOMEA. Results showed that adding neighborhood search does not consistently improve the performance. However, seeding the initial population of GOMEA by solutions generated by the constructive LR heuristic was shown to be an effective technique. We have experimentally compared permutation GOMEA - seeded with the constructive heuristic LR - with the highly successful VNS4 algorithm for unstructured and structured Permutation Flowshop Scheduling problems. VNS4 is an Iterated Local Search algorithm using a variable neighborhood that combines the job insertion neighborhood with the job swap neighborhood.

For the unstructured Taillard instances, GOMEA almost always outperforms VNS4. Also for the job correlated structured instances and for the mixed job/machine correlated instances GOMEA outperforms VNS4. Only for machine correlated structured instances with a high amount of structure ( $\alpha \geq 0.4$ ), VNS4 outperforms permutation GOMEA.

As a general conclusion, this paper has shown that the use of a multi-solution constructive heuristic to seed the initial population of permutation GOMEA

leads to an effective model-based evolutionary algorithm. It has also been shown that adding neighborhood search algorithms does not always result in more efficient results given a fixed computational time budget.

## References

1. Aalvanger, G.: Incorporating domain knowledge in permutation gene-pool optimal mixing evolutionary algorithms. Master's thesis. Utrecht University, The Netherlands (2017). <https://dspace.library.uu.nl/handle/1874/353005>
2. Bosman, P.A., Luong, N.H., Thierens, D.: Expanding from discrete Cartesian to permutation gene-pool optimal mixing evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 637–644. ACM (2016)
3. Ceberio, J., Irurozki, E., Mendiburu, A., Lozano, J.A.: Extending distance-based ranking models in estimation of distribution algorithms. In: 2014 IEEE Congress on Evolutionary Computation, CEC, pp. 2459–2466, July 2014
4. Costa, W.E., Goldbarg, M.C., Goldbarg, E.G.: New VNS heuristic for total flowtime flowshop scheduling problem. *Expert Syst. Appl.* **39**(9), 8149–8161 (2012)
5. Jarboui, B., Eddaly, M., Siarry, P.: An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. *Comput. Oper. Res.* **36**, 2638–2646 (2009)
6. Liu, J., Reeves, C.R.: Constructive and composite heuristic solutions to the p|| $\sum C_i$  scheduling problem. *EJOR* **132**(2), 439–452 (2001)
7. Taillard, E.: Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.* **64**(2), 278–285 (1993)
8. Tasgetiren, M.F., Pan, Q.-K., Suganthan, P.N., Chen, A.H.-L.: A discrete artificial bee colony algorithm for the permutation flow shop scheduling problem with total flowtime criterion. In: Proceedings of the IEEE World Congress on Computational Intelligence, WCCI-2010, pp. 137–144. IEEE (2010)
9. Thierens, D., Bosman, P.A.: Optimal mixing evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 617–624 (2011)
10. Watson, J.-P., Barbulescu, L., Whitley, L.D., Howe, A.E.: Contrasting structured and random permutation flow-shop scheduling problems. *INFORMS J. Comput.* **14**(2), 98–123 (2002)
11. Xu, X., Xu, Z., Gu, X.: An asynchronous genetic local search algorithm for the permutation flowshop scheduling problem with total flowtime minimization. *Expert Syst. Appl.* **38**, 7970–7979 (2011)



# On the Performance of Baseline Evolutionary Algorithms on the Dynamic Knapsack Problem

Vahid Roostapour<sup>(✉)</sup>, Aneta Neumann, and Frank Neumann

Optimisation and Logistics, School of Computer Science,  
The University of Adelaide, Adelaide, Australia  
[vahid.roostapour@adelaide.edu.au](mailto:vahid.roostapour@adelaide.edu.au)

**Abstract.** Evolutionary algorithms are bio-inspired algorithms that can easily adapt to changing environments. In this paper, we study single- and multi-objective baseline evolutionary algorithms for the classical knapsack problem where the capacity of the knapsack varies over time. We establish different benchmark scenarios where the capacity changes every  $\tau$  iterations according to a uniform or normal distribution. Our experimental investigations analyze the behavior of our algorithms in terms of the magnitude of changes determined by parameters of the chosen distribution, the frequency determined by  $\tau$  and the class of knapsack instance under consideration. Our results show that the multi-objective approaches using a population that caters for dynamic changes have a clear advantage on many benchmarks scenarios when the frequency of changes is not too high.

## 1 Introduction

Evolutionary algorithms [1] have been widely applied to a wide range of combinatorial optimization problems. They often provide good solutions to complex problems without a large design effort. Furthermore, evolutionary algorithms and other bio-inspired computing have been applied to many dynamic and stochastic problems [2, 3] as they have the ability to easily adapt to changing environments.

Most studies for dynamic problems so far focus on dynamic fitness functions [4]. However, in real-world applications the optimization goal, such as maximizing profit or minimizing costs, often does not change. Instead, resources to achieve this goal change over time and influence the quality of solutions that can be obtained. In the context of continuous optimization, dynamically changing constraints have been investigated in [2, 5]. Theoretical investigations for combinatorial optimization problems with dynamically changing constraints have recently been carried out [6, 7]. The goal of this paper is to contribute to this research direction from an experimental perspective.

In this paper, we investigate evolutionary algorithms for the knapsack problem where the capacity of the knapsack changes dynamically. We design a benchmark set for the dynamic knapsack problem. This benchmark set builds on classical static knapsack instances and varies the constraint bound over time. The

change in the constraint bound is done randomly every  $\tau$  iterations, where  $\tau$  is a parameter determining the frequency of changes. The magnitude of a change is either chosen according to a uniform distribution in an interval  $[-r, r]$ , where  $r$  determines the magnitude of changes. Furthermore, we examine changes according to the normal distribution  $\mathcal{N}(0, \sigma^2)$  with mean 0 and standard deviation  $\sigma$ . Here  $\sigma$  is used to determine the magnitude of changes and large values of  $\sigma$  make larger changes more likely. We investigate different approaches analyzed theoretically with respect to their runtime behavior in [7]. The algorithms that we consider are a classical (1+1) EA and multi-objective approaches that are able to store infeasible solutions as part of the population in addition to feasible solutions. Furthermore, the range of feasible and infeasible solutions stored in the multi-objective algorithms can be set based on the anticipated change of the constraint bound.

In our experimental investigations, we start by examining the knapsack problem where all weights are set to one and vary the constraint bound. This matches the setting of the optimization of a linear function with a dynamic uniform constraint analyzed in [7]. Our experimental results match the theoretical ones obtained in this paper and show that the multi-objective approaches using a population to cater for dynamic changes significantly reduce the offline error that occurred during the run of the algorithms. For the general setting, we investigate different classes of knapsack problem, such as with uniformly chosen weights and profits and bounded strongly correlated instances. We examine the behaviour of the algorithms while varying the frequency and magnitude of changes. Our results show that the (1+1) EA has an advantage over the multi-objective algorithms when the frequency of changes is high. In this case, the population of the multi-objective approaches is slower to adapt to the changes that occur. On the other hand, a lower frequency of changes plays in favor of the multi-objective approaches, if the weights and profits are not correlated to make the instances particularly difficult to solve.

The outline of the paper is as follows: Sect. 2 introduces the problem definition and three algorithms we studied; the dynamic knapsack problem and experimental setting is presented in Sect. 3; in Sect. 4 we analyze the experimental results in detail, and a conclusion follows in Sect. 5.

## 2 Preliminaries

In this section, we define the Knapsack Problem (KP) and further notations used in the rest of this paper. We present (1+1) EA and two multi-objective algorithms called MOEA and MOEA\_D that are considered in this paper.

### 2.1 Problem Definition

We investigate the performance of different evolutionary algorithms on the KP under dynamic constraint. There are  $n$  items with profits  $\{p_1, \dots, p_n\}$  and weights  $\{w_1, \dots, w_n\}$ . A solution  $x$  is a bit string of  $\{0, 1\}^n$  which has the overall

---

**Algorithm 1.** (1+1) EA

---

```

1  $x \leftarrow$  previous best solution;
2 while stopping criterion not met do
3    $y \leftarrow$  flip each bit of  $x$  independently with probability of  $\frac{1}{n}$ ;
4   if  $f_{1+1}(y) \geq f_{1+1}(x)$  then
5      $x \leftarrow y$ ;

```

---

weight  $w(x) = \sum_{i=1}^n w_i x_i$  and profit  $p(x) = \sum_{i=1}^n p_i x_i$ . The goal is to compute a solution  $x^* = \arg \max \{p(x) \mid x \in \{0, 1\}^n \wedge w(x) \leq C\}$  of maximal profit which has weight at most  $C$ .

We consider two types of this problem based on the consideration of the weights. Firstly, we assume that all the weights are one and uniform dynamic constraint is applied. In this case, the limitation is on the number of items chosen for each solution and the optimal solution is to pick  $C$  items with the highest profits. Next, we consider the general case where the profits and weights are linear integers under linear constraint on the weight.

## 2.2 Algorithms

We investigate the performance of three algorithms in this paper. The initial solution for all these algorithms is a solution with items chosen uniformly at random. After a dynamic change to constraint  $C$  happens, all the algorithms update the solution(s) and start the optimization process with the new capacity. This update is addressing the issue that after a dynamic change, current solutions may become infeasible or the distance of its weight from the new capacity become such that it is not worth to be kept anymore. (1+1) EA (Algorithm 1) flips each bit of the current solution with the probability of  $\frac{1}{n}$  as the mutation step. Afterward, the algorithm chooses between the original solution and the mutated one using the value of the fitness function. Let  $p_{max} = \max_{i=1}^n p_i$  be the maximum profit among all the items. The fitness function that we use in (1+1) EA is as follows:

$$f_{1+1}(x) = p(x) - (n \cdot p_{max} + 1) \cdot \nu(x)$$

where  $\nu(x) = \max \{0, w(x) - C\}$  is the constraint violation of  $x$ . If  $x$  is a feasible solution, then  $w(x) \leq C$  and  $\nu(x) = 0$ . Otherwise,  $\nu(x)$  is the weight distance of  $w(x)$  from  $C$ .

The algorithm aims to maximize  $f_{1+1}$  which consists of two terms. The first term is the total profit of the chosen items and the second term is the applied penalty to infeasible solutions. The amount of penalty guarantees that a feasible solution always dominates an infeasible solution. Moreover, between two infeasible solutions, the one with weight closer to  $C$  dominates the other one.

The other algorithm we consider in this paper is a multi-objective evolutionary algorithm (Algorithm 2), which is inspired by a theoretical study

---

**Algorithm 2.** MOEA

---

```

1 Update  $C$ ;
2  $S^+ \leftarrow \{z \in S^+ \cup S^- \mid C < w(z) \leq C + \delta\}$ ;
3  $S^- \leftarrow \{z \in S^+ \cup S^- \mid C - \delta \leq w(z) \leq C\}$ ;
4 if  $S^+ \cup S^- = \emptyset$  then
5    $q \leftarrow$  best previous solution;
6 if  $C < w(q) \leq C + \delta$  then
7    $S^+ \leftarrow \{q\} \cup S^+$ ;
8 else if  $C - \delta \leq w(q) \leq C$  then
9    $S^- \leftarrow \{q\} \cup S^-$ ;
10 while a change happens do
11   if  $S^+ \cup S^- = \emptyset$  then
12     Initialize  $S^+$  and  $S^-$  by Repair( $q, \delta, C$ );
13   else
14     choose  $x \in S^+ \cup S^-$  uniformly at random;
15      $y \leftarrow$  flip each bit of  $x$  independently with probability  $\frac{1}{n}$ ;
16     if  $(C < w(y) \leq C + \delta) \wedge (\nexists p \in S^+ : p \succ_{MOEA} y)$  then
17        $S^+ \leftarrow (S^+ \cup \{y\}) \setminus \{z \in S^+ \mid y \succ_{MOEA} z\}$ ;
18     if  $(C - \delta \leq w(y) \leq C) \wedge (\nexists p \in S^- : p \succ_{MOEA} y)$  then
19        $S^- \leftarrow (S^- \cup \{y\}) \setminus \{z \in S^- \mid y \succ_{MOEA} z\}$ ;

```

---

on the performance of evolutionary algorithms in the reoptimization of linear functions under dynamic uniform constraints [7]. Each solution  $x$  in the objective space is a two-dimensional point  $f_{MOEA}(x) = (w(x), p(x))$ . We say solution  $y$  dominates solution  $x$  w.r.t.  $f_{MOEA}$ , denoted by  $y \succ_{MOEA} x$ , if  $w(y) = w(x) \wedge f_{(1+1)}(y) \geq f_{(1+1)}(x)$ .

According to the definition of  $\succ_{MOEA}$ , two solutions are comparable only if they have the same weight. Note that if  $x$  and  $y$  are infeasible and comparable, then the one with higher profit dominates. MOEA uses a parameter denoted by  $\delta$ , which determines the maximum number of individuals that the algorithm is allowed to store around the current  $C$ . For any weight in  $[C - \delta, C + \delta]$ , MOEA keeps a solution. The algorithm prepares for the dynamic changes by storing nearby solutions, even if they are infeasible as they may become feasible after the next change. A large  $\delta$ , however, causes a large number of solutions to be kept, which reduces the probability of choosing anyone. Since the algorithm chooses only one solution to mutate in each iteration, this affects the MOEA's performance in finding the optimal solution.

After each dynamic change, MOEA updates the sets of solutions. If a change occurs such that all the current stored solutions are outside of the storing range, namely  $[C - \delta, C + \delta]$ , then the algorithm consider the previous best solution as the initial solution and uses the Repair function (Algorithm 3), which behaves similar to (1+1) EA, until a solution with weight distance  $\delta$  from  $C$  is found.



---

**Algorithm 3.** Repair

---

**input** : Initial solution  $q$ ,  $\delta$ ,  $C$   
**output**:  $S^+$  and  $S^-$  such that  $|S^+ \cup S^-| = 1$

```

1 while  $|S^+ \cup S^-| = 0$  do
2    $y \leftarrow$  flip each bit of  $q$  independently with probability of  $\frac{1}{n}$ ;
3   if  $f_{1+1}(y) \geq f_{1+1}(q)$  then
4      $q \leftarrow y$ ;
5     if  $C < w(q) \leq C + \delta$  then
6        $S^+ \leftarrow \{q\} \cup S^+$ ;
7     else if  $C - \delta \leq w(q) \leq C$  then
8        $S^- \leftarrow \{q\} \cup S^-$ ;

```

---



---

**Algorithm 4.** MOEA\_D (Dominance and Selection)

---

```

14 choose  $x \in S^+ \cup S^-$  uniformly at random;
15  $y \leftarrow$  flip each bit of  $x$  independently with probability  $\frac{1}{n}$ ;
16 if  $(C < w(y) \leq C + \delta) \wedge (\nexists p \in S^+ : p \succ_{MOEA\_D} y)$  then
17    $S^+ \leftarrow (S^+ \cup \{y\}) \setminus \{z \in S^+ | y \succ_{MOEA\_D} z\}$ ;
18 if  $(C - \delta \leq w(y) \leq C) \wedge (\nexists p \in S^- : p \succ_{MOEA\_D} y)$  then
19    $S^- \leftarrow (S^- \cup \{y\}) \setminus \{z \in S^- | y \succ_{MOEA\_D} z\}$ ;

```

---

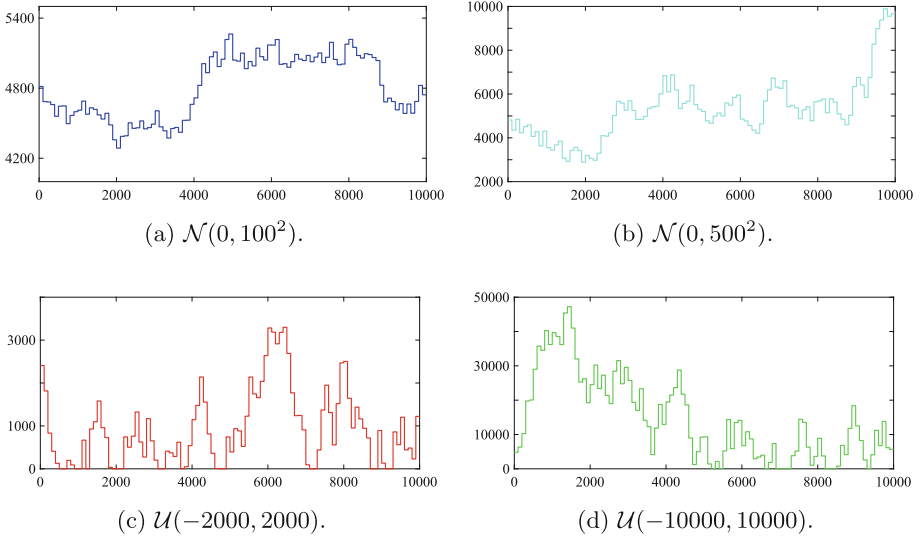
To address the slow rate of improvement of MOEA caused by a large  $\delta$ , we defined a new dominance procedure. We use the standard definition of dominance in multi-objective optimization and say that solution  $y$  dominates solution  $x$ , denoted by  $\succ_{MOEA\_D}$ , if  $w(y) \leq w(x) \wedge p(y) \geq p(x)$ . This new algorithm, called MOEA\_D, is obtained by replacing lines 14–19 of Algorithm 2 with Algorithm 4. It should be noticed that if  $y$  is an infeasible solution then it is only compared with other infeasible solutions and if  $y$  is feasible it is only compared with other feasible solutions. MOEA\_D keeps fewer solutions than MOEA and overall the quality of the kept solutions is higher, since they are not-dominated by any other solution in the population.

### 3 Benchmarking for the Dynamic Knapsack Problem

In the following section, the dynamic version of KP used for the experiments is described, and we explain how the dynamic changes occur during the optimization process. In addition, the dynamic benchmarks and the experimental settings are presented.

#### 3.1 The Dynamic Knapsack Problem

In the dynamic version of KP considered in this paper, the capacity dynamically changes during the optimization with a preset frequency factor denoted by  $\tau$ .



**Fig. 1.** Examples for constraint bound  $C$  over 10000 generations with  $\tau = 100$  using uniform and normal distributions. Initial value  $C = 4815$ .

A change happens every  $\tau$  generations, i.e., the algorithm has  $\tau$  generations to find the optimum of the current capacity and to prepare for the next change. In the case of uniformly random alterations, the capacity of next interval is achieved by adding a uniformly random value in  $[-r, r]$  to  $C$ . Moreover, we consider another case in which the amount of the changes is chosen from the Gaussian distribution  $\mathcal{N}(0, \sigma^2)$ . Figure 1 illustrates how dynamic changes from different distributions affect the capacity. Note that the scales of the subfigures are not the same. For example, the total change after 100 dynamic changes under  $\mathcal{N}(0, 100^2)$  is less than 1000 (Fig. 1a) while the capacity reached almost 45000 with dynamic changes under  $\mathcal{U}(-10000, 10000)$  (Fig. 1d). This indicates that there are different types of challenges, resulting from the dynamic changes that the algorithms must consider.

The combination of different distributions and frequencies brings interesting challenges for the algorithms. In an environment where the constraint changes with a high frequency, the algorithms have less time to find the optimal solution, hence, it is likely that an algorithm which tries to improve only one solution will perform better than another algorithm that needs to optimize among several solutions. Furthermore, the uniform distribution guarantees upper and lower bounds on the magnitude of the changes. This property could be beneficial for the algorithms which keep a number of solutions in each generation, which they do get ready and react faster after a dynamic change. If the changes happen under a normal distribution, however, there is no strict bound on the value of any particular change, which means it is not easy to predict which algorithms will perform better in this type of environment.

### 3.2 Benchmark and Experimental Setting

In this experiment we use eli101 benchmarks, which were originally generated for Traveling Thief Problem [8], ignoring the cities and only using the items. The weights and profits are generated in three different classes. In Uncorrelated (uncorr) instances, the weights and profits are integers chosen uniformly at random within  $[1, 1000]$ . Uncorrelated Similar Weights (unc-s-w) instances have uniformly distributed random integers as the weights and profits within  $[1000, 1010]$  and  $[1, 1000]$ , respectively. Finally, there is the Bounded Strongly Correlated (bou-s-c) variations which result in the hardest instances and comes from the bounded knapsack problem. The weights of this instance are chosen uniformly at random within  $[1, 1000]$  and the profits are set according to the weights within the weights plus 100. In addition, in Sect. 4.1, where the weights are one, we set all the weights to one and consider the profits as they are in the benchmarks. The initial capacity in this version is calculated by dividing the original capacity by the average of the profits. Dynamic changes add a value to  $C$  each  $\tau$  generations. Four different situations in terms of frequencies are considered: high frequent changes with  $\tau = 100$ , medium frequent changes with  $\tau = 1000$ ,  $\tau = 5000$  and low frequent changes with  $\tau = 15000$ .

In the case that weights are 1, the value of dynamic changes are chosen uniformly at random within the interval  $[-r, r]$ , where  $r = 1$  or  $r = 10$ . In the case of linear weights, when changes are uniformly random, we investigate two values for  $r$ :  $r = 2000, 10000$ . Also, changes from normal distribution is experimented for  $\sigma = 100, \sigma = 500$ .

We use the offline errors to compute the performance of the algorithms. In each generation, we record error  $e_i = p(x_i^*) - p(x_i)$  where  $x_i^*$  and  $x_i$  are the optimal solution and the best achieved feasible solution in generation  $i$ , respectively. If the best achieved solution is infeasible, then we have  $e_i = C - w(x)$ , which is negative. The final error for  $m$  generations would be  $\sum_{i=1}^m e_i/m$ .

The benchmarks for dynamic changes are thirty different files. Each file consists of 100000 changes, as numbers in  $[-r, r]$  generated uniformly at random. Similarly, there are thirty other files with 100000 numbers generated under the normal distribution  $\mathcal{N}(0, \sigma^2)$ . The algorithms start from the beginning of each file and pick the number of change values from the files. Hence, for each setting, we run the algorithms thirty times with different dynamic change values and record the total offline error of each run.

In order to establish a statistical comparison of the results among different algorithms, we use a multiple comparisons test. In particularity, we focus on the method that compares a set of algorithms. For statistical validation we use the Kruskal-Wallis test with 95% confidence. Afterwards, we apply the Bonferroni post-hoc statistical procedures that are used for multiple comparisons of a control algorithm against two or more other algorithms. For more detailed descriptions of the statistical tests we refer the reader to [9].

Our results are summarized in the Tables 1, 2 and 3. The columns represent the algorithms (1+1) EA, MOEA, MOEA\_D, with the corresponding mean value and standard deviation. Note,  $X^{(+)}$  is equivalent to the statement that the

algorithm in the column outperformed algorithm  $X$ , and  $X^{(-)}$  is equivalent to the statement that  $X$  outperformed the algorithm in the given column. If the algorithm  $X$  does not appear, this means that no significant difference was observed between the algorithms.

## 4 Experimental Results

In this section we describe the initial settings of the algorithms and analyze their performance using the mentioned statistical tests. The initial solution for all the algorithms is a pack of items which are chosen uniformly at random. Each algorithm initially runs for 10000 generations without any dynamic change. After this, the first change is introduced, and the algorithms run one million further generations with dynamic changes in every  $\tau$  generations. For the multi-objective algorithms, it is necessary to initially provide a value for  $\delta$ . These algorithms keep at most  $\delta$  feasible solutions and  $\delta$  infeasible solutions, to help them efficiently deal with a dynamic change. When the dynamic changes come from  $\mathcal{U}(-r, r)$ , it is known that the capacity will change at most  $r$ . Hence, we set  $\delta = r$ . In case of changes from  $\mathcal{N}(0, \sigma^2)$ ,  $\delta$  is set to  $2\sigma$ , since 95% of values will be within  $2\sigma$  of the mean value. Note that a larger  $\delta$  value increases the population size of the algorithms and there is a trade-off between the size of the population and the speed of algorithm in reacting to the next change.

### 4.1 Dynamic Uniform Constraint

In this section, we validate the theoretical results against the performance of (1+1) EA and Multi-Objective Evolutionary Algorithm. Shi et al. [7] state that the multi-objective approach performs better than (1+1) EA in reoptimizing the optimal solution of dynamic KP under uniform constraint. Although the MOEA that we used in this experiment is not identical to the multi-objective algorithm studied previously by Shi et al. [7] and they only considered the reoptimization time, the experiments show that multi-objective approaches outperform (1+1) EA in the case of uniform constraints (Table 1). An important reason for this remarkable performance is the relation between optimal solutions in different weights. In this type of constraint, the difference between the optimal solution of weight  $w$  and  $w + 1$  is one item. As a result of this, keeping non-dominated solutions near the constrained bound helps the algorithm to find the current optimum more efficiently and react faster after a dynamic change.

Furthermore, according to the results, there is no significant difference between using MOEA and MOEA\_D in this type of KP. Considering the experiments in Sect. 4.2, a possible reason is that the size of population in MOEA remains small when weights are one. Hence, MOEA\_D, which stores fewer items because of its dominance definition, has no advantage in this manner anymore. In addition, the constraint is actually on the number of the items. Thus, both definitions for dominance result the same in many cases.

**Table 1.** The mean, standard deviation values and statistical tests of the offline error for (1+1) EA, MOEA, MOEA\_D based on the uniform distribution with all the weights as one.

	n	r	$\tau$	(1+1) EA (1)			MOEA (2)			MOEA_D (3)		
				Mean	St	Stat	Mean	St	Stat	Mean	St	Stat
uncor	100	5	100	4889.39	144.42	$2^{(-)}, 3^{(-)}$	1530.00	120.76	$1^{(+)}$	1486.85	123.00	$1^{(+)}$
	100	5	1000	1194.23	86.52	$2^{(-)}, 3^{(-)}$	44.75	8.96	$1^{(+)}$	46.69	8.51	$1^{(+)}$
uncor-s-w	100	5	100	4990.80	144.87	$2^{(-)}, 3^{(-)}$	1545.36	115.15	$1^{(+)}$	1500.07	106.70	$1^{(+)}$
	100	5	1000	1160.23	130.32	$2^{(-)}, 3^{(-)}$	41.90	6.13	$1^{(+)}$	43.06	7.22	$1^{(+)}$
bou-s-c	100	5	100	13021.98	780.76	$2^{(-)}, 3^{(-)}$	4258.53	580.77	$1^{(+)}$	4190.55	573.13	$1^{(+)}$
	100	5	1000	3874.76	911.50	$2^{(-)}, 3^{(-)}$	177.62	83.16	$1^{(+)}$	175.14	80.73	$1^{(+)}$

## 4.2 Dynamic Linear Constraint

In this section, we consider the same algorithms in more difficult environments where weights are arbitrary under dynamic linear constraint. As it is shown in Sect. 4.1, the multi-objective approaches outperform (1+1) EA in the case that weights are one. Now we try to answer the question: Does the relationship between the algorithms hold when the weights are arbitrary?

The data in Table 2 shows the experimental results in the case of dynamic linear constraints and changes under a uniform distribution. It can be observed that (as expected) the mean of errors decreases as  $\tau$  increases. Larger  $\tau$  values give more time to the algorithm to get closer to the optimal solution. Moreover, starting from a solution which is near to the optimal for the previous capacity, can help to speed up the process of finding the new optimal solution in many cases.

We first consider the results of dynamic changes under the uniform distribution. We observe in Table 2 that unlike with uniform constraint, in almost all the settings, MOEA has the worst performance of all the algorithms. The first reason for this might be that items selected in optimal solutions with close weights are also close in terms of Hamming distance. In other words, when weights are one, we can achieve the optimal solution for weight  $w$  by adding an item to the optimal solution for weight  $w - 1$  or by deleting an item from the optimal solution for  $w + 1$ . However, in case of arbitrary weights, the optimal solutions of weight  $w$  and  $w + d$  could have completely different items, even if  $d$  is small. Another reason could be the effect of having a large population. A large population may cause the optimization process to take longer and it could get worse because of the definition of  $\succ_{MOEA}$ , which only compares solutions with equal weights. If  $s$  is a new solution and there is no solution with  $w(s)$  in the set of existing solutions, MOEA keeps  $s$  whether  $s$  is a good solution or not, i.e., regardless of whether it is really a non-dominated solution or whether it would be dominated by other solutions in the set. This comparison also does not consider if  $s$  has any good properties to be inherited by the next generation. Moreover, putting  $s$  in

the set of solutions decreases the probability of choosing any other solution, even those solutions that are very close to the optimal solution. As it can be seen in the Table 2, however, there is only one case in which MOEA beat the (1+1) EA: when the weights are similar, and the magnitude of changes are small (2000), which means the population size is also small (in comparison to 10000), and finally  $\tau$  is at its maximum to let the MOEA to use its population to optimize the problem.

Although MOEA does not perform very well in instances with general weights, the multi-objective approach with a better defined dominance, MOEA\_D, does outperform (1+1) EA in many cases. We compare the performance of (1+1) EA and MOEA\_D below.

**Table 2.** The mean, standard deviation values and statistical tests of the offline error for (1+1) EA, MOEA, MOEA\_D based on the uniform distribution.

	n	r	$\tau$	(1+1) EA (1)			MOEA (2)			MOEA_D (3)		
				mean	st	stat	mean	st	stat	mean	st	stat
uncor	100	2000	100	5564.37	463.39	2(+),3(-)	11386.40	769.77	1(-),3(-)	3684.26	525.50	1(+),2(+)
	100	2000	1000	2365.56	403.64	2(+),3(-)	7219.17	587.50	1(-),3(-)	776.14	334.69	1(+),2(+)
	100	2000	5000	1415.42	167.08	2(+),3(-)	3598.29	420.12	1(-),3(-)	270.90	121.43	1(+),2(+)
	100	2000	15000	914.55	102.82	2(+),3(-)	2004.16	368.82	1(-),3(-)	88.80	43.98	1(+),2(+)
unc-s-w	100	2000	100	3128.43	188.36	2(+),3(-)	5911.11	534.24	1(-),3(-)	2106.45	249.28	1(+),2(+)
	100	2000	1000	606.14	99.23	2(+),3(-)	1564.23	619.97	1(-),3(-)	302.34	24.60	1(+),2(+)
	100	2000	5000	147.55	31.80	3(-)	174.23	95.98	3(-)	60.94	9.12	1(+),2(+)
	100	2000	15000	64.65	17.13	2(-),3(-)	40.66	15.51	1(+),3(-)	19.26	4.04	1(+),2(+)
bou-s-c	100	2000	100	3271.07	266.54	2(+)	5583.53	337.81	1(-),3(-)	3036.97	297.33	2(+)
	100	2000	1000	1483.01	85.14	2(+),3(-)	2639.16	106.47	1(-),3(-)	617.92	186.35	1(+),2(+)
	100	2000	5000	796.77	89.80	2(+),3(-)	1256.62	118.27	1(-),3(-)	251.41	109.58	1(+),2(+)
	100	2000	15000	538.45	66.98	2(+),3(-)	687.95	116.91	1(-),3(-)	104.27	61.06	1(+),2(+)
uncor	100	10000	100	10256.72	210.51	2(+),3(+)	16278.97	248.43	1(-),3(-)	11038.07	236.91	1(-),2(+)
	100	10000	1000	3604.18	285.73	2(+)	13340.20	704.32	1(-),3(-)	3508.51	473.42	2(+)
	100	10000	5000	1607.78	278.60	2(+),3(-)	10614.45	1660.32	1(-),3(-)	1183.52	411.83	1(+),2(+)
	100	10000	15000	987.64	219.53	2(+),3(-)	8006.35	1612.20	1(-),3(-)	566.69	219.54	1(+),2(+)
unc-s-w	100	10000	100	7192.82	153.93	2(+),3(+)	12617.69	318.23	1(-),3(-)	8057.44	274.17	1(-),2(+)
	100	10000	1000	1846.43	115.23	2(+)	6981.81	768.78	1(-),3(-)	1743.12	364.38	2(+)
	100	10000	5000	539.39	65.39	2(+)	3488.28	819.51	1(-),3(-)	519.63	75.22	2(+)
	100	10000	15000	208.73	36.91	2(+)	1525.23	306.72	1(-),3(-)	201.97	79.28	2(+)
bou-s-c	100	10000	100	7187.80	122.59	2(+),3(+)	15111.38	231.53	1(-),3(-)	12736.55	229.48	1(-),2(+)
	100	10000	1000	2282.81	219.24	2(+),3(+)	8301.43	569.90	1(-),3(-)	3575.26	550.54	1(-),2(+)
	100	10000	5000	1370.48	250.59	2(+)	5248.40	1045.78	1(-),3(-)	1472.19	493.88	2(+)
	100	10000	15000	955.38	133.33	2(+)	3852.07	752.84	1(-),3(-)	977.41	397.75	2(+)

When changes are smaller, it can be seen in Table 2 that the mean of offline errors of MOEA\_D is smaller than (1+1) EA. The dominance of MOEA\_D is such that only keeps the dominant solutions. When a new solution is found, the algorithm removes solutions that are dominated by it and keeps it only if it is not dominated by the any other one. This process improves the quality of the solutions by increasing the probability of keeping a solution beneficial to future generations. Moreover, it reduces the size of the population significantly. Large changes to the capacity, however, makes the MOEA\_D keep more individuals, and it is in this circumstance that (1+1) EA may perform better than MOEA\_D.

When  $r = 10000$ , MOEA\_D does not have significantly better results in all cases unlike in the case of  $r = 2000$ , and in most of the situations it performs as well as (1+1) EA. In all high frequency conditions where  $\tau = 100$ , the

(1+1) EA has better performance. It may be caused by MOEA\_D needing more time to optimize a population with a larger size. Moreover, when the magnitude of changes is large, it is more likely that a new change will force MOEA\_D to remove all of its stored individuals and start from scratch.

We now study the experimental results that came from considering the dynamic changes under the normal distribution (Table 3). The results confirm that (1+1) EA is faster with more frequent changes. Skipping the case with uncorrelated similar weights and frequent changes, MOEA\_D has always been the best algorithm in terms of performance and MOEA has been the worst.

**Table 3.** The mean, standard deviation values and statistical tests of the offline error for (1+1) EA, MOEA, MOEA\_D based on the normal distribution.

	n	$\sigma$	$\tau$	(1+1) EA (1)			MOEA (2)			MOEA_D (3)		
				mean	st	stat	mean	st	stat	mean	st	stat
uncor	100	100	100	2714.72	106.06	2(+),3(+)	9016.83	2392.48	1(-),3(-)	4271.09	789.94	1(-),2(+)
	100	100	1000	1386.66	97.11	2(+),3(-)	3714.89	737.11	1(-),3(-)	412.89	27.25	1(+),2(+)
	100	100	5000	801.54	73.67	2(+),3(-)	1266.35	119.25	1(-),3(-)	108.28	14.22	1(+),2(+)
	100	100	15000	549.71	78.98	2(+),3(-)	749.86	148.03	1(-),3(-)	61.93	17.03	1(+),2(+)
unc-s-w	100	100	100	412.24	111.07	2(+),3(+)	1979.65	914.35	1(-)	1904.09	877.55	1(-)
	100	100	1000	85.55	23.13	2(+),3(+)	1566.54	409.32	1(-)	1482.37	391.75	1(-)
	100	100	5000	36.94	13.61	2(+),3(+)	1414.66	448.78	1(-)	1322.35	414.27	1(-)
	100	100	15000	29.14	19.70	2(+),3(+)	1237.67	665.27	1(-)	1137.80	648.73	1(-)
bou-s-c	100	100	100	1491.36	260.72	2(+),3(+)	4625.49	1302.52	1(-),3(-)	2903.77	717.92	1(-),2(+)
	100	100	1000	736.10	53.99	2(+),3(-)	1748.61	189.94	1(-),3(-)	312.88	35.52	1(+),2(+)
	100	100	5000	446.94	39.36	2(+),3(-)	640.60	91.29	1(-),3(-)	101.21	17.47	1(+),2(+)
	100	100	15000	337.85	40.44	2(+),3(-)	469.16	93.99	1(-),3(-)	70.16	22.26	1(+),2(+)
uncor	100	500	100	13400.88	305.14	2(+),3(+)	46395.44	4565.61	1(-),3(-)	19218.94	1035.72	1(-),2(+)
	100	500	1000	6363.16	194.59	2(+),3(-)	25747.08	1181.11	1(-),3(-)	2387.61	151.73	1(+),2(+)
	100	500	5000	3983.06	254.38	2(+),3(-)	18004.03	1243.66	1(-),3(-)	1467.58	152.77	1(+),2(+)
	100	500	15000	3112.73	315.29	2(+),3(-)	17610.35	1265.50	1(-),3(-)	1348.25	194.71	1(+),2(+)
unc-s-w	100	500	100	2845.31	146.80	2(+),3(+)	11803.99	1256.99	1(-)	11438.04	1247.62	1(-)
	100	500	1000	595.70	86.19	2(+),3(+)	8851.36	1488.59	1(-)	8478.21	1313.55	1(-)
	100	500	5000	222.79	62.22	2(+),3(+)	7025.45	2639.34	1(-)	6488.47	2335.92	1(-)
	100	500	15000	171.33	50.28	2(+),3(+)	7188.67	4184.84	1(-)	6278.10	4146.54	1(-)
bou-s-c	100	500	100	7444.23	290.00	2(+),3(+)	24462.58	1330.93	1(-),3(-)	15592.66	791.70	1(-),2(+)
	100	500	1000	4062.63	210.49	2(+),3(-)	12291.63	589.18	1(-),3(-)	2781.20	317.88	1(+),2(+)
	100	500	5000	3013.35	289.29	2(+),3(-)	9667.96	571.34	1(-),3(-)	1971.56	220.63	1(+),2(+)
	100	500	15000	2722.29	342.39	2(+),3(-)	9308.28	719.25	1(-),3(-)	1760.51	251.51	1(+),2(+)

The most notable results occur in the case with uncorrelated similar weights. (1+1) EA outperforms both other algorithms in this instance. This happens because of the value of  $\delta$  and the weights of the instances.  $\delta$  is set to  $2\sigma$  in the multi-objective approaches and the weights of items are integers in  $[1001, 1010]$  in this type of instance. (1+1) EA is able to freely get closer to the optimal solutions from both directions, while the multi-objective approaches are only allowed to consider solutions in range of  $[C - \delta, C + \delta]$ . In other words, it is possible that there is only one solution in that range or even no solution. Hence, multi-objective approaches have no advantage in this type of instances according to the value of  $\delta$  and weights of the items, and in fact, may have a disadvantage.

## 5 Conclusions and Future Work

In this paper we studied the evolutionary algorithms for the KP where the capacity dynamically changes during the optimization process. In the introduced

dynamic setting, the frequency of changes is determined by  $\tau$ . The magnitude of changes is chosen randomly either under the uniform distribution  $\mathcal{U}(-r, r)$  or under the normal distribution  $\mathcal{N}(0, \sigma^2)$ . We compared the performance of (1+1) EA and two multi-objective approaches with different dominance definitions (MOEA, MOEA\_D). Our experiments in the case of weights set to one verified the previous theoretical studies for (1+1) EA and MOEA [7]. It is shown that the multi-objective approach, which uses a population in the optimization, outperforms (1+1) EA. In addition, we considered the algorithms in the case of general weights for different classes of instances with a variation of frequencies and magnitudes. Our results illustrated that MOEA does not perform well in the general case due to its dominance procedure. However, MOEA\_D, which benefits from a population with a smaller size and non-dominated solutions, beats (1+1) EA in most cases. On the other hand, in the environments with highly frequent changes, (1+1) EA performs better than the multi-objective approaches. In such cases, the population slows down MOEA\_D in reacting to the dynamic change.

**Acknowledgment.** This work has been supported through Australian Research Council (ARC) grant DP160102401.

## References

1. Eiben, A., Smith, J.: Introduction to Evolutionary Computing, 2nd edn. Springer, Heidelberg (2007). <https://doi.org/10.1007/978-3-662-44874-8>
2. Nguyen, T., Yao, X.: Continuous dynamic constrained optimization: the challenges. *IEEE Trans. Evol. Comput.* **16**(6), 769–786 (2012)
3. Rakshit, P., Konar, A., Das, S.: Noisy evolutionary optimization algorithms - a comprehensive survey. *Swarm Evol. Comput.* **33**, 18–45 (2017)
4. Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: a survey of the state of the art. *Swarm Evol. Comput.* **6**, 1–24 (2012)
5. Ameca-Alducin, M.-Y., Hasani-Shoreh, M., Neumann, F.: On the use of repair methods in differential evolution for dynamic constrained optimization. In: Sim, K., Kaufmann, P. (eds.) *EvoApplications 2018*. LNCS, vol. 10784, pp. 832–847. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77538-8\\_55](https://doi.org/10.1007/978-3-319-77538-8_55)
6. Pourhassan, M., Gao, W., Neumann, F.: Maintaining 2-approximations for the dynamic vertex cover problem using evolutionary algorithms. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 903–910. ACM (2015)
7. Shi, F., Schirneck, M., Friedrich, T., Kötzing, T., Neumann, F.: Reoptimization times of evolutionary algorithms on linear functions under dynamic uniform constraints. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1407–1414. ACM (2017)
8. Polyakovskiy, S., Bonyadi, M.R., Wagner, M., Michalewicz, Z., Neumann, F.: A comprehensive benchmark set and heuristics for the traveling thief problem. In: *Proceedings of Conference on Genetic and Evolutionary Computation*, pp. 477–484. ACM (2014)
9. Corder, G.W., Foreman, D.I.: *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Wiley, Hoboken (2009)





# On the Synthesis of Perturbative Heuristics for Multiple Combinatorial Optimisation Domains

Christopher Stone<sup>(✉)</sup>, Emma Hart, and Ben Paechter

School of Computing, Edinburgh Napier University, Scotland, UK  
{c.stone,e.hart,b.paechter}@napier.ac.uk

**Abstract.** Hyper-heuristic frameworks, although intended to be cross-domain at the highest level, rely on a set of domain-specific low-level heuristics at lower levels. For some domains, there is a lack of available heuristics, while for novel problems, no heuristics might exist. We address this issue by introducing a novel method, applicable in multiple domains, that constructs new low-level heuristics for a domain. The method uses grammatical evolution to construct iterated local search heuristics: it can be considered cross-domain in that the *same* grammar can evolve heuristics in multiple domains without requiring any modification, assuming that solutions are represented in the same form. We evaluate the method using benchmarks from the travelling-salesman (TSP) and multi-dimensional knapsack (MKP) domain. Comparison to existing methods demonstrates that the approach generates low-level heuristics that outperform heuristic methods for TSP and are competitive for MKP.

## 1 Introduction

The *hyper-heuristic* method was first introduced in an attempt to raise the generality at which search methodologies operate [2]. One of the main motivations was to produce a method that was cheaper to implement and easier to use than problem specific special purpose methods, while producing solutions of acceptable quality to an end-user in an appropriate time-frame. Specifically, it aimed to address a concern that the practical impact of search-based optimisation techniques in commercial and industrial organisations had not been as great as might have been expected, due to the prevalence of problem-specific or knowledge-intensive techniques, which were inaccessible to the non-expert or expensive to implement.

The canonical hyper-heuristic framework introduces a domain barrier that separates a general algorithm to choose heuristics from a set of low-level heuristics. The low-level heuristics are specific to a particular domain, and may be designed by hand, relying on intuition or human-expertise [2], or can be evolved by methods such as Genetic Programming [14]. The success of the high-level heuristic is strongly influenced by the number and the quality of the low-level heuristics available. Given a new problem domain that does not map well to

well-studied domains in the literature, it can be challenging to find a suitable set of low-level heuristics to utilise with a hyper-heuristic. Although this can be addressed through evolving new heuristics [1], this process requires in-depth understanding of the problem and effort designing a specialist algorithm to evolve the heuristic. We propose to address this by introducing a method of creating new heuristics that is *cross-domain*, that is, the method can be used without modification to create heuristics in multiple domains, assuming a common problem representation.

As a step towards raising the generality of creating *low-level* heuristics, we focus on domains that can be mapped to a graph-based representation. This includes obvious applications such as routing and scheduling [14], as well as many less obvious ones including packing problems [11] and utility maximisation in complex negotiations [12]. We describe a novel method using grammatical evolution that produces a set of local-search heuristics for solving travelling-salesperson (TSP) problems, and another for multi-dimensional knapsack (MKP) problems. In each case, an identical grammar is used to evolve heuristics that modifies a permutation representing a TSP or MKP problem. The grammar is trained on a small subset of randomly generated instances in each case and shown to produce competitive results on benchmarks when compared to human design heuristics and almost as good as specially design meta-heuristics.

This research lays the foundation for a paradigm shift in designing heuristics for combinatorial optimisation domains in which no heuristics currently exist, or those domains in which hyper-heuristic methods would benefit from additional low-level heuristics. The approach significantly reduces the burden on human experts, as it only requires that the problem can be represented as a graph, with no further specialisation, and does not require a large database of training examples. The contributions are threefold: (1) it describes a novel grammar that generates mutation operators that perturb a permutation via partial permutations and inversions; (2) the grammar is trained to produce single instances of new ‘move’ operators using a *very small set* of randomly generated instances from each problem domain; (3) it demonstrates that competitive results can be obtained from a generic grammar, even when using a representation that is not necessarily considered the most natural for a domain.

## 2 Background

Hyper-Heuristics are class of algorithms that explore the space of heuristics rather than the space of solutions, and have found application in a broad range of combinatorial optimisation domains [2]. As previously mentioned, the core idea is to create a *generic* algorithm that selects and applies heuristics, separated by a domain-barrier from a subset of low-level domain-specific heuristics. Most initial work focused on development of the generic controlling algorithms [2]. More recent attention has focused on the role of the low-level heuristics themselves. Low-level heuristics fall into two categories [2]. *Constructive* heuristics build a solution from scratch, adding an element at a time, e.g. [14]. On the other hand,

*perturbative* heuristics modify an existing solution, e.g. re-ordering elements in a permutation [4] or modifying genes [2].

In many practical domains, hand-designed low-level heuristics are readily available, e.g. [2]. However, a tranche of research has focused on generation of *new* heuristics, typically using methods from Genetic Programming [1], Grammatical Evolution [8, 13] and Memetic Algorithms [6]. Specifically in the domain of *perturbative* heuristics, GP approaches to generating novel local search heuristics for satisfiability testing were proposed by [2]. Grammatical Evolution is applied to evolve new local-search heuristics for 1d-bin packing in [2, 7]. It is also worth mentioning the progress made in cross-domain optimisation thanks to HyFlex [9]: however, note that here the controlling hyper-heuristics are cross-domain but the framework still relies on pools of domain *specific* low-level heuristics.

Despite some success in the areas just described, we note that in each case, the function and terminal nodes used in GP or the grammar specification in GE are specifically tailored to a single domain. While clearly specialisation is likely to be beneficial, it can require significant expertise and investment in algorithm design. For a practitioner, such knowledge is unlikely to be available, and for new domains, this may be time-consuming even for an expert. Therefore, we are motivated to design a general-purpose method that is capable—without modification—of producing heuristics in multiple domains. While we do not expect such a generator to compete with specialised heuristics or meta-heuristics, we evaluate whether the approach can be used as a “quick and dirty” method of generating a heuristic that produces an acceptable quality solution in multiple domains.

### 3 Method

Our generator makes use of Grammatical Evolution [10] for the production of new heuristics. In particular we specify *one* grammar and this single grammar is used to produce heuristics in two different domains. Our method can be described by three fundamental steps:

- Represent the problem-domain of interest as an ordering problem.
- Use Grammatical Evolution to breed heuristics that perturb the order of a solution, using a small training set of examples. The new heuristics are evaluated according their effectiveness as a mutation operator in an iterated local-search algorithm.
- Re-use the evolved heuristics on unseen instances from the same domain.

#### 3.1 Grammatical Evolution

Grammatical Evolution (GE) is a population based evolutionary computation approach used to construct sequence of symbols in an arbitrary language defined by a BNF grammar. A BNF Grammar consist of a set of *production rules* composed of terminal and non-terminal nodes. The production rules are used to

substitute the non-terminal nodes with other nodes, which can be both non-terminal or terminal nodes, repeatedly until a whole sequence of terminal nodes is composed. Each non terminal node has its own set of production rules. *Codons* (represented as a single integer) specify which specific production rule should be chosen at each step.

We use GE to evolve a Python program that takes a sequence (i.e a permutation) as an input and returns a modified version of the same sequence (permutation) with the same length. Our implementation uses the GE library described by Fenton *et al.* [5]. This version of GE proved to be accessible, straightforward to reuse, and is the most recent version of GE. A detailed description of the complete implementation can be found in [5]. The code is also open-source and available on *github*<sup>1</sup>. The main implementation details relevant to this work are as follows:

**Genome:** Fenton’s implementation uses a linear genome representation that is encoded as a list of integers (codons). The mapping between the genotype and the phenotype is actuated by the use of the modulus operator on the value of the codon, i.e.  $Selectednode = c \bmod n$ , where  $c$  is the integer value of the codon to be mapped and  $n$  is the number of options available in the specific production rule.

**Mutation:** An integer flip at the level of the codons is used. One of the codons that has been used for the phenotype is changed each iteration and substituted with a completely new codon.

**Crossover:** Variable one-point crossover, where the crossing point between 2 individuals is chosen randomly.

**Replacement:** Generational replacement strategy with elitism 1, i.e one genome is guaranteed to stay in the pool on the next generation.

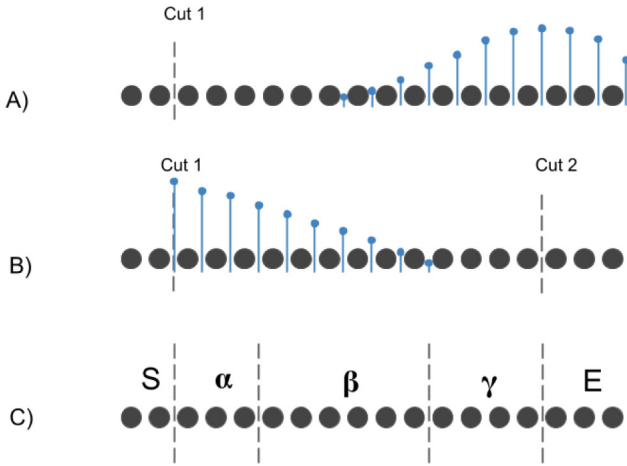
### 3.2 Grammar and Mechanics of the Operator

The operator constructed by our grammar can be thought of as a form of  $k$ -opt, that is configurable and includes extra functions to determine where to break a sequence. The formulation and implementation is vertex centric instead of edge centric. The mechanics of the algorithm are as follows:

**Number of Cuts:** This determines in how many places a sequence will be cut creating  $(k - 1)$  subsequences where  $k$  is the number of cuts. The number of possible loci of the cuts is equal to  $n + 1$ , where  $n$  is the number of vertices (the sequence can be cut both before the first element and after the last element).

**Location of Cuts:** The grammar associates a strategy to each cut that will determine the location of the specific cut. A strategy may contain a reference location such as the ends of the sequence or subsequence, a specific place in the sequences or a random location. The reference can be used together with

<sup>1</sup> <https://github.com/PonyGE/PonyGE2>.



**Fig. 1.** (A) Example of a sequence with one cut and a probability mass function that will decide the loci of the second cut. (B) Both cuts now shown (C) final set of subsequences after  $k$ -cuts

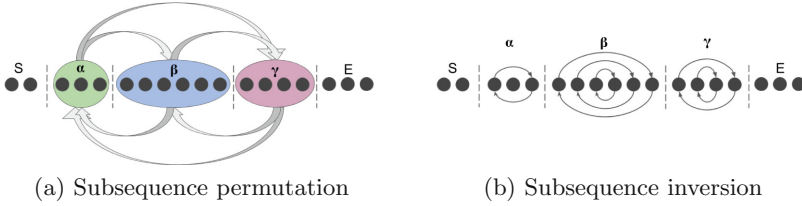
a probability distribution that determines the chances of any given location to be the place of the next cut. These probability distributions *de facto* regulate the length of each subsequence. Two probability distributions can be selected by the grammar: a discretised triangular distribution and a negative binomial distribution. An example can be seen in Fig. 1A and B.

After the cutting phase the subsequences are given symbols with S being always the leftmost subsequence and E being the rightmost subsequence such as in Fig. 1C. The start and end sequences ( $S, E$ ) are never altered by the evolved operator which only acts on the sequences labelled  $\alpha$ - $\beta$  in Fig. 1C. Note that subsequences may be empty. This can happen if the leftmost cut is on the left of the first element (leaving  $S$  empty), if the rightmost cut is after the last element (leaving  $E$  empty) or if two different cuts are applied in the same place.

**Permutation of the Subsequence:** After cutting the sequence the subsequences becomes the units of a new sequence. The grammar can specify if the subsequence will be reordered to a specific permutation (including the identity, i.e no change) or to a random permutation. An example can be seen in Fig. 2a.

**Inversion of the Subsequences:** The grammar specifies whether the order of each specific subsequence should be reversed or if the reversing should be decided randomly for each subsequence each iteration.

**Iteration Effect:** Another component of the grammar is the iteration effect which may associate a specific function that regulate the change in the initial cutting location at each iteration. We have specified four types of effect: *random*, which means that the starting location of the first cut will be random; *oscillate* that makes the starting position move in a wave like manner and returns to the



**Fig. 2.** Example perturbations of the subsequences produced by the grammar

initial loci after a number of iterations; *step* simply moves one step on the right of the previous starting position and finally *none* which has no effect.

### 3.3 Problem Domains and Training Examples

We apply the grammar in two problem domains. The *Travelling Salesman Problem* (TSP) is one of the most studied problems in combinatorial optimisation, in which a tour passing by all points must be *minimised*. Due to the fact that it is naturally encoded as an ordering problem represented by a permutation it plays the role of *base case* for our experiments.

The *Multidimensional Knapsack Problem* (MKP) is another of the most studied problem in combinatorial optimisation with applications in budgeting, packing and cutting problems. In this case the profit from items selected among a collection must be *maximised* while respecting the constraints of the knapsack. This problem is chosen as in its typical form, it is not represented as ordering problem. However, a formulation based on chains and graphs was recently introduced in [15]. The goal here is to demonstrate that the approach can produce acceptable heuristics from a generic representation, without requiring the expert knowledge required to formulate a problem-specific approach.

A set of heuristics is evolved in each domain, using a set of example training instances in each case. It is well known that having better training instances leads to better outcomes [2]. However, as the ultimate goal of this work is produce a system that can produce acceptable heuristics in an unknown domain in which good training examples might not be available (or in an existing domain in which we cannot predict characteristics of future problems) we synthesise a random set of training instances in each case. Parameters of the synthesisers are given in Table 1. 5 TSP instances are synthesised using a uniform random distribution. Each instance has 100 cities placed in a 2D Euclidean plane. For MKP, each of 5 instances has 100 objects with 10 constraints. Each constraint is a sample from a uniform random distribution between 0 and 100. The profits of each object are taken from a normal distribution with mean equal to the sum of the constraints and standard deviation 50. The constraints of the knapsack are sampled from a normal distribution with mean 2500 and standard deviation 300. We recognise that real-instances are unlikely to be uniformly distributed; our implementation therefore represents the worst-case scenario in which the system can be evolved.

<op>	→	addCut(<loci_ref>,<distance>) <c> <ExtraCuts> Iteration_effect(<motion>,<loci_computation>) permutation(<perm_behaviour>) inversions(<inv_behaviour>)
<ExtraCuts>	→	∅   <c>   <c><c>   <c><c><c>
<c>	→	addCut(<loci_ref>,<distance>) isInverted(<invert>) permutationFactor(<r>)
<motion>	→	'random'   'oscillate'   'steps'   'none'
<loci_ref>	→	'none'   'left'   'right'   'limit'
<distance>	→	'linear'   'negative_binomial',(<r>,<p>)
<r>	→	1   2   3   4   5   6   7   8   9   10
<p>	→	0.1   0.2   0.3   0.4   0.5   0.6   0.7
<loci_computation>	→	'once'   'always'
<perm_behaviour>	→	'fixed'   'random'
<inv_behaviour>	→	'fixed'   'random'
<invert>	→	0   1

Fig. 3. Grammar used to produce the local search operator

## 4 Experiments

**Training Phase:** One-point local-search heuristics are generated using an off-line learning approach. The system is applied *separately* to each domain, but uses an *identical* grammar in both. At each iteration of the GE, each heuristic in the population is applied within a hill-climbing algorithm to each of the 5 training instances starting from an randomly initialised solution. The hill-climber runs for  $x$  iterations with an improvement only acceptance criteria. For TSP,  $x = 1000$  and for MKP,  $x = 2500$  (based on initial experimentation). The fitness at the end-point is averaged over the 5 instances and assigned to the heuristic (i.e. distance for TSP and profit for MKP). Experiments are repeated in each domain 10 times, with a new set of 5 problems generated for each run. The best performing heuristic from each run is retained, creating an ensemble of 10 heuristics as a result. All the parameters of the synthesisers are give in Table 1a while the GE parameters are in Table 1b.

**Testing Phase:** The generated ensemble is tested on benchmark instances from the literature. For TSP, we use 19 problems taken from the TSPLib. MKP heuristics are tested on at total of 54 problems from 6 benchmark datasets from the OR-library. Each of the 10 heuristics is applied 5 times to each problem for  $10^5$  iterations, starting from a randomly initialised solution, using an improvement only acceptance criteria (hill-climber). We record the average performance of each heuristic over 5 runs, as well as the best, and the worst.

For TSP, we compare the results with 50 runs per instance of a classic two opt algorithm<sup>2</sup>, chosen as a commonly used example of high-performing local-search heuristic. For MKP, the vast majority of published results use meta-heuristic approaches. We compare with two approaches from [3], the Chaotic Binary Particle Swarm Optimisation with Time Varying Acceleration Coefficient (CBPSO),

<sup>2</sup> Using the R package TSPLIB.

and an improved version of this algorithm that includes a self-adaptive check and repair operator (SACRO CBPSO), the most recent and highest-performing methods in MKP optimisation. Both algorithms use problem specific knowledge: a penalty function in the former, and a utility ratio estimation function in the latter, with a binary representation for their solution. Both are allocated a considerably larger evaluation budget than our experiments. The heuristics evolved using our approach would not be expected to outperform these approaches—however, we wish to investigate whether the approach can produce solutions within reasonable range of known optima that would be acceptable to a practitioner requiring a quick solution.

**Table 1.** Experimental parameters

Parameter	Value	Parameter	Value
Number of cities	100	Generations	80
Cities distribution type	Uniform	Population	100
Cities distribution range	0-100	Mutation	int flip
Number of objects	100	Crossover Prob.	0.80
Number of constraints	10	Crossover type	one point
Object constraints distribution	Uniform	Max initial tree	10
Object constraints range	0-100	Max tree depth	17
Object profit distribution	Normal	Replacement	generational
Object profit mean	Sum of constraints	Tournament size	2
Object profit deviation	50		
Knapsack constraints dist.	Normal		
Knapsack constraints mean	2500		
Knapsack constraints deviation	300		

(a) Problem synthesisers

(b) Grammatical Evolution

## 5 Results and Analysis

We refer to our algorithm as *HHGE* in all reported results. Table 3 shows the best, worst and median performance of the evolved heuristics and the two-opt based algorithm for TSP. With the exception of a single case, the evolved heuristics perform better in term of best, worst and median results. For each instance, we apply a Wilcoxon Rank-sum test on the 50 pairs of samples, and provide a p-value in the rightmost column. Improvements are statistically significant at the 5% level in all cases.

Results for MKP are reported in Table 2, averaged over 10 heuristics in each case. Note that despite the simplistic nature of our approach—a hill-climber with an evolved mutation operator—our approach out-performs CBSPO in 22 out of



**Table 2.** Generated heuristics vs specialised meta-heuristics from [3]. Highlighted values for HHGE indicate where it outperforms CBPSO. SACRO-BPSO performs best in all instances

Instance	HHGE				Optima	CBPSO		SACRO-BPSO	
	Best	Worst	Average	Median		Best	Average	Best	Average
hp1	3418	3385	<b>3410.56</b>	3418	3418	3418	3403.9	3418	<i>3413.38</i>
hp2	3186	2997	3171.54	3186	3186	3186	3173.61	3186	<i>3184.74</i>
pb1	3090	3057	<b>3083.32</b>	3090	3090	3090	3079.74	3090	<i>3086.78</i>
pb2	3186	3114	<b>3179.88</b>	3186	3186	3186	3171.55	3186	<i>3186</i>
pb4	95168	90961	93515.54	93897	95168	95168	94863.67	95168	<i>95168</i>
pb5	2139	2085	<b>2120.06</b>	2130.5	2139	2139	2135.6	2139	<i>2139</i>
pb6	776	641	733.12	735.5	776	776	758.26	776	<i>776</i>
pb7	1035	983	1018.9	1025	1035	1035	1021.95	1035	<i>1035</i>
pet2	87061	78574	85409.32	87061	87061	-	-	-	-
pet3	4015	3165	3955.8	4015	4015	-	-	-	-
pet4	6120	5440	6040.2	6110	6120	-	-	-	-
pet5	12400	12090	12363.1	12400	12400	-	-	-	-
pet6	10618	10107	10592.1	10604	10618	-	-	-	-
pet7	16537	15683	16504.48	16537	16537	-	-	-	-
sent01	7772	7491	<b>7706.92</b>	7749.5	7772	7772	7635.72	7772	<i>7769.48</i>
sent02	8722	8614	<b>8691.02</b>	8704	8722	8722	8668.47	8722	<i>8722</i>
weing1	141278	135673	140619.36	141278	141278	141278	141226.8	141278	<i>141278</i>
weing2	130883	118035	128542.94	130712	130883	130883	130759.8	130883	<i>130883</i>
weing3	95677	77897	93099.5	94908	95677	95677	95503.93	95677	<i>95676.39</i>
weing4	119337	100734	117811.56	119337	119337	119337	119294.2	119337	<i>119337</i>
weing5	98796	78155	95912	98475.5	98796	98796	98710.4	98796	<i>98796</i>
weing6	130623	117715	129452.56	130233	130623	130623	130531.3	130623	<i>130623</i>
weing7	1095382	1088277	<b>1093583.14</b>	1093595	1095445	1095382	1084172	1095382	<i>1094349</i>
weing8	624319	525663	<b>606175.12</b>	613070	624319	624319	597190.6	624319	<i>622079.9</i>
weish01	4554	4298	4494.34	4530	4554	4554	4548.55	4554	<i>4554</i>
weish02	4536	4164	4485.12	4536	4536	4536	4531.88	4536	<i>4536</i>
weish03	4115	3707	3963.08	3985	4115	4115	4105.79	4115	<i>4115</i>
weish04	4561	3921	4385.5	4455	4561	4561	4552.41	4561	<i>4561</i>
weish05	4514	3754	4265.56	4479.5	4514	4514	4505.89	4514	<i>4514</i>
weish06	5557	5238	5503.16	5538	5557	5557	5533.79	5557	<i>5553.75</i>
weish07	5567	5230	5496.56	5542	5567	5567	5547.83	5567	<i>5567</i>
weish08	5605	5276	5534.82	5597.5	5605	5605	5596.16	5605	<i>5605</i>
weish09	5246	4626	5062.24	5128	5246	5246	5232.99	5246	<i>5246</i>
weish10	6339	5986	6244.82	6314	6339	6339	6271.84	6339	<i>6339</i>
weish11	5643	5192	5522.18	5631.5	5643	5643	5532.15	5643	<i>5643</i>
weish12	6339	5951	6217.14	6322.5	6339	6339	6231.5	6339	<i>6339</i>
weish13	6159	5780	6032.28	6056	6159	6159	6120.38	6159	<i>6159</i>
weish14	6954	6581	6827.9	6852	6954	6954	6837.77	6954	<i>6954</i>
weish15	7486	7113	<b>7391</b>	7445.5	7486	7486	7324.55	7486	<i>7486</i>
weish16	7289	6902	7154.82	7159.5	7289	7289	7288.7	7289	<i>7288.7</i>
weish17	8633	8506	<b>8609</b>	8633	8633	8633	8547.71	8633	<i>8633</i>
weish18	9580	9310	<b>9527</b>	9560.5	9580	9580	9480.86	9580	<i>9578.46</i>
weish19	7698	7272	7505.3	7527	7698	7698	7528.55	7698	<i>7698</i>
weish20	9450	9117	<b>9381.32</b>	9430	9450	9450	9332.11	9450	<i>9450</i>
weish21	9074	8655	<b>8972.9</b>	9025	9074	9074	8948.22	9074	<i>9074</i>
weish22	8947	8466	<b>8814.7</b>	8871	8947	8947	8774.2	8947	<i>8936.92</i>
weish23	8344	7809	<b>8202.06</b>	8217.5	8344	8344	8165	8344	<i>8344</i>
weish24	10220	9923	<b>10154.54</b>	10185.5	10220	10220	10106.28	10220	<i>10219.7</i>
weish25	9939	9667	<b>9872.48</b>	9909.5	9939	9939	9826.57	9939	<i>9939</i>
weish26	9584	9175	<b>9434.92</b>	9473	9584	9584	9313.87	9584	<i>9584</i>
weish27	9819	9244	<b>9652.3</b>	9671	9819	9819	9607.54	9819	<i>9819</i>
weish28	9492	8970	<b>9328.52</b>	9347.5	9492	9492	9123.26	9492	<i>9492</i>
weish29	9410	8794	<b>9217.28</b>	9279	9410	9410	9025.5	9410	<i>9410</i>
weish30	11191	10960	<b>11135.64</b>	11161	11191	11191	10987.21	11191	<i>11190.12</i>

**Table 3.** Comparison between evolved heuristics and classic two-opt. For each instance we compute the Wilcoxon Rank-sum test using 50 pairs of samples

	HHGE			2-opt			Ranksum p-value
	Best	Worst	Median	Best	Worst	Median	
<i>berlin52</i>	7793	8825	8170	7741	9388	8310	0.0033
<i>ch130</i>	6418	7108	6722	6488	7444	6984	0.0030
<i>d198</i>	16256	17033	16651	16400	18213	17291	$\ll 0.001$
<i>eil101</i>	674	739	702	680	749	709	0.0073
<i>eil51</i>	435	484	456	442	494	473	$\ll 0.001$
<i>eil76</i>	563	616	593	583	628	611	$\ll 0.001$
<i>kroA150</i>	28109	31473	29344	29223	31994	30509	$\ll 0.001$
<i>kroA200</i>	31470	34528	32634	31828	35170	32893	0.0005
<i>kroB150</i>	27028	30283	28767	28114	30941	29134	$\ll 0.001$
<i>kroB200</i>	31315	35319	33029	31509	35077	33422	0.0455
<i>kroC100</i>	21418	24353	22885	22953	25503	23977	$\ll 0.001$
<i>kroD100</i>	21817	24405	23233	22772	26428	23430	$\ll 0.001$
<i>kroE100</i>	22660	25509	24178	23012	26695	24216	0.0021
<i>lin105</i>	14675	16965	15642	14966	17057	16191	$\ll 0.001$
<i>pr107</i>	45547	50313	47560	47597	51932	50002	0.0001
<i>pr144</i>	58847	68722	61534	59058	67272	64660	0.0002
<i>pr152</i>	75615	81458	78073	77307	81850	79964	$\ll 0.001$
<i>pr226</i>	81811	96484	86244	83566	101582	91512	0.0021
<i>u159</i>	44826	51353	47461	45297	51505	48124	0.1276

54 instances when considering average performance<sup>3</sup>. SACRO-BPSO (currently the best available meta-heuristic) performs better across the board, as expected.

In Table 4 we compare the Average Success Rate (ASR) across all instances group by dataset against the results presented by [3] on 2 versions of SACRO algorithms and an additional fish-swarm method. In [3], ASR is calculated as the number of times the global optima was found for each instance divided by the number of trials. For HHGE, we define a trial as successful if at least one of the 10 heuristics found the optima in the trial, and repeat this 5 times. It can be seen that the results are comparable to those of specialised algorithms, and in fact outperform these methods on Weing and HP sets.

<sup>3</sup> We do not provide statistical significance information as the PSO results, which are reported directly from [3], use a population based approach and vastly different number of evaluations.

**Table 4.** Comparison with latest specialised meta-heuristics (PSO) from the literature: a fish-swarm algorithm IbaFSA and the two most recent SACRO algorithms, results taken directly from [3]

Problem Set	Instances	ASR			
		IbaFSA	BPSO-TVAC	CBPSO-TVAC	<b>HHGE</b>
Sento	2	1.000	0.9100	0.9100	0.90
Weing	8	0.7875	0.7825	0.7838	0.80
Weish	30	0.9844	0.9450	0.9520	0.907
Hp	2	0.9833	0.8000	0.8600	1.00
Pb	6	1.000	0.9617	0.9517	0.967
Pet	6	na	na	na	1.00

## 6 Conclusions

We have presented a method based on grammatical evolution for generating perturbative low-level heuristics for multiple problem domains that is cross-domain: the same grammar generates heuristics for a domain that can be represented as an ordering problem. The method was demonstrated on two specific domains, TSP (a natural ordering problem) and MKP. We have compared the synthesised heuristics with a specialised human-designed heuristic in the TSP domain where the synthesised heuristic outperformed the well-known 2-opt heuristic. In the MKP domain, we compared the generated heuristics against two of the latest specialised meta-heuristics. The heuristics outperform one of these methods, and are at least comparable to the best method. We also note that the ensemble of 10 generated heuristics demonstrate high success rates in finding known optima when each heuristic is applied several times.

The approach represents the first steps towards increasing the cross-domain nature of hyper-heuristics: current approaches tend to focus on the high-level hyper-heuristic as cross-domain, while relying on specialised low-level heuristics below the domain barrier. Our approach extends existing work by also making methods for the automated generation of low-level heuristics cross-domain, without requiring specialist human-expertise. The proposed approach is applicable to a subset of domains that can be represented as ordering problems. While we believe this subset is large, it clearly does not include all domains. However, the same approach could be generalised to develop a portfolio of modifiable grammars, each addressing a broad class of problems.

Recall that in each case, HHGE was trained using a very small, uniformly generated set of instances, and in the case of MKP, applied to a non-typical representation, yet still provides acceptable results. We believe this fits with the original intention of hyper-heuristics, i.e. to provide quick and acceptable solutions to a range of problems with minimal effort. Although specialised representations and large sets of specialised training instances undoubtedly have

their place in producing very high-quality results when required, these results demonstrate that a specialised representation is not *strictly* necessary and can be off-set by an appropriate move-operator.

## Reproducibility

The code used for the experiments and for the analysis of the results is available at <https://github.com/c-stone2099/HHGE-PPSN2018>.

## References

1. Bader-El-Den, M., Poli, R.: Generating SAT local-search heuristics using a GP hyper-heuristic framework. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., Lutton, E. (eds.) EA 2007. LNCS, vol. 4926, pp. 37–49. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-79305-2\\_4](https://doi.org/10.1007/978-3-540-79305-2_4)
2. Edmund, K., et al.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)
3. Chih, M.: Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem. Appl. Soft Comput. **26**, 378–389 (2015)
4. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44629-X\\_11](https://doi.org/10.1007/3-540-44629-X_11)
5. Fenton, M., McDermott, J., Fagan, D., Forstenlechner, S., Hemberg, E., O’Neill, M.: PonyGE2: grammatical evolution in python. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1194–1201. ACM (2017)
6. Krasnogor, N., Gustafson, S.: A study on the use of “self-generation” in memetic algorithms. Nat. Comput. **3**(1), 53–76 (2004)
7. Mascia, F., López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T.: From grammars to parameters: automatic iterated greedy design for the permutation flow-shop problem with weighted tardiness. In: Nicosia, G., Pardalos, P. (eds.) LION 2013. LNCS, vol. 7997, pp. 321–334. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-44973-4\\_36](https://doi.org/10.1007/978-3-642-44973-4_36)
8. Mascia, F., López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T.: Grammar-based generation of stochastic local search heuristics through automatic algorithm configuration tools. Comput. Oper. Res. **51**, 190–199 (2014)
9. Ochoa, G., et al.: HyFlex: a benchmark framework for cross-domain heuristic search. In: Hao, J.-K., Middendorf, M. (eds.) EvoCOP 2012. LNCS, vol. 7245, pp. 136–147. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29124-1\\_12](https://doi.org/10.1007/978-3-642-29124-1_12)
10. O’Neill, M., Ryan, C.: Grammatical evolution. IEEE Trans. Evol. Comput. **5**(4), 349–358 (2001)
11. Pferschy, U., Schauer, J.: The knapsack problem with conflict graphs. J. Graph Algorithms Appl. **13**(2), 233–249 (2009)
12. Robu, V., Somefun, D.J.A., La Poutré, J.A.: Modeling complex multi-issue negotiations using utility graphs. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 280–287. ACM (2005)

13. Sabar, N.R., Ayob, M., Kendall, G., Qu, R.: Grammatical evolution hyper-heuristic for combinatorial optimization problems. *Strategies* **3**, 4 (2012)
14. Sim, K., Hart, E.: A combined generative and selective hyper-heuristic for the vehicle routing problem. In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 1093–1100. ACM (2016)
15. Stone, C., Hart, E., Paechter, B.: Automatic generation of constructive heuristics for multiple types of combinatorial optimisation problems with grammatical evolution and geometric graphs. In: Sim, K., Kaufmann, P. (eds.) *EvoApplications 2018*. LNCS, vol. 10784, pp. 578–593. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77538-8\\_40](https://doi.org/10.1007/978-3-319-77538-8_40)

# **Genetic Programming**



# EDDA-V2 – An Improvement of the Evolutionary Demes Despeciation Algorithm

Illya Bakurov<sup>1</sup>, Leonardo Vanneschi<sup>1</sup>, Mauro Castelli<sup>1</sup> (✉),  
and Francesco Fontanella<sup>2</sup>

<sup>1</sup> NOVA Information Management School (NOVA IMS),  
Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisbon, Portugal  
{ibakurov,lvanneschi,mcastelli}@novaims.unl.pt

<sup>2</sup> Dipartimento di Ingegneria Elettrica e dell'Informazione (DIEI),  
Università di Cassino e del Lazio Meridionale, Cassino, FR, Italy  
fontanella@unicas.it

**Abstract.** For any population-based algorithm, the initialization of the population is a very important step. In Genetic Programming (GP), in particular, initialization is known to play a crucial role - traditionally, a wide variety of trees of various sizes and shapes are desirable. In this paper, we propose an advancement of a previously conceived Evolutionary Demes Despeciation Algorithm (EDDA), inspired by the biological phenomenon of demes despeciation. In the pioneer design of EDDA, the initial population is generated using the best individuals obtained from a set of independent subpopulations (demes), which are evolved for a few generations, by means of conceptually different evolutionary algorithms - some use standard syntax-based GP and others use a semantics-based GP system. The new technique we propose here (EDDA-V2), imposes more diverse evolutionary conditions - each deme evolves using a distinct random sample of training data instances and input features. Experimental results show that EDDA-V2 is a feasible initialization technique: populations converge towards solutions with comparable or even better generalization ability with respect to the ones initialized with EDDA, by using significantly reduced computational time.

**Keywords:** Initialization algorithm · Semantics · Despeciation

## 1 Introduction

Initialization of the population is the first step of Genetic Programming (GP). John Koza proposed three generative methods of the initial population - Grow, Full and Ramped Half-and-Half (RHH) [7]. All of them consist in constructing trees in an almost random fashion and vary only in the doctrine which guides the process. Since the RHH method is a mixture of both the Full and Grow methods, it allows the production of trees of various sizes and shapes and it was

frequently used in many applications. The emergence of new geometric semantic operators [8], which introduce semantic awareness into Genetic Programming (GP), strengthened the importance of semantics-awareness in GP. Semantics was considered as a fundamental factor for the success of the evolutionary search process, leading to the definition of initialization algorithms [2,3] that aimed at increasing semantic diversity in the initial GP population. These studies clearly showed the importance of semantics in this part of the evolutionary process. Other contributions had already recognized that an initial population characterized by a high diversity increases the effectiveness of GP, bestowing a wider exploration ability on the process [7,12]. With the aim of directly searching in the semantic space, Moraglio and colleagues introduced Geometric Semantic Genetic Programming (GSGP) [8], which rapidly raised an impressive interest in the GP community - in part, because of its interesting property of inducing a fitness landscape characterized by the absence of locally suboptimal solutions for any problem consisting in matching sets of input data into known targets [13]. In GSGP, standard crossover and mutation variation operators are replaced with so-called geometric semantic operators, that have precise effects on the semantics of the trees, from now on, called individuals. After the emergence of Geometric Semantic Operators (GSOs), a conceptually distinct sub-field aiming at investigating the properties of GSGP was born inside the GP community. As a result, new techniques were proposed to favor the search process of GSGP, making it more efficient. In the context of the initialization process, Pawlak and Krawiec introduced semantic geometric initialization [10] and Oliveira and colleagues introduced the concept of dispersion of solutions, for increasing the effectiveness of geometric semantic crossover [9]. Following this research track, in 2017, we proposed a new initialization method which mimics the evolution of demes, followed by despeciation, called Evolutionary Demes Despeciation Algorithm [14]. In summary, our idea consisted in seeding the initial population of  $N$  individuals with good quality individuals that have been evolved, for few a generations, in  $N$  independent subpopulations (demes), by means of conceptually different evolutionary algorithms. In this system,  $n\%$  of demes use standard GP and the remaining  $(100 - n)\%$  use GSGP. After evolving one deme, the best individual is extracted to seed the initial population in the Main Evolutionary Process (MEP). The work presented in this paper improves the previously proposed initialization method, EDDA, by including even more adverse evolutionary conditions in each deme. In summary, in our advancement of the EDDA method, which we will from now on call, EDDA-V2, every deme evolves using a distinct random sample of training data instances and input features.

This document is organized as follows: In Sect. 2 we recall basic concepts related to GSGP. Section 3 describes the previous and new EDDA variants, showing their differences. Section 4 presents the experimental study. Section 5 discusses the experimental results. Finally, Sect. 6 concludes the work summarizing its contribution.



## 2 Geometric Semantic Genetic Programming

The term semantics, in the GP community, refers to the vector of output values produced by evaluating an individual on a set of training instances [15]. Under this definition, a GP individual can be seen as a point in a multi-dimensional semantic space, where the number of dimensions is equal to the number of fitness cases. In standard GP, variation operators produce an offspring by making syntactic manipulation of the parent trees, in the hope that such manipulation will result in a semantics which is closer to the target one. The term Geometric Semantic Genetic Programming (GSGP) designates a GP variant in which syntactic-based GP operators - crossover and mutation - are replaced with so-called Geometric Semantic Operators (GSOs). GSOs introduce semantic awareness in the search process and induce a unimodal fitness landscape in any supervised problem where the fitness function can be defined as a distance between a solution and the target. GSOs, introduced in [8], raised an impressive interest in the GP community [16] because of their attractive property of directly searching the space of underlying semantics of the programs. In this paper, we report the definition of the GSOs for real functions domains, because we used them in our experimental phase. For applications that consider other types of data, the reader is referred to [8].

*Geometric semantic crossover (GSC)* generates, as the unique offspring of parents  $T_1, T_2 : \mathbb{R}^n \rightarrow \mathbb{R}$ , the expression  $T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$ , where  $T_R$  is a random real function whose output values range in the interval  $[0, 1]$ . Analogously, *geometric semantic mutation (GSM)* returns, as the result of the mutation of an individual  $T : \mathbb{R}^n \rightarrow \mathbb{R}$ , the expression  $T_M = T + ms \cdot (T_{R1} - T_{R2})$ , where  $T_{R1}$  and  $T_{R2}$  are random real functions with codomain in  $[0, 1]$  and  $ms$  is a parameter called mutation step. In their work, Moraglio and colleagues show that GSOs create an offspring of significantly larger size with respect to standard GP operators. This makes the fitness evaluation unacceptably slow and considerably constrains practical usability of the GSGP system. To overcome this limitation a possible workaround was proposed in [5], with an efficient implementation of GSOs that makes them usable in practice. This is the implementation used in this work.

## 3 Evolutionary Demes Despeciation Algorithm

In this paper, we propose an advancement of a previously conceived initialization technique, Evolutionary Demes Despeciation Algorithm (EDDA) [14], inspired by the biological concepts of demes evolution and despeciation. In biology, demes are local populations, or subpopulations, of polytypic species that actively interbreed with one another and share distinct gene pools [17]. The term despeciation indicates the combination of demes of previously distinct species into a new population [11]. Albeit not so common in nature, despeciation is a well-known biological phenomenon, and in some cases, it leads to a fortification of the populations. The main idea of EDDA consists in seeding the initial population of  $N$  individuals with good quality individuals that have been evolved, for

a few generations, in  $N$  independent subpopulations (demes), on which distinct evolutionary conditions were imposed. Concretely, demes are evolved by means of conceptually different evolutionary algorithms -  $n\%$  of demes are evolved by means of GSGP, while the remaining  $(100 - n)\%$  use standard GP - and distinct - mostly randomly generated - parameter sets. The experimental results presented in [14] have shown the effectiveness of EDDA. In particular, in all the benchmark problems taken into account, the search process, whose population was initialized with EDDA, ended with solutions with higher, or at least comparable, generalization ability, but with significantly smaller size than the ones found by GSGP using the traditional RHH generative method to initialize the population. In the remaining part of the paper, we will refer to EDDA with the term EDDA-V1.

The advancement we propose in this work, denominated as EDDA-V2, imposes even more adverse evolutionary conditions. Concretely, each deme is evolved using not only a different evolutionary algorithm and parameter set but also a distinct random sample of training data instances and input features. In the following pseudo-code, the distinctive algorithmic features of EDDA-V2 in comparison to EDDA-V1 are presented in bold.

EDDA- $n\%$  (evolving demes for  $m$  generations) ::

1. Create an empty population  $P$  of size  $N$ ;
2. Repeat  $N * (n/100)$  times:
  - (a) Create an empty deme  $D$ ;
  - (b) **Create a sample  $s$  of the training dataset by randomly selecting  $i$  data instances and  $f$  input features;**
  - (c) Randomly initialize this  $D$  using traditional initialization algorithm (RHH is used here);
  - (d) **Using**  $s$ , evolve  $D$ , for  $m$  generations, by means of GSGP;
  - (e) After finishing 2.c), select the best individual from  $D$  and store it in  $P$ ;
3. Repeat  $N * (1 - n/100)$  times:
  - (a) Create an empty deme  $D$ ;
  - (b) **Create a sample  $s$  of the training dataset by randomly selecting  $i$  data instances and  $f$  input features;**
  - (c) Randomly initialize this  $D$  using traditional initialization algorithm (RHH is used here);
  - (d) **Using**  $s$ , evolve  $D$ , for  $m$  generations, by means of GP;
  - (e) After finishing 3.c), select the best individual from  $D$  and store it in  $P$ ;
4. Retrieve  $P$  and use it as the initial population of GSGP **with full training data set.**

**Fig. 1.** Pseudo-code of the EDDA- $n\%$  system, in which demes are left to evolve for  $m$  generations.

As one can see from the pseudo-code reported in Fig. 1, EDDA-V2 uses a different subset of the training instances in each deme, as well as different input features. We claim is that the presence of demes with different fitness cases

and input features should increase the diversity of the initial population, with individuals that are focused on different areas of the semantic space. As a final result of the search process, we would expect a model with an increased generalization ability with respect to GSGP initialized with EDDA-V1 and RHH. As a side effect, using only a percentage of the training instances and input features can be beneficial for reducing the computational effort when a vast amount of training data is available.

## 4 Experimental Study

### 4.1 Test Problems

To assess the suitability of EDDA-V2 as a technique for initializing a population, three real-life symbolic regression problems were considered. Two of them - Plasma Protein Binding level (PPB), and Toxicity (LD50) - are problems from the drug discovery area and their objective is to predict the value of a pharmacokinetic parameter, as a function of a set of molecular descriptors of potential new drugs. The third benchmark is the Energy problem, where the objective is to predict the energy consumption in particular geographic areas and on particular days, as a function of some observable features of those days, including meteorological data. Table 1 reports, for each one of these problems, the number of input features (variables) and data instances (observations) in the respective datasets. The table also reports a bibliographic reference for every benchmark, where a more detailed description of these datasets is available.

**Table 1.** Description of the benchmark problems. For each dataset, the number of features (independent variables) and the number of instances (observations) were reported.

Dataset	# Features	# Instances
Protein plasma binding level (PPB) [1]	626	131
Toxicity (LD50) [1]	626	234
Energy [6]	8	768

### 4.2 Experimental Settings

During the experimental study, we compared the performance of EDDA-V2 against EDDA-V1. The performance are evaluated by considering the quality of the solution obtained at the end of the evolutionary process considering populations initialized with EDDA-V1 and EDDA-V2. Additionally, to consolidate results of our previous work, we also included the GSGP evolutionary algorithm that uses the traditional RHH initialization algorithm. Table 2 reports, in the first column, the main parametrization used for every initialization algorithm (columns two, three and four). The first line in the table contains the number of generations after initialization.

**Table 2.** Parametrization used in every initialization algorithm.

	Parameters	RHH	EDDA.V1	EDDA.V2
1	# generations	2750	{2250, 1250}	{2250, 1250}
2	# generations/deme	-	{5, 15}	{5, 15}
3	# demes	-	100	100
4	% GSGP demes	-	{0, 25, 50, 75, 100}	{0, 25, 50, 75, 100}
5	% sampled instances	-	-	{25, 50, 75}
6	% sampled features	-	-	{25, 50, 75}

For each experiment, any considered initialization algorithm was used to create 100 initial individuals, later evolved by means of GSGP evolutionary process for a given number of generations. In order to ensure comparability of results, all the studied systems performed the same number of fitness evaluations - including, in particular, demes evolution in both EDDA variants. In our experiments, there are 275000 fitness evaluations per run, independently on initialization technique. Every deme, regardless of EDDA variant and parametrization, was initialized by means of the traditional RHH algorithm with 100 individuals, later evolved for some generations. Whenever the traditional RHH was used, tree initialization was performed with a maximum initial depth equal to 6 and no upper limit to the size of the individuals was imposed during the evolution. Depending on the number of iterations used for demes evolution in EDDA variants, the number of generations after despeciation may vary. For example, if, in a given experiment, demes are evolved for 5 generations, then the number of generations after despeciation will be 2250. Similarly, if demes are evolved for 15 generations, then the number of generations after despeciation will be 1250. For both previously mentioned cases, the number of fitness evaluations per run is 275000.

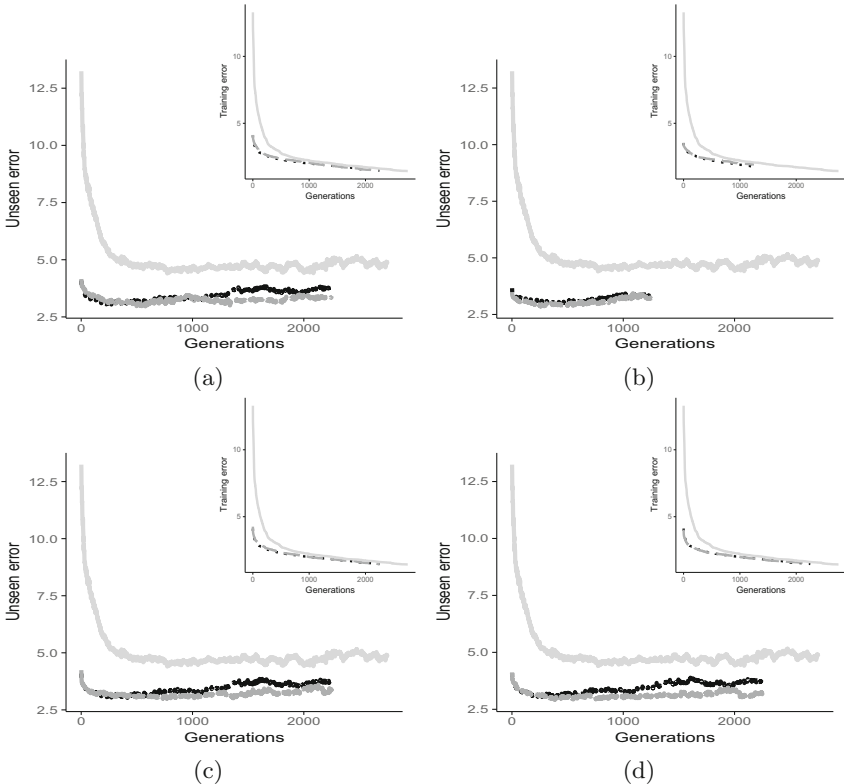
The function set we considered in our experiments was  $\{+, -, *, /\}$ , where  $/$  was protected as in [7]. Fitness was calculated as the Root Mean Squared Error (RMSE) between predicted and expected outputs. The terminal set contained the number of variables corresponding to the number of features in each dataset. Tournament selection of size 5 was used. Survival was elitist as it always copied the best individual into the next generation. As done in [4], the probability of applying GSC and GSM is dynamically adapted during the evolutionary process where the crossover rate is  $p$  and the mutation rate is  $1 - p$ . Following [16], the mutation step  $ms$  of GSM was randomly generated, with uniform probability in  $[0, 1]$ , at each mutation event.

For all the considered test problems, 30 independent runs of each studied system were executed. In each one of these runs, the data was split into a training and a test set, where the former contains 70% of the data samples selected randomly with uniform distribution, while the latter contains the remaining 30% of the observations. For each generation of every studied system, the best individual on the training set has been considered, and its fitness (RMSE) on

the training and test sets was stored. For simplicity, from now on, we will refer to the former as training error and to the latter as test error or unseen error.

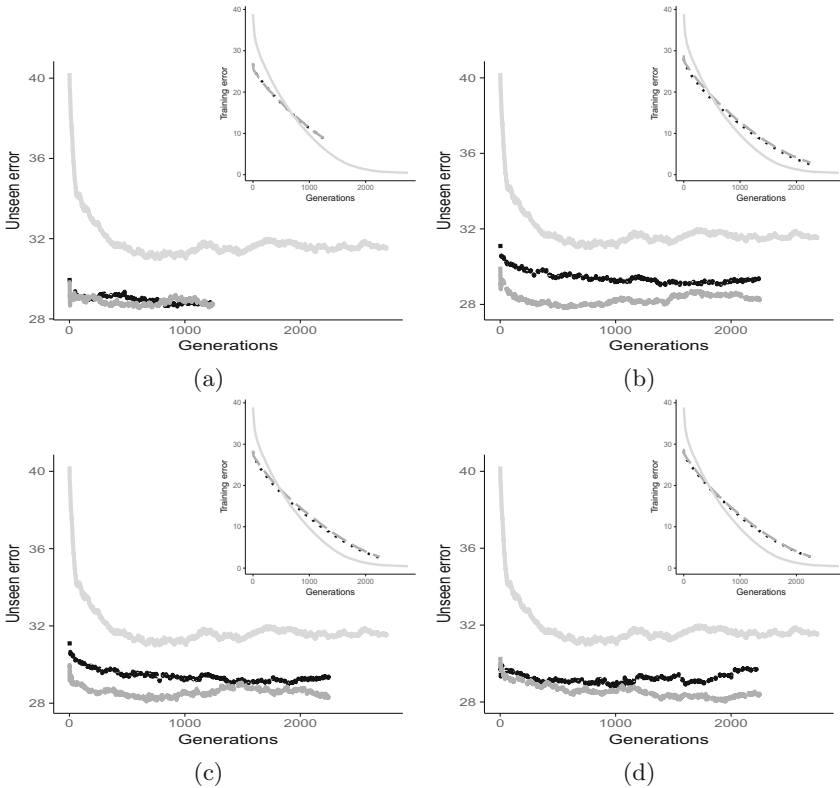
## 5 Results

This section presents the results obtained in the experimental phase. In particular, the section aims at highlighting the differences in terms of performance between the two EDDA variants taken into account and GSGP. For each benchmark, we considered four different parameterizations of EDDA-V1 and EDDA-V2. We denote each parametrization by using the quadruple  $\%GSGP\_MATURITY\_ \%INSTANCES\_ \%FEATURES$ , where the first term corresponds to the percentage of individuals in each deme evolved by means of GSGP, the maturity (i.e., the number of generations the individuals are evolved), the percentage of instances in the dataset and, finally, the percentage of input



**Fig. 2.** Evolution of the (median) best fitness on the training (insets image) and test sets for the energy benchmark and the following parameterizations: (a) 50\_5\_25\_25; (b) 50\_15\_50\_50; (c) 50\_5\_75\_75; (d) 50\_5\_50\_50. The legend for all the plots is: ■ EDDA-V1 — EDDA-V2 — RHH

features considered. The results reported in this section consider values of these four parameters that were randomly selected from the values reported in Table 2. This allows analyzing the performance of EDDA-2 across different problems and parameterizations. Results of the experimental phase are reported from Figs. 2, 3 and 4. Each plot displays the generalization error (i.e., the fitness on unseen instances) and contains an inset showing the training fitness. Considering the training fitness, one can notice the same evolution of the fitness in all the considered benchmarks. EDDA-V1, in particular, is the best performer followed by EDDA-V2 and GSGP. Focusing on the two EDDA variants, these results were expected since EDDA-V1 is learning a model by using the whole training set and all the available features. On the other hand, EDDA-V2 is learning a model of the data considering a sample of the whole training set and, additionally, only a reduced number of features. Under this light, it is interesting to comment on the performance of EDDA-V2 and GSGP. The experimental results suggest that the

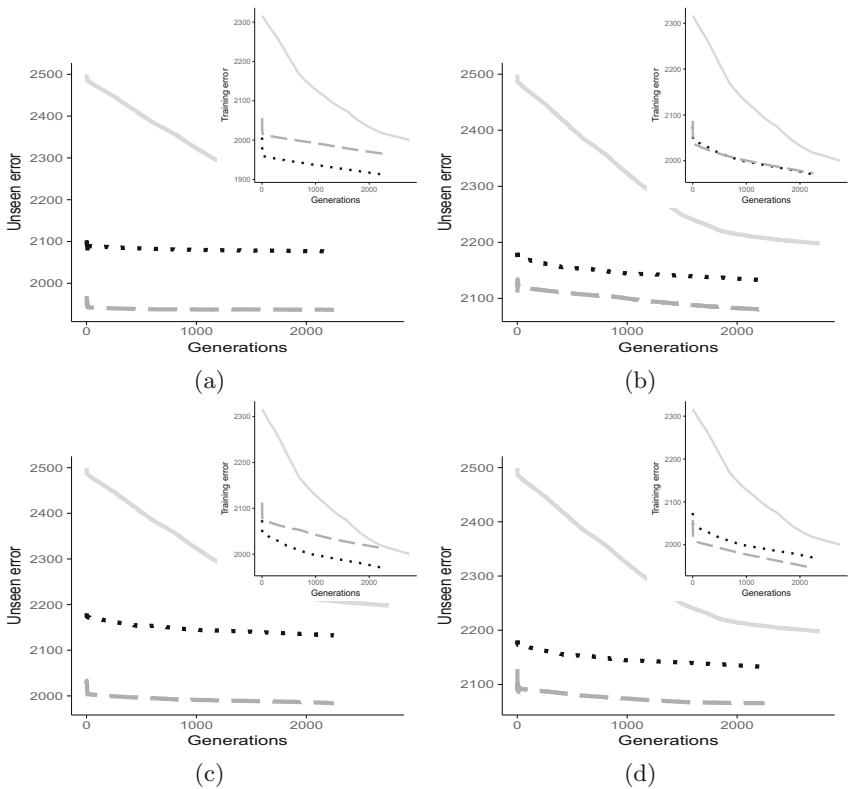


**Fig. 3.** Evolution of the (median) best fitness on the training (insets image) and test sets for the PPB benchmark and the following parametrizations: (a) 25\_15\_75\_75; (b) 25\_5\_50\_50; (c) 25\_5\_75\_75; (d) 75\_5\_75\_75. The legend for all the plots is: ■ EDDA-V1 — EDDA-V2 — RHH

usage of RHH for initializing the population results in poor performance when compared to EDDA-V2.

To summarize, results show the superior performance of EDDA-V1 when training error is taken into account, but EDDA-V2 produces a final model with an error that is smaller than the one produced by GSGP. This is a notable result because it shows that the proposed initialization method can outperform GSGP initialized with ramped half and half by considering a lower number of training instances and features.

While results on the training set are important to understand the ability of EDDA-V2 to learn the model of the training data, it is even more important and interesting to evaluate its performance on unseen instances. Considering the plots reported from Figs. 2, 3 and 4, one can see that the EDDA-V2 actually produces good quality solutions that are able to generalize over unseen instances. In particular, EDDA-V2 presents a very nice behavior in the vast majority of



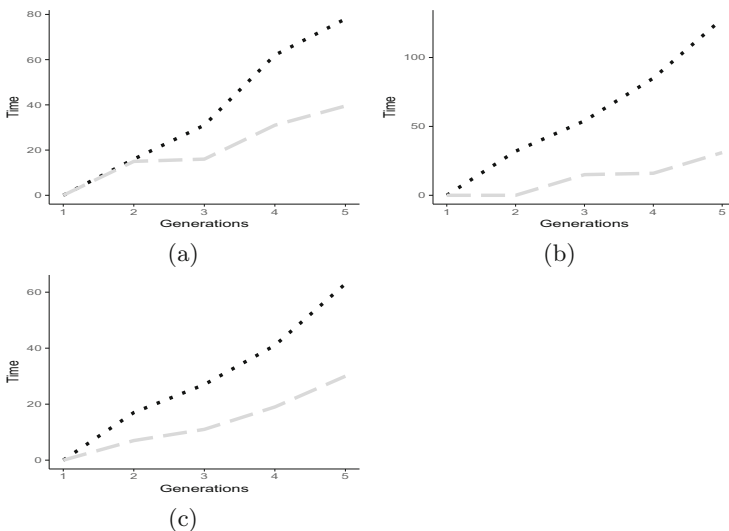
**Fig. 4.** Evolution of the (median) best fitness on the training (insets image) and test sets for the LD50 benchmark and the following parameterizations: (a) 25\_5\_75\_75; (b) 10\_5\_25\_25; 10\_5\_50\_50(c); (d) 10\_5\_75\_75. The legend for all the plots is: ■ EDDA-V1 — EDDA-V2 — RHH

the problems, showing better or comparable performance with respect to the other competitors without overfitting to the training data. Focusing on the other techniques, GSGP is the worst performer over all the considered benchmarks and parameterizations.

To summarize the results of this first part of the experimental phase, it is possible to state that EDDA-V2 outperforms GSGP with respect to the training fitness by also producing models able to generalize over unseen instances. When EDDA-V2 is compared against EDDA-V1, it performs poorer on the training instances, but the generalization error is better with respect to the latter system.

To conclude the experimental phase, Fig. 5 shows the time (ms) needed to initialize and evolve demes with EDDA-V1 and EDDA-V2. As expected EDDA-V2 requires less computational time.

To assess the statistical significance of these results, a statistical validation was performed considering the results achieved with the EDDA variants. First of all, given that it is not possible to assume a normal distribution of the values obtained by running the different EDDA variants, we ran the Shapiro-Wilk test and we considered a value of  $\alpha = 0.05$ . The null hypothesis of this test is that the values are normally distributed. The result of the test suggests that the null hypothesis should be rejected. Hence, we used the Mann-Whitney U test for comparing the results returned EDDA-V2 against the ones produced by EDDA-V1 under the null hypotheses that the distributions are the same across repeated measures. Also, in this test a value of  $\alpha = 0.05$  was used.



**Fig. 5.** Time needed to initialize and evolve a deme for 5 iterations, with a 50% of GSGP individuals, 50% of training instances, and 50% of features. Median calculated over all the demes and runs for (a) Energy, (b) PPB, and (c) LD50. The legend for all the plots is: ■ EDDA-V1 — EDDA-V2



Table 3 reports the  $p$ -values returned by the Mann–Whitney test, and **bold** is used to denote values suggesting that the null hypotheses should be rejected. Considering these results, it is interesting to note that with respect to the training error, EDDA-V2 and EDDA-V1 produced comparable results in the vast majority of the benchmarks and configurations taken into account. The same result applies to the test error, where it is important to highlight that, in each benchmark, there exists at least one parameters configuration that allows EDDA-V2 to outperform EDDA-V1.

**Table 3.**  $p$ -values returned by the Mann–Whitney U test. Test and training error achieved by populations initialized with EDDA-V2 and EDDA-V1 are compared. Values in the column parametrization correspond to the ones used in subplots (A), (B), (C), (D) of Figs. 2, 3, and 4. **Bold** is used to denote  $p$ -values suggesting that the null hypotheses should be rejected.

Parametrization	Test			Training		
	Energy	PPB	LD50	Enrgy	PPB	Ld50
A	<b>0.048</b>	0.203	0.065	<b>0.043</b>	0.523	0.109
B	0.708	0.267	0.230	0.123	<b>0.035</b>	0.273
C	0.440	0.230	<b>0.016</b>	0.187	0.142	<b>0.031</b>
D	0.708	0.203	0.390	0.109	0.843	0.901

## 6 Conclusions

Population initialization plays a fundamental role in the success of GP. Different methods were developed and investigated in the EA literature, all of them pointing out the importance of maintaining diversity among the different individuals in order to avoid premature convergence. A recent contribution, called Evolutionary Demes Despeciation Algorithm (EDDA-V1 in this paper), introduced an initialization technique in GP inspired by the biological phenomenon of demes despeciation. The method seeds a population of  $N$  individuals with the best solutions obtained by the independent evolution of  $N$  different populations, or demes. EDDA-V1 has demonstrated its effectiveness in initializing a GSGP population when compared to the standard ramped half and half method. This paper extended the initialization technique by defining a new method, called EDDA-V2 that initializes a population by evolving different parallel demes and, in each deme, it uses a different subset of the training instances and a different subset of the input features. This ensures an increased level of diversity, by also reducing the time needed for the initialization step. Experimental results obtained over three benchmark problems demonstrated that populations initialized with EDDA-V2 and evolved by GSGP converge towards solutions with a comparable or better generalization ability with respect to the ones initialized with EDDA-V1 and the traditional ramped half and half technique.

## References

1. Archetti, F., Lanzeni, S., Messina, E., Vanneschi, L.: Genetic programming for computational pharmacokinetics in drug discovery and development. *Genet. Program. Evol. Mach.* **8**(4), 413–432 (2007)
2. Beadle, L.C.J.: Semantic and structural analysis of genetic programming. Ph.D. thesis, University of Kent, Canterbury, July 2009
3. Beadle, L.C.J., Johnson, C.G.: Semantic analysis of program initialisation in genetic programming. *Genet. Program. Evol. Mach.* **10**(3), 307–337 (2009)
4. Castelli, M., Manzoni, L., Vanneschi, L., Silva, S., Popovič, A.: Self-tuning geometric semantic genetic programming. *Genet. Program. Evol. Mach.* **17**(1), 55–74 (2016)
5. Castelli, M., Silva, S., Vanneschi, L.: A C++ framework for geometric semantic genetic programming. *Genet. Program. Evol. Mach.* **16**(1), 73–81 (2015)
6. Castelli, M., Vanneschi, L., Felice, M.D.: Forecasting short-term electricity consumption using a semantics-based genetic programming framework: the south italy case. *Energy Econ.* **47**, 37–41 (2015)
7. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge (1992)
8. 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. LNCS*, vol. 7491, pp. 21–31. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32937-1\\_3](https://doi.org/10.1007/978-3-642-32937-1_3)
9. Oliveira, L.O.V., Otero, F.E., Pappa, G.L.: A dispersion operator for geometric semantic genetic programming. In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO 2016*, pp. 773–780. ACM (2016)
10. Pawlak, T.P., Wieloch, B., Krawiec, K.: Review and comparative analysis of geometric semantic crossovers. *Genet. Program. Evol. Mach.* **16**(3), 351–386 (2015)
11. Taylor, E.B., Boughman, J.W., Groenenboom, M., Sniatynski, M., Schluter, D., Gow, J.L.: Speciation in reverse: morphological and genetic evidence of the collapse of a three-spined stickleback (*gasterosteus aculeatus*) species pair. *Mol. Ecol.* **15**(2), 343–355 (2006)
12. Tomassini, M., Vanneschi, L., Collard, P., Clergue, M.: A study of fitness distance correlation as a difficulty measure in genetic programming. *Evol. Comput.* **13**(2), 213–239 (2005)
13. Vanneschi, L.: An introduction to geometric semantic genetic programming. In: Schütze, O., Trujillo, L., Legrand, P., Maldonado, Y. (eds.) *NEO 2015. SCI*, vol. 663, pp. 3–42. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-44003-3\\_1](https://doi.org/10.1007/978-3-319-44003-3_1)
14. Vanneschi, L., Bakurov, I., Castelli, M.: An initialization technique for geometric semantic GP based on demes evolution and despeciation. In: *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, 5–8 June 2017*, pp. 113–120 (2017)
15. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. *Genet. Program. Evol. Mach.* **15**(2), 195–214 (2014)
16. Vanneschi, L., Silva, S., Castelli, M., Manzoni, L.: Geometric semantic genetic programming for real life applications. In: Riolo, R., Moore, J.H., Kotanchek, M. (eds.) *Genetic Programming Theory and Practice XI. GEC*, pp. 191–209. Springer, New York (2014). [https://doi.org/10.1007/978-1-4939-0375-7\\_11](https://doi.org/10.1007/978-1-4939-0375-7_11)
17. Wilson, D.S.: Structured demes and the evolution of group-advantageous traits. *Am. Nat.* **111**(977), 157–185 (1977). <https://doi.org/10.1086/283146>



# Extending Program Synthesis Grammars for Grammar-Guided Genetic Programming

Stefan Forstenlechner<sup>(✉)</sup>, David Fagan, Miguel Nicolau, and Michael O’Neill

Natural Computing Research and Applications Group, School of Business,  
University College Dublin, Dublin, Ireland  
stefan.forstenlechner@ucdconnect.ie,  
{david.fagan,miguel.nicolau,m.oneill}@ucd.ie

**Abstract.** Program synthesis is a problem domain that due to its importance is tackled by many different fields, one being Genetic Programming. Two variants, Grammar-Guided Genetic Programming (G3P) and PushGP, have been applied to a vast general program synthesis benchmark suite and solved a variety of problems although with varying success rates. While G3P achieved higher success rates on some problems, PushGP was able to find solutions to more problem instances. Reason why G3P fails at some problems might be missing functionality in the grammars or knowledge that has to be discovered during the runs. In this paper the current shortcomings of G3P are analysed and the paper’s contributions include an example of extending grammars for program synthesis, a fairer comparison between PushGP and G3P with a more similar function set as well as new results on problems that have not been solved with G3P and one that has not been solved with PushGP.

**Keywords:** Genetic Programming · Grammar · Program synthesis

## 1 Introduction

Genetic Programming has shown potential to solve a range of general program synthesis problems. In contrast to other problem domains like regression where an approximation of the solution might be acceptable, a partially correct solution is usually of no use in program synthesis. But for GP to be successful in program synthesis, the ability to find a correct solution should be high, as practitioners should not have to be required to run GP multiple times while researchers only do multiple runs for statistical tests. At the same time, it is essential that GP can solve a wide range of program synthesis problems rather than special cases.

To this end, a range of difficult or unsolved problems is identified in the general program synthesis benchmark suite [8], that has been used recently to test GP on program synthesis, especially with G3P [3] and PushGP [17]. While G3P was able to achieve a higher percentage of successful solutions found in

cases it found solutions, PushGP was able to solve more problems at least once in general.

The focus of this paper lies in identifying differences between the function set of G3P and PushGP, extending the grammars according to those differences as well as the identified difficult problems from the benchmark suite and extending the grammars accordingly. At the same time, the grammars shall stay as general as possible to be able to use them outside of the context of benchmark problems and should not be trimmed to “cheat” on any particular problem within the benchmark suite. As the benchmark suite that has been used so far, proposes to have an explicit *char* data type which is currently missing in G3P [3] the possibility of adding it is further investigated. Therefore, the functionality available in the grammars is not allowed to be extended further than the function set available to PushGP.

The rest of the paper is structured in the following way. Section 2 summaries related work on program synthesis. Section 3 describes the benchmark suite used in the GP community for program synthesis and what problems have been difficult for GP and particularly for G3P. Afterwards, Sect. 4 describes in what ways grammars can be extended to overcome the previous shortcomings. The experimental setup used to tackle the benchmark suite is described in Sect. 5 and the results are compared to previous approaches in Sect. 6. Finally, conclusion and future work are discussed in Sect. 7.

## 2 Related Work

Program synthesis problems have been tackled even before GP was used and many different approaches exist [11]. Nevertheless, GP systems have proven to be very flexible and successful at doing this. Therefore this paper will focus on GP systems.

### 2.1 Grammar-Guided Genetic Programming

Grammar-Guided Genetic Programming [12] is a GP variant that uses grammars to define the search space. This makes it easy to use and flexible as a grammar can be defined outside of the GP system instead of restricting GP to a certain function set. Additionally, it is quite powerful, because any program that can be generated with the grammar can be found by GP. Grammars also provide the possibility of adding bias, if necessary. The most famous variants are CFG-GP by Whigham [19] and grammatical evolution [14].

Forstenlechner et al. [3] proposed a grammar design for GP to tackle general program synthesis problems, as mainly bespoke grammars have been used before to solve program synthesis [13], which can not be reused to solve other problems. The idea of the grammar design is to have multiple smaller grammars and every grammar contains only the functionality for a single data type. Additionally, one general grammar exists which contains the structure of the program. The benefit of this design is that it is not limited to a single programming

language and depending on the problem at hand a subset of the data types required to solve the problem can be chosen. Therefore, the design is capable of solving general purpose program synthesis problems, while the search space can be kept small by not including unnecessary data types. Functions that require multiple data types of which some are not available, will be removed from the grammar automatically when combining the grammars for a chosen problem.

### 3 General Program Synthesis Benchmark Suite Remarks

A general program synthesis benchmark suite was introduced by Helmuth and Spector [8]. It provides a variety of problems from introductory computer science courses. It consists of a total of 29 problems with a description, training and test set, fitness function and general parameter settings, mainly for PushGP [17], for every problem. Additionally, every problem requires specific data types to be available to be solved. A more detailed description is available in form of a technical report [18], which also contains information about how to generate the training and test data as well as the instructions available for PushGP.

The two GP systems that have been tested on the benchmark suite are a G3P by Forstenlechner [3] and PushGP [17]. PushGP is a GP system that evolves programs in the language Push, which was solely designed for evolutionary algorithms. Push uses stacks to store data instead of using variables. It has a stack for every data type as well as for the code that is executed, which makes it possible to manipulate the code during runtime.

An additional comparison of systems outside of the GP community, namely Flash Fill [5] and MagicHaskeller [9], was done on the benchmark suite in [15]. The comparison showed that GP systems are more flexible and more successful on this benchmark suite, although it should be mentioned that these systems have been created with other use cases in mind like Flash Fill is used in Microsoft Excel for string manipulation tasks.

In the initial introduction of the grammar design for program synthesis problems [3], the functionality was kept to the basics of Python without including more than was available in PushGP. For example, adding the built-in *sum* function from Python would make solving the problem Vector Average fairly easy.

Table 1 shows the results achieved with G3P on the general program synthesis benchmark suite. The results have been taken from [3]. The datasets of Checksum and Vector Average have been changed since the benchmark suite has been introduced and a simpler version of Super Anagrams has been used in [3]. The table indicates that G3P with the current grammars has difficulty to solve problems that require *char* as a data type. At the moment it only uses *string*, most likely because the initial grammars are based on Python which treats *char* as string. While a programmer has no difficulty to understand how or when to use a single character string, it is definitely more complicated for GP to find out how or when to use it. Adding a *char* data type could yield better results. Additionally, PushGP was able to solve more problems from the benchmark suite, although in many cases with a low success rate. Nevertheless, adding further functionality could help improve the results of G3P.

**Table 1.** Results of G3P on the general program synthesis benchmark suite sorted by successfully found solutions. *String* and *Char* column indicate if these data types have to be used when solving the problem. A \* indicates if the data set has been changed, since the results have been acquired.

	NumberIO	Smallest	Vectors Summed	Median	String Lengths Backwards	Negative To Zero	Grade	Last Index of Zero	Super Anagrams*	Count Odds	For Loop Index	Small Or Large	Vector Average*	Sum of Squares	Compare String Lengths	Scrabble Score	Even Square	Checksum*	Collatz Number	Digits	Double Letters	Mirror Image	Pig Latin	Replace Space with Newline	Syllables	Wallis Pi	Word Stat	X-Word Lines
Successes	94	94	91	79	68	63	31	22	21	12	8	7	5	3	2	2	1	0	0	0	0	0	0	0	0	0	0	0
String					X	X			X		X			X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Char									X						X	X	X	X	X	X	X	X	X	X	X	X	X	X

## 4 Extending Program Synthesis Grammars

This section describes how the program synthesis grammars from [3] have been extended to include an additional *char* data type as well as additional functionality to have a fairer comparison to PushGP. Extending the grammar also means increasing the size of the search space as more programs can be generated from the grammar. Therefore, the extension of the grammars can also have a negative effect on the search performance.

### 4.1 Data Type Char

As shown in Sect. 3, G3P does poorly on problems that require a data type *char*. G3P only used *string* as it mainly relied on Python even though the concepts can be applied to other languages as well and because a *char* can be interpreted as a string of length one. As many problems in the general program synthesis benchmark suite require to check or manipulate single characters, G3P not using a *char* grammar could explain why it currently fails at solving such problems. While programmers have the intrinsic knowledge that a string consists of characters and a string of length one can be treated similar to a *char*, GP either has to discover this knowledge or has to be told a priori. The currently available grammar data types are *bool*, *integer*, *float* and *string*, as well as a list version grammar of each of these data types, plus the new *char* grammar. A list of *char* grammar is currently not included as the benchmark suite does not require it and strings can be viewed as a list of *char*. As G3P adds variables of the data types of every used grammar to the evolved program, including the *char* grammar makes it very likely that *chars* are used as opposed to before where G3P had to find that a string of length one is required.

## 4.2 Recursion

Recursion is a method of programming where a program calls itself to solve a smaller instance of the same problem first and uses that solution to solve the initial problem. Recursion is not an uncommon strategy to tackle problems in GP [1,20]. In many cases, a recursive solution can be significantly shorter in terms of code than an iterative program, which might make it easier for GP to find. PushGP is capable of evolving recursive programs and for a fair comparison should be part of the grammars for G3P as well.

To allow recursion, a program needs to be able to call itself and a way to stop the recursion, usually an *if* condition called guard. As the grammars in G3P are automatically merged together depending on the required data types, and the number of input/output variables, as well as their types, a rule for a recursive call can be generated and added to the grammar. The following is an example where `outputX` is replaced with the correct type variable non-terminal (e.g. `<bool_var>`) and `inputX` with the correct type (e.g. `<bool>`):

```
<output1>', '...', '<outputN>' = evolve('<input1>', '...', '<inputN>')
```

In a similar way, a return statement can be generated:

```
'return result1, ..., resultN'
```

The grammar used to define the control flow (*structure.bnf*) already contains *if* statements, but it is very likely that it might not be used and the program gets stuck in an infinite recursion and at some point will throw an error due to a stack overflow. A problem that occurs with infinite loops as well and was handled by adding a guard to avoid any additional iterations if a certain limit is reached. A similar guard is used to avoid infinite recursion. The benefit of using this mechanism is that evolved programs will not throw an error and return a value. Therefore, the program will be given a fitness value based on what it returns instead of a default worst case fitness due to an error.

## 4.3 List Operations

When the grammars for program synthesis were introduced grammars for lists of all data types were included but kept to the essential functionality. Items could be added at the end, inserted or replaced at a specific index or removed. Lists could be iterated, compared, checked if they are empty and their length could be determined as well as slicing of lists was possible. Any additional functionality the algorithm had to find. PushGP offers more functionality out of the box that can be used, which has been added to the grammars for G3P, like reversing a list, counting the occurrences of an item, replacing or removing items if a condition is met etc. All of this functionality could be discovered as well, but as for example O'Neill et al. [13] showed that GP has difficulties finding a solution to the integer sorting problem, but by adding a swap function the problem was easily solvable. As stated before no further functionality has been added, that was not already available for PushGP as well. At the same time, it should be noted that adding

additional functionality also increases the search space, which can make it more difficult to find a correct solution. Even though the additional functionality can make it easier to solve one problem, it can make it more difficult to solve another. Therefore a decrease of successful solutions found on some problems is to be expected.

#### 4.4 Additional Methods

Similar to the list operations in the previous section, additional methods were added to other data types that in general could have been discovered by G3P. One example that is also often not included for boolean problems is *XOR*, as it can be constructed with *AND*, *OR* and *NOT* and can make certain problems like multiplexer too easy [10]. To be able to have a better comparison between G3P and PushGP, such methods have been added as well. As there are too many to mention every single one of them, the reader is referred to the grammars themselves that are provided online [2] as well as [18]. Again, it should be noted that the extended grammars do not exceed the functionality that is provided by PushGP.

## 5 Experimental Setup

For the experiments, the extended grammars, which are described in the previous section, are used with the same G3P system as in [3], which is available online [2] including the extended grammars. The experiments are run on the problems from the general program synthesis benchmark suite [8]. The parameter settings are summarized in Table 2. The number of generations is set to 300<sup>1</sup>. As soon as a successful solution is found, the run is stopped as GP cannot improve it anymore. Lexicase selection [6] is used, as it has shown to be the most successful selection operator with GP on program synthesis problems. Instead of using a single fitness value for selection, lexicase operates on the fitness values of every single training cases. It randomly selects a fitness case and selects the best individual based on that case. In case of a tie, lexicase selection continues with a subset of individuals that were in this tie and continue to select other training cases until a single individual is left or until no fitness case is left, in which case an individual is selected randomly.

## 6 Results

First the overall performance of G3P with the extended grammars and also to PushGP. Afterwards, the effect of the extended grammars on the search is analysed in more detail.

---

<sup>1</sup> 200 for Normal IO, Median and Smallest as proposed in [8].



**Table 2.** Experimental parameter settings

Parameter	Setting
Runs	100
Generations	300 (see footnote 1)
Population size	1000
Selection	Lexicase
Crossover probability	0.9
Mutation probability	0.05
Elite size	1
Node limit	250
Variables per type	3
Max execution time	1 s
Max_Tries	10

## 6.1 Successful Solutions

Table 3 shows the solutions found for each problem with G3P with extended grammars for training and test with 100 runs. The results are compared to the previously achieved successful solutions of G3P from [3]. Of the eight problems that require a *char* data type and have not been solved with G3P before, three have been solved with the extended grammars, namely Pig Latin, Replace Space with Newline and Syllables. Pig Latin is one that has not been solved with PushGP either. Additionally, Mirror Image has been solved as well, probably due to the additional list operations, which was not solved with the G3P with previous grammars. Table 3 also includes the p-value for the Wilcoxon Rank sum test on best test fitness of the two grammar approaches and shows a significant difference for nearly all of the problems. This is not surprising as the grammar has a massive influence on the search, as a function set has on normal GP.

The results also show that due to the increased search space, which is caused by the additional functions added to the grammar, the number of successful solutions decreases for some problems. Three problems, Compare String Lengths, Even Squares and Vector Average, could not be solved anymore, but the success rate of the first two was rather small before as well. Especially, Compare String Lengths is highly overfit as 96 successful solutions were found on test, but none generalizes on test. This is a problem that occurs on multiple problem instances and has been noticed before [7].

Even though on the final experiments some problems, even those which require *char* as data type are still not solved, in preliminary experiments Checksum and Double Letters have been solved with G3P with extended grammars as well. Even then the success rate was rather small, but theoretically, it has been found that they can be solved with the extended grammars as well.

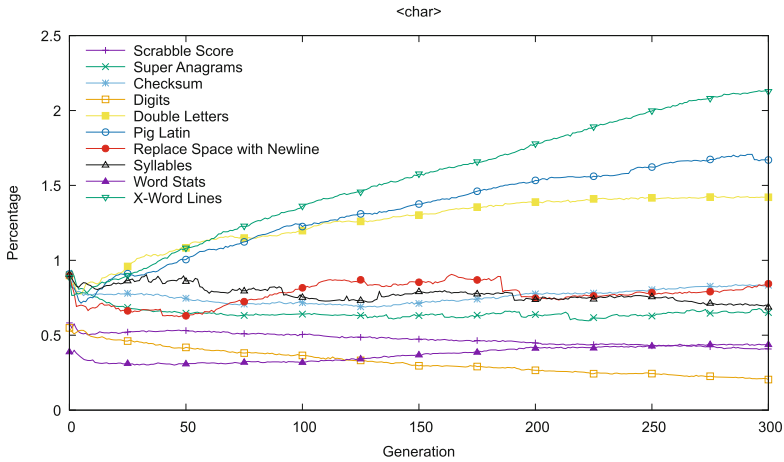
**Table 3.** Successful solutions found with G3P with extended grammars on training and test with 100 runs as well as increase and decrease to the previous grammars in brackets. The p-value shows if there is a significant difference in the best test performance between the two different grammars with 0.05 as level of significance. A significant difference is highlighted in bold. Finally, the results of PushGP on the benchmark suite from [8] and the difference to G3P with extended grammars in brackets are compared.

Problem Name	Test	G3P		p-value	PushGP Test
		Training			
Checksum	0 (+0)	0 (+0)		<b>8.74E-32</b>	0 (+0)
Collatz Numbers	0 (+0)	0 (+0)		0.0991	0 (+0)
Compare String Lengths	0 (-2)	96 (-1)		<b>8.06E-05</b>	7 (+7)
Count Odds	3 (-9)	4 (-8)		<b>1.81E-15</b>	8 (+5)
Digits	0 (+0)	0 (+0)		<b>0.0298</b>	7 (+7)
Double Letters	0 (+0)	0 (+0)		<b>0.0040</b>	6 (+6)
Even Squares	0 (-1)	0 (-1)		0.5683	2 (+2)
For Loop Index	6 (-2)	9 (-11)		<b>0.0006</b>	1 (-5)
Grade	31 (+0)	63 (-18)		0.1005	4 (-27)
Last Index of Zero	44 (+22)	97 (+43)		<b>5.71E-11</b>	21 (-23)
Median	59 (-20)	99 (-1)		<b>0.0039</b>	45 (-14)
Mirror Image	25 (+25)	89 (+38)		<b>3.25E-18</b>	78 (+53)
Negative To Zero	13 (-50)	24 (-42)		<b>9.12E-07</b>	45 (+32)
Number IO	83 (-11)	95 (-5)		<b>1.21E-15</b>	98 (+15)
Pig Latin	3 (+3)	4 (+4)		<b>4.02E-25</b>	0 (-3)
Replace Space with Newline	16 (+16)	29 (+29)		<b>5.08E-30</b>	51 (+35)
Scrabble Score	1 (-1)	1 (-4)		<b>0.0008</b>	2 (+1)
Small Or Large	9 (+2)	39 (-12)		0.5493	5 (-4)
Smallest	73 (-21)	100 (+0)		<b>9.58E-05</b>	81 (+8)
String Lengths Backwards	18 (-50)	20 (-48)		<b>6.70E-17</b>	66 (+48)
Sum of Squares	5 (+2)	5 (+2)		<b>8.02E-05</b>	6 (+1)
Super Anagrams	0 (+0)	43 (-1)		<b>2.33E-34</b>	0 (+0)
Syllables	39 (+39)	53 (+53)		<b>4.28E-29</b>	18 (-21)
Vector Average	0 (-16)	0 (-17)		<b>6.94E-32</b>	16 (+16)
Vectors Summed	21 (-70)	28 (-65)		<b>1.84E-23</b>	1 (-20)
Wallis Pi	0 (+0)	0 (+0)		<b>3.03E-24</b>	0 (+0)
Word Stats	0 (+0)	0 (+0)		0.7722	0 (+0)
X-Word Lines	0 (+0)	0 (+0)		<b>2.56E-34</b>	8 (+8)

Finally, Table 3 shows the results of PushGP taken from [8] compared to G3P with extended grammars. According to [7], PushGP is able to solve Checksum after the original dataset has been changed. The comparison shows that both approaches have problems where one method is more capable to find solutions than the other, but there does not seem to be a clear advantage over one or the other. Some problems have been solved with PushGP that have currently not been solved with G3P, but again the success rates of these problems are very small, below 10, in most cases, which makes a comparison difficult. The low success rate is an issue that needs to be addressed by both approaches.

### 6.2 Char Analysis

The grammar for the *char* data type is used by 10 problems. The grammar contains a rule `<char>` with productions for char variables, char constants and all functions that return a char value. Therefore, checking the percentage of nodes in individuals shows if GP is making use of the additional data type. Figure 1 depicts this usage.



**Fig. 1.** Percentage of `<char>` nodes in individuals averaged over 100 runs over generations.

In the initial generation, the percentage of nodes being `<char>` is nearly identical for some problems, which is expected as these problems require the same data types, which means the grammars are nearly identical, except maybe input and output variables. Therefore the grammar has the same structure and the same number of possible nodes, which leads to this effect. The percentage of `<char>` nodes used may seem small being between 0.5% and 1.5%, but considering the number of productions available in the grammar, it is rather high. In case of almost all problems, the usage of `<char>` nodes is either constant or increases over time, after a few generations. The only problem that seems to slowly decrease the usage of `<char>` nodes is Digits. This can be explained by how G3P is tackling the problems. While PushGP prints every integer for Digits, G3P has to return a list of integers as it does not use print statements and therefore does not necessarily need a char data type.

For some problems, especially Replace Space with Newline, Syllables, Super Anagrams and Pig Latin, the lines are not as stable as for the other problems. The reason is that solutions that solve the problem at least for training have been found and runs are stopped as soon as this happens. Hence, the average percentage might drop or increase. In most cases, a sudden drop has been found, which shows that runs that use `<char>` nodes more often seem to be able to find a successful solution earlier. This indicates that the char grammar improves the search for successful solutions.

### 6.3 Recursion Analysis

The percentage of recursion used can be checked in a similar way as in the previous section for `char`. Figure 2 depicts the percentage of recursion nodes used over generations. The initial percentage is lower than with `<char>`, because there is only one recursion production rule in the grammar, whereas `<char>` is used by multiple functions. Afterwards, it drops even lower for all problems and is barely used overall. As explained in Sect. 4.2, to use recursion, a method needs to be able to call itself and a stopping criterion. At the moment the GP system can evolve a method to call itself, but at the same time has to evolve a stopping criterion, which seems to make it too complicated to be used. Without the stopping criterion, the evolved program runs into an infinite loop, which leads to a stack overflow or a timeout by the G3P system. A way to improve this might be to adapt the grammar that a stopping criterion is added to the same production rule as the recursion to always have both added at the same time. This could increase the chance to make G3P use recursion to solve problems.

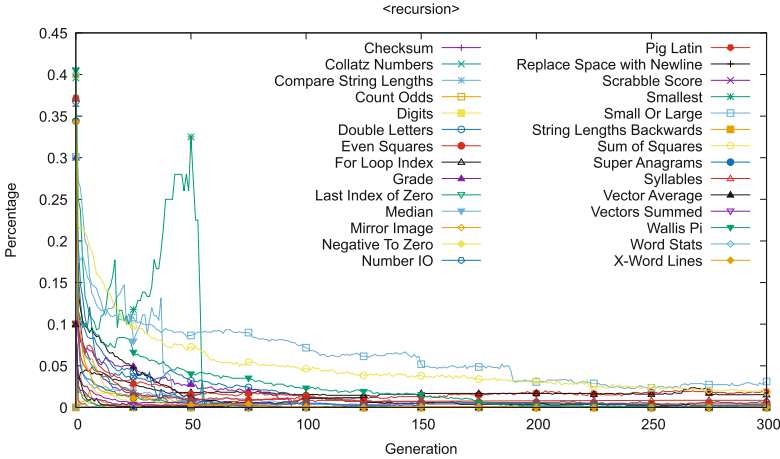


Fig. 2. Percentage of recursion nodes in individuals averaged over 100 runs over generations.

## 7 Conclusion and Future Work

The difficulties of solving multiple problems of the general program synthesis benchmark suite with a grammar design approach [3] have been discussed. As some of these problems have been solved with another approach before, the functionality of the grammars has been extended in various ways to be closer to previous approaches, without “cheating” by adding functionality not used before. An important enhancement of the grammars is that an explicit *char* grammar has been added as many problems operate on single characters instead of strings.

Programmers are able to identify such characteristics of a problem easily, while GP would have to discover such knowledge. As the benchmark suite proposes to use *char* as its own data type, this additional information does not give G3P an unfair advantage when comparing to other systems.

Afterwards, the extended grammars are used to tackle the program synthesis benchmark suite and the results are compared to the grammar design of [3]. The results show significant differences for nearly all problems and successful solutions have been found for previously unsolved problems with G3P. One problem, Pig Latin, has been successfully solved that was not solved by any other approach before. Additionally, a comparison with PushGP has been made, as the extended grammars are closer in functionality to PushGP as in [3].

Due to the increased search space created by the extended grammars, a decrease of successful solutions found on previously solved problems was expected. A way to dynamically adjust the functionality of grammars during runs could help avoid this problem [16]. Even the success rates of newly solved problems were rather low. This is a problem not only of G3P, but also of other approaches, and should be addressed in the future to make program synthesis with GP more usable outside of the research community as well. Current approaches include smarter operators [4] or post-run simplifications [7], but further research is required to increase success rates.

**Acknowledgments.** This research is based upon works supported by the Science Foundation Ireland, under Grant No. 13/IA/1850.


## References

1. Agapitos, A., Lucas, S.M.: 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). [https://doi.org/10.1007/11729976\\_15](https://doi.org/10.1007/11729976_15)
2. Forstenlechner, S.: Github repository: HeuristicLab.CFGGP: Provides context free grammar problems for HeuristicLab (2016). <https://github.com/t-h-e/HeuristicLab.CFGGP>. Accessed 22 Mar 2018
3. Forstenlechner, S., Fagan, D., Nicolau, M., O’Neill, M.: A grammar design pattern for arbitrary program synthesis problems in genetic programming. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E., García-Sánchez, P. (eds.) EuroGP 2017. LNCS, vol. 10196, pp. 262–277. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55696-3\\_17](https://doi.org/10.1007/978-3-319-55696-3_17)
4. Forstenlechner, S., Fagan, D., Nicolau, M., O’Neill, M.: Semantics-based crossover for program synthesis in genetic programming. In: Lutton, E., Legrand, P., Parrend, P., Monmarché, N., Schoenauer, M. (eds.) EA 2017. LNCS, vol. 10764, pp. 58–71. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78133-4\\_5](https://doi.org/10.1007/978-3-319-78133-4_5)
5. Gulwani, S.: Automating string processing in spreadsheets using input-output examples. In: Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, pp. 317–330. ACM, New York (2011)
6. Helmuth, T., Spector, L., Matheson, J.: Solving uncompromising problems with lexicase selection. *IEEE Trans. Evol. Comput.* **19**(5), 630–643 (2015)

7. Helmuth, T., McPhee, N.F., Pantridge, E., Spector, L.: Improving generalization of evolved programs through automatic simplification. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 937–944. ACM, Berlin, 15–19 July 2017
8. Helmuth, T., Spector, L.: General program synthesis benchmark suite. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, GECCO 15, pp. 1039–1046. ACM, Madrid, 11–15 July 2015
9. Katayama, S.: Recent improvements of MagicHaskell. In: Schmid, U., Kitzelmann, E., Plasmeyer, R. (eds.) AAIP 2009. LNCS, vol. 5812, pp. 174–193. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-11931-6\\_9](https://doi.org/10.1007/978-3-642-11931-6_9)
10. Keijzer, M., Ryan, C., Murphy, G., Cattolico, M.: Undirected training of run transferable libraries. In: Keijzer, M., Tettamanzi, A., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 361–370. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-31989-4\\_33](https://doi.org/10.1007/978-3-540-31989-4_33)
11. Kitzelmann, E.: Inductive programming: a survey of program synthesis techniques. In: Schmid, U., Kitzelmann, E., Plasmeyer, R. (eds.) AAIP 2009. LNCS, vol. 5812, pp. 50–73. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-11931-6\\_3](https://doi.org/10.1007/978-3-642-11931-6_3)
12. McKay, R., Hoai, N., Whigham, P., Shan, Y., O’Neill, M.: Grammar-based genetic programming: a survey. *Genet. Program. Evol. Mach.* **11**(3–4), 365–396 (2010)
13. O’Neill, M., Nicolau, M., Agapitos, A.: Experiments in program synthesis with grammatical evolution: a focus on integer sorting. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1504–1511, July 2014
14. O’Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers, Norwell (2003)
15. Pantridge, E., Helmuth, T., McPhee, N.F., Spector, L.: On the difficulty of benchmarking inductive program synthesis methods. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2017, pp. 1589–1596. ACM, New York (2017)
16. Saber, T., Fagan, D., Lynch, D., Kucera, S., Claussen, H., O’Neill, M.: Multi-level grammar genetic programming for scheduling in heterogeneous networks. In: Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S., García-Sánchez, P. (eds.) EuroGP 2018. LNCS, vol. 10781, pp. 118–134. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77553-1\\_8](https://doi.org/10.1007/978-3-319-77553-1_8)
17. Spector, L., Robinson, A.: Genetic programming and autoconstructive evolution with the push programming language. *Genet. Program. Evol. Mach.* **3**(1), 7–40 (2002)
18. Helmuth, T., Spector, L.: Detailed problem descriptions for general program synthesis benchmark suite. Technical report, School of Computer Science, University of Massachusetts Amherst (2015)
19. Whigham, P.A.: Grammatical bias for evolutionary learning. Ph.D. thesis, University of New South Wales, Australia (1996)
20. Yu, T.: A higher-order function approach to evolve recursive programs. In: Yu, T., Riolo, R., Worzel, B. (eds.) Genetic Programming Theory and Practice III. GPTEM, pp. 93–108. Springer, Boston (2006). [https://doi.org/10.1007/0-387-28111-8\\_7](https://doi.org/10.1007/0-387-28111-8_7)



# Filtering Outliers in One Step with Genetic Programming

Uriel López<sup>1,2</sup>, Leonardo Trujillo<sup>1,2</sup>(✉) , and Pierrick Legrand<sup>3,4,5</sup>

<sup>1</sup> Tecnológico Nacional de México/I.T. Tijuana, Tijuana, BC, Mexico  
{uriel.lopez,leonardo.trujillo}@tectijuana.edu.mx

<sup>2</sup> BioISI, Faculty of Sciences, University of Lisbon, Lisbon, Portugal

<sup>3</sup> IMB, UMR CNRS 5251, 351 cours de la libération, Talence, France

<sup>4</sup> Inria Bordeaux Sud-Ouest, Talence, France

<sup>5</sup> University of Bordeaux, Bordeaux, France

pierrick.legrand@u-bordeaux.fr

**Abstract.** Outliers are one of the most difficult issues when dealing with real-world modeling tasks. Even a small percentage of outliers can impede a learning algorithm's ability to fit a dataset. While robust regression algorithms exist, they fail when a dataset is corrupted by more than 50% of outliers (breakdown point). In the case of Genetic Programming, robust regression has not been properly studied. In this paper we present a method that works as a filter, removing outliers from the target variable (vertical outliers). The algorithm is simple, it uses a randomly generated population of GP trees to determine which target values should be labeled as outliers. The method is highly efficient. Results show that it can return a clean dataset when contamination reaches as high as 90%, and may be able to handle higher levels of contamination. In this study only synthetic univariate benchmarks are used to evaluate the approach, but it must be stressed that no other approaches can deal with such high levels of outlier contamination while requiring such small computational effort.

**Keywords:** Outliers · Robust regression · Genetic programming

## 1 Introduction

The main application domain for Genetic Programming (GP) continues to be symbolic regression. The ability of GP to model difficult non-linear problems, and to produce relatively compact models when appropriate bloat control is used [1], makes it a good option in this common machine learning task. Unlike random ensemble regression, SVM regression or Neural Networks, for example, GP has the potential of delivering human-readable solutions that are also accurate and efficient. However, like any data-driven approach to modeling, much of the quality of the final solution will be determined by the nature of the training data; i.e.; even the best algorithm cannot produce a model when the output

variable shows no relationship to the input variables. One particularly difficult case in learning is when the training data is corrupted by *outliers*. Indeed, outlier data points can severely skew the learning process and bias the search towards unwanted regions in solution space.

A common way to deal with this situation is to apply a filtering process or to use a robust objective measure, such that performance estimation is not affected by the presence of outlier points [2]. Such methods can handle outliers effectively under the assumption that they are relatively rare; i.e. outliers represent a minority of the data points in the dataset. However, this work considers an extreme case; where the percentage of outliers far outnumbers the inliers, reaching as high as 90% of the entire training set<sup>1</sup>. In this case, even the most robust objective function cannot properly guide a learning algorithm, and it is the same for GP.

One way to deal with outliers is to use specialized sampling techniques such as Random Sampling Consensus (RANSAC) [3], which is often used in computer vision systems for data calibration [4]. Indeed, this method has been successfully combined with GP to derive accurate models even when the data contamination is above the breakdown point; i.e., the number of outliers is above 50%. However, a noteworthy drawback of such a method is that its computational cost is extremely high, with the number of expected samples required to build a sufficiently accurate model increasing exponentially with the total contamination in the dataset. However, and this cannot be stressed enough, in current literature no other methods exist that can deal with such extreme cases of data contamination in an automatic manner [5–7]. In many cases, the best suggestion is to simply inspect the data visually and clean it manually. In the case of GP, for instance, such conditions are never even tested, not even for robust GP systems [8–10].

The present work fills this notable gap by presenting an approach to clean highly contaminated datasets in regression tasks. Particularly, this work considers the case where the output variable is highly contaminated by outliers, measurements that deviate sharply from the true signal that is trying to be modeled. The proposed method can be used as a preprocessing step to clean the data, applied before the actual modeling process is performed. The method is efficient, requiring the same amount of computational effort that is required to evaluate a single GP population. It uses a single randomly generated population to determine which data points are outliers and which are not. Besides this effort, all that is required is to order the population based on fitness, and with that an efficient criterion for cleaning the data is proposed. The method can be integrated not just with GP, but with any regression method. If used with GP, however, then the outlier removal process is obtained basically for free since the initial population of the GP search could be used to perform the filtering at generation 0.

---

<sup>1</sup> In fact, while the reported experiments only consider up to this level of data contamination, it is straightforward to extend our approach to more severe scenarios.



The proposed method operates under the assumption that outlier points are more difficult to model than inliers, when the models (GP trees) are generated randomly. While we do not derive any formal proofs that show that this assumption will hold in general, the experimental work confirms that this assumption is valid for the set of test cases used to evaluate the proposal. We test the algorithm on datasets that are contaminated by as much as 90% of outliers, and are able to remove a sufficiently large proportion of the outlier instances, it then becomes feasible for a standard robust regression method to tackle the problem.

The remainder of this paper proceeds as follows. Section 2 reviews basic concepts on robust regression and outlier detection. Then, Sect. 3 presents our outlier filter, the reader will notice that the most important characteristic of the proposal is its simplicity; the section also reviews related works. Section 4 presents the experimental work, following the general methodology of [2]. Finally, Sect. 5 provides a discussion, outlines our main conclusions and describes future work.

## 2 Background

### 2.1 Outliers

All regression and automatic modeling systems are heavily influenced by the presence of anomalies in the training data [7]. These anomalies are usually referred to as outliers, and can be present in the input variables, the output variable, or in both. Outliers can be generated by several causes, such as gross human error, equipment malfunction, extremely severe random noise or missing data [5–7]. When outliers are rare, then it is possible to define them as data points that are very different from the rest of the observations included in the dataset. However, such a definition is not useful when the number of outliers exceeds the number of inliers (non-outlier data points). Moreover, it is important to distinguish between outliers and just signal noise. In our opinion, the two most important distinctions are: (1) noise can be effectively modelled, and thus filtered; and (2) outlier points deviate from inliers at a large scale, i.e. outliers are anomalous w.r.t. the inliers, which is not a formal definition but in practice it is a useful one. Therefore, since in this work we are concerned with the presence of outliers in the output or target variable in regression problems (also called vertical outliers), we can use the following definition for outliers [2]:

**Definition 1.** *An outlier is a measurement of a system that is anomalous with respect to the true behavior of the system.*

While this definition may be seen as a tautology, there is an aspect of it that is not immediately obvious. Notice that we are defining an outlier relative to the “true behavior of the system”, whether this behavior is observable or not. The definition is not based on the observed behavior in a representative dataset from which we can pose a regression or learning task. This is crucial, because it may seem counter intuitive to have a dataset where the majority of samples are in fact outliers. However, if we know the true behavior of a system, in a controlled

experiment with synthetic problems, it is straightforward to build a dataset where the majority of points are outliers based on Definition 1. Moreover, such a scenario is often encountered in real-world problems as well, one well-known domain is computer vision [4].

### 2.2 Robust Regression

First, let us define the standard regression problem. Given a training dataset  $\mathbb{T} = \{(\mathbf{x}_i, y_i); i = 1, \dots, n\}$ , the goal is to derive a model that predicts  $y_i$  based on  $\mathbf{x}_i$ , where  $\mathbf{x}_i \in \mathbb{R}^p$  and  $y_i \in \mathbb{R}$ . In GP literature we can refer to each input/output pair  $(\mathbf{x}_i, y_i)$  as a fitness case, a training instance or a data point. For linear regression, the model is expressed as

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i \quad i = 1, \dots, n \tag{1}$$

where the model parameters  $\beta = (\beta_0, \beta_1, \dots, \beta_p) \in \mathbb{R}^{p+1}$ , can be estimated by  $\widehat{\beta}_0, \dots, \widehat{\beta}_p$  using the least squares method [11], which can be expressed as

$$(\widehat{\beta}_0, \dots, \widehat{\beta}_p) \leftarrow \underset{\beta \in \mathbb{R}^{p+1}}{\arg \min} \sum_{i=1}^n r_i^2, \tag{2}$$

to find the best fit parameters of the linear model, where  $r_i$  denotes the residuals  $r_i(\widehat{\beta}_0, \dots, \widehat{\beta}_p) = y_i - (\widehat{\beta}_0 + \widehat{\beta}_1 x_{i1} + \dots + \widehat{\beta}_p x_{ip})$  and the errors  $\varepsilon_i$  have an expected value of zero [12]; if the summation in Eq. 2 is divided by  $n$ , the error measure that must be minimized is the mean squared error (MSE). The issue with outliers is that they bias the standard objective measures defined above (and others, such as regularized approaches). In classical regression, there are several robust regression methods that deal with the presence of outliers by modifying the objective function used to perform the regression. For instance, Least Median Squares (LMS) [13]

$$(\widehat{\beta}_0, \dots, \widehat{\beta}_p) \leftarrow \underset{\beta_0, \dots, \beta_p}{\arg \min} \text{med} \{r_1^2, \dots, r_n^2\} \tag{3}$$

where *med* represents the median. Another approach is Least Trimmed Squares (LTS) [13], given by

$$(\widehat{\beta}_0, \dots, \widehat{\beta}_p) \leftarrow \underset{\beta_0, \dots, \beta_p}{\arg \min} \sum_{i=1}^{hp} \{r_1^2, \dots, r_n^2\}_{i:n}. \tag{4}$$

where  $hp$  with  $p \leq hp \leq n$  is typically set to  $hp = (n + p + 1)/2$  for maximum breakdown point, and  $p$  (size of the sample) is an algorithm parameter. In the case of LMS, the idea is to use the median of the residuals instead of an aggregate fitness such as the average error. A generalization of this method is quantile regression [14]. Moreover, similar approaches have been applied with more sophisticated regression methods, such as random decision trees [15]. LTS

searches for a subset of training cases that give the lowest error, since the lowest error will be obtained when only inliers are present in the subset. Moreover, there is an efficient implementation of this algorithm called FAST-LTS; a review of robust methods can be found in [16]. These methods are indeed robust for linear regression, but only when the number of outliers does not exceed 50% of the training data, which is referred to as the breakdown point of the method. Beyond this breakdown point, these methods also fail, but consider that standard LS has a breakdown point of 0 and theoretically the 50% breakdown cannot be exceeded for linear regression problems. Moreover, recently it was shown that combining LMS and LTS with GP can allow it to solve symbolic regression problems with the same order of accuracy when the dataset is contaminated by as much as 50% of outliers, empirically showing that their breakdown point holds in symbolic regression with GP [2].

For problems where the contamination of the dataset is above 50%, sampling and approximate methods must be used. For instance, one approach is RANSAC [3], a sampling method to solve parameter estimation problems where the contamination level exceeds 50%. RANSAC has proven to be very useful in at least one domain, computer vision [4], and many different version of the algorithm have been derived such as the M-estimator Sample Consensus (MSAC), the Maximum Likelihood Estimation Sample and Consensus (MLESC) [17], and the Optimal RANSAC algorithm [18]. Moreover, [2] has also shown that combining RANSAC with GP can achieve robustness in symbolic regression modeling in extreme contamination scenarios, with empirical evidence presented for up to 90% contamination by outliers. The main drawback with RANSAC is that it requires multiple random samples of the dataset, which have a low probability of being sufficiently clean (composed primarily by inliers) when the original dataset is highly contaminated, making a computationally expensive method<sup>2</sup>.

### 3 Outlier Removal with Genetic Programming

Before describing the proposed algorithm, we must first state the main assumption on which it is based. Consider a training set  $\mathbb{T} = \{(\mathbf{x}_i, y_i)\}$  where some fitness cases are inliers and other are outliers, where the  $l$ -th fitness case represents an outlier while the  $j$ -th fitness case represents an inlier. The assumption is that for a randomly generated GP tree (model or program)  $K$ , there is a high probability that the residual  $r_l$  is larger than the residual  $r_j$ . In other words, the residuals on the outliers will be larger than the residuals on the inliers for randomly generated models. While not at all obvious, there are some clear motivations for this assumption. First, random GP trees will only be able to detect simple and coarse relationships between the input and output variables, what can also be considered to be as low-frequencies in the signal. On the other hand, outliers will mostly appear as singularities in the training data, or high-frequency

<sup>2</sup> While it would be relatively simple to parallelize the algorithm, since all samples are taken independently, the cost can still become quite high if the modelling is done with GP.

components. Second, it is conceivable that a particular program might actually produce a low residual for one (or a few) outlier(s), and in this cases the assumption will not hold. However, since outliers do not follow a particular model (they are not noise), then the residuals in all other outliers can be expected to be relatively high. Finally, even if the majority of points in the training set are outliers this assumption can be expected to hold since the models are not fitted to the training data; i.e. the GP trees do not *learn* the outliers since they are randomly generated.

In what follows we will define an algorithm for detecting outliers based on this assumption and validate it in the experimental work reported afterward.

### 3.1 Proposed Algorithm

Based on the previous assumption, the proposed filtering process is summarized in Algorithm 1. The main inputs are the training set  $\mathbb{T}$  of size  $n$ , and a percentile parameter  $\rho$  which defines the percentage of fitness cases that will be returned. In step 1, Ramped Half and Half is used to generate a total of  $p$  GP trees, using a specified function set  $F$  and a maximum depth  $d$ . The terminal set required to generate the random models is always composed by the input variables and randomly generated constants in the range  $[-1, 1]$ . Several informal tests showed that the method was quite robust to parameters  $d$ ,  $p$  and  $F$ .

In step 2, the residuals of each GP tree on each fitness case is computed, constructing the matrix of residuals  $R_{p \times n}$ , where each element  $r_{i,j}$  is the residual from the  $i$ -th model  $K_i$  (GP tree) on the  $j$ -th training instance  $x_j \in \mathbb{T}$ .

Step 3 is the key step, where the information contained in  $R_{p \times n}$  is used to sort the training set and identify outliers, working under the assumption that outliers will have higher associated residuals for most GP trees. Therefore, we compute the column wise median of  $R_{p \times n}$ , generating a vector  $V$  of size  $n$  containing the median residual of each training instance evaluated over all random models. Therefore, set  $C$  will contain the  $\rho\%$  of training instances from  $\mathbb{T}$  that have the lowest associated median residuals.

---

#### Algorithm 1. Proposed algorithm for outlier removal.

---

Input: Contaminated training set  $\mathbb{T}$  of size  $n$ .

Input: Cut-off percentile  $\rho$  in  $(0, 1]$ .

Input: Number of GP trees  $p$ .

Input: Function set  $F$  and model size parameter  $d$ .

Output: Set  $C \subset \mathbb{T}$  of inliers.

1. Generate a random set  $P$  of models  $k : \mathbb{R}^m \rightarrow \mathbb{R}$ , with  $|P| = p$  using  $F$  and  $d$ .
  2. Obtain the matrix of residuals  $R_{p \times n}$  such that each  $r_{i,j}$  is the residual from each model  $k_i \in P$  and each training instance  $x_j \in \mathbb{T}$ .
  3. Sort  $\mathbb{T}$  based on the column wise median vector of  $R_{p \times n}$ , and return the lowest  $\rho\%$  training instances in set  $C$ .
-

### 3.2 Discussion

There are two general strategies to deal with outliers. The first approach is to use the regression process to detect outliers and to basically build a model while excluding the outliers. This approach is taken by most of the robust techniques described above, such as LMS, LTS and even RANSAC, since the determination of which points are outliers depends on obtaining the residuals from a fitted model.

The second approach is to use a filtering process. A particularly well known filter is the Hampel identifier, where a data point  $(x_i, y_i)$  is tagged as an outlier if

$$|y_i - y_o| > t\zeta \quad (5)$$

where  $y_i$  is the value to be characterized,  $y_o$  is a reference value,  $\zeta$  is a measure of data variation, and  $t$  is a user defined threshold [7]. The Hampel identifier uses a window  $W$  centered on  $x_i$  to compute  $y_o$  and  $\zeta$ , with  $y_o$  set to the median of all  $y_j$  in  $W$  and  $\zeta$  is  $1.4826 \times \text{MAD}$  (Mean Absolute Deviation) within  $W$ ; the value 1.4826 is chosen so that the results are not biased towards a Gaussian distribution.

The proposed method can be considered to be a hybrid between these two approaches. On the one hand, it is meant as a preprocessing step, used to remove outliers before another learning algorithm is applied to the data, thus it can be considered to be a filter. On the other hand, it is also based on the residuals computed for each training instance. However, unlike other robust methods, the residuals are derived from a random sampling of models, basically a population of GP trees, and learning or parameter fitting is not performed at all.

### 3.3 Related Works in GP

As stated above, [2] presents several results that are relevant to robust regression in GP. That work showed that both LMS and LTS are applicable to GP, and empirically their breakdown also applies to GP. Also, given the general usefulness of sampling the training instances to perform robust regression [16], that work also tested the applicability of sampling techniques in GP, such as interleaved sampling [19] and Lexicase selection [20]. Results showed that none of those approaches were useful for robust regression. The best results were obtained using RANSAC for sampling the training set and applying LMS on each selected subset, achieving almost equal test set prediction than directly learning on a clean training set. The method was called RANSAC-GP. The main drawback of RANSAC-GP is the high computational cost, since GP had to be executed on each sample and many samples were required as the percentage of outliers increases. Moreover, one underlying assumption of RANSAC-GP is that the GP search will be able to find a fairly accurate model on a clean subset of training examples, since models obtained from different samples will be discriminated based on their training performance. This assumption might not hold for some real-world problems.

Robust GP regression has not received much attention in GP, but some works are notable. In [9] GP and Geometric Semantic Genetic Programming (GSGP) are compared to determine which method was more sensitive to noisy data. The training sets are corrupted with Gaussian noise, up to a maximum of 20% of the training instances, concluding that GSGP is more robust when the contamination is above 10%. However, outliers are not considered. Another example is [10], in this case focusing on classification problems with GP-based multiple feature construction when the data set is incomplete, which can also be considered to be outliers. The proposed method performs well, even when there is up to 20% of missing data, but extreme cases such as the ones tested here are not reported. A more related work is [21], where the authors build ensembles of GP models evolved using a multiobjective approach, where both accuracy and program size are minimized. The proposed approach is based on the same general assumption of many techniques intended to be robust to outliers, that model performance will be worse on outlier points than inliers. The ensembles are built from hundreds of independent GP runs, a process that is much more expensive than the one proposed in the present work. Moreover, results are only presented for a single test case, where it is not known how many outliers are present, but results indicate that it is not higher than 5%. The method also requires human interpretation and analysis of the results, while the method proposed in this work is mostly automated except for the algorithm parameters.

## 4 Experimental Evaluation

### 4.1 Experimental Setup

As a first experimental test, we use the same procedure followed in [2]. First, we use the synthetic problems defined in Table 1. The datasets for each problem consist of 200 data points; i.e. input/output pairs of the form  $(x_i, y_i)$ . The independent variable (input) was randomly sampled using a uniform distribution within the domain of each problem (see Table 1), and the corresponding value of the dependent variable (output) was then computed with the known model syntax. These represent the clean data samples or inliers of each problem. Then, these datasets were contaminated by different amounts of outliers, from 10% to 90% contamination in increments of 10%, for each. Thus, for each problem we have nine different datasets, each with a different amount of outliers. The proposed method is executed 30 times on each dataset, for each problem and for each level of contamination, to evaluate the robustness of the approach. To turn a particular fitness case  $(x_i, y_i)$  into an outlier, we first solve inequality 5 for  $y_i$ , such that

$$\begin{aligned} y_i &> y^o + t\zeta \\ \text{or } y_i &< y^o - t\zeta. \end{aligned} \tag{6}$$

The decision to add or subtract from  $y_i$ , as defined in Eq. 5, is done randomly, and the value of  $t$  is set randomly within the range [10, 100] to guarantee a large

**Table 1.** Benchmark problems used in this work, where  $U[a, b, c]$  denotes  $c$  uniform random samples drawn from  $a$  to  $b$ , that specifies how the initial training sets are constructed consisting solely of inliers.

Objective function	Training set
$x^4 + x^3 + x^2 + x$	$U[-1, 1, 200]$
$x^5 - 2x^3 + x$	$U[-1, 1, 200]$
$x^3 + x^2 + x$	$U[-1, 1, 200]$
$x^5 + x^4 + x^3 + x^2 + x$	$U[-1, 1, 200]$
$x^6 + x^5 + x^4 + x^3 + x^2 + x$	$U[-1, 1, 200]$

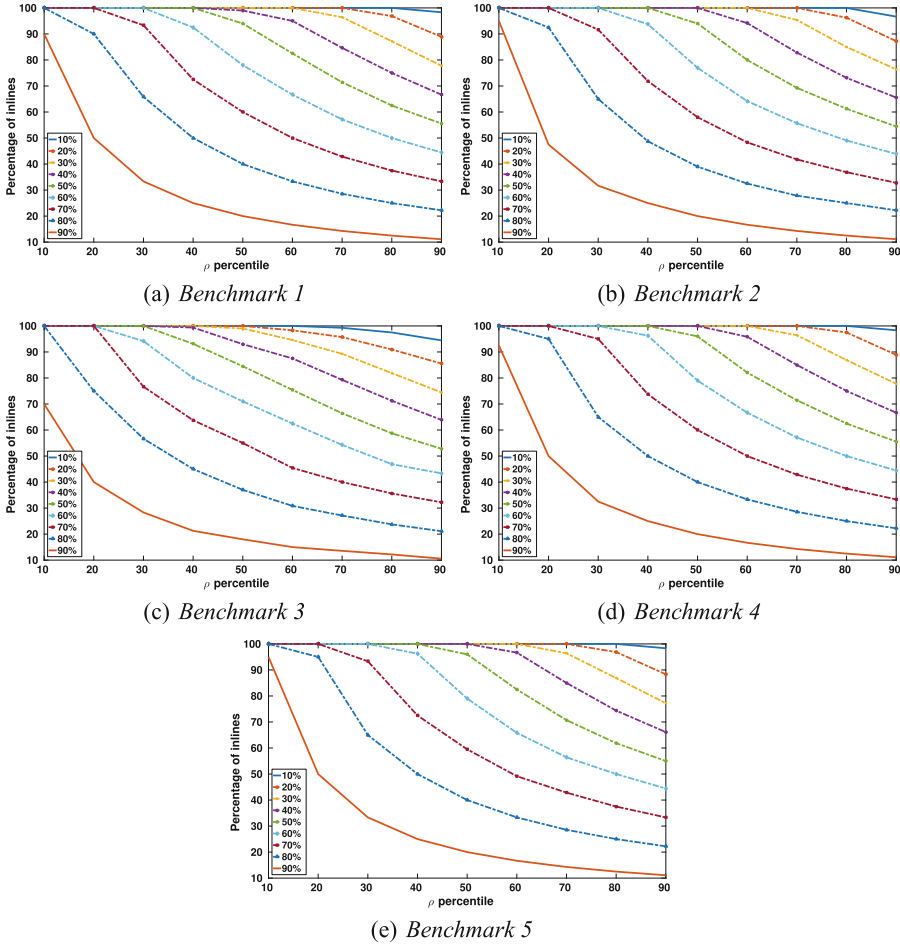
amount of deviance from the original data, with  $\zeta$  computed by the median of all  $y_i$  within the function domain of each symbolic regression benchmark.

The parameters for the proposed method are set as follows. The function set is given by  $F = \{+, -, \times, \div, \sin, \cos\}$  where  $\div$  is the protected division, the maximum tree depth is set to  $d = 3$ , and the number of randomly generated models is  $p = 100$ . The percentile parameter  $\rho$  is evaluated from 10% to 90% in 10% increments. The method was coded using the Distributed Evolutionary Algorithms in Python library (DEAP) [22], basically building on top of the population initialization function.

## 4.2 Results

Figure 1 presents the main results. In each plot, the horizontal axis corresponds to the value of the  $\rho$  parameter, while the vertical axis represents the level of contamination in the output set  $C$ . In other words, the vertical axis shows the percentage of inliers contained in the *clean* set  $C$ , which in the best case would be 100%. However, it is important to remember, particularly when the contamination is above 50%, that a desired goal is for the vertical axis to be as high as possible, but in practice it can be sufficient if it is above 50%. In such a case it would be possible to use a robust regression method to solve the resulting modeling problem with set  $C$ . Each plot corresponds to one of the benchmarks from Table 1, and each shows nine curves, one for each contamination level. Each curve corresponds to the median performance over all 30 executions on each of the contaminated training sets.

All of the curves show a regular and informative pattern. First, on each problem the top curve corresponds to the lowest level of contamination 10%. As  $\rho$  increases, more points are returned as possible inliers but might in fact be outliers; i.e.,  $C$  is larger, therefore the probability of the set being completely clean gradually declines. While the 10% level of contamination seems rather low in our tests, it is far above the breakdown point of non-robust regression methods. However, for this simplest case the percentage of inliers never falls below 90%. Second, as the level of contamination increases the performance on each problem gradually degrades, but not in a significant manner. Take for



**Fig. 1.** Performance on the benchmark problems. The horizontal axis corresponds to the percentile parameter  $\rho$ , and the vertical axis represents the percentage of inliers in the resulting clean set  $C$ . Each curve represents the median value performance over 30 independent runs for each level of contamination.

instance the most extreme case, when contamination is at the 90% level. Using a conservative value for  $\rho$  of only 10%, the set returned contains a high amount of inliers. In 4 problems it is above 90% and in only one case it falls to about 70%. In this latter case, Benchmark 3, this means that the new training set  $C$  now contains only 30% of outliers instead of the original 90%. This is useful, since it is now possible to build a model using a robust regression approach, such as LMS or LTS. For all other contamination levels, the performance is even more encouraging. For example, for contamination at 80% or lower it would be possible to set  $\rho = 30\%$  and produce a clean dataset that contains less than 40% of outliers. These are highly encouraging results, showing that the proposed



**Table 2.** Median performance on Benchmark 1, shown as the percentage of inliers in the returned clean set  $C$ ; bold values represent the level of contamination where the number of detected inliers falls below 100%.

Outliers	$\rho$ value								
	$\rho = 10$	$\rho = 20$	$\rho = 30$	$\rho = 40$	$\rho = 50$	$\rho = 60$	$\rho = 70$	$\rho = 80$	$\rho = 90$
10%	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	<b>98.3</b>
20%	100.0	100.0	100.0	100.0	100.0	100.0	100.0	<b>96.8</b>	88.8
30%	100.0	100.0	100.0	100.0	100.0	100.0	<b>96.4</b>	87.1	77.7
40%	100.0	100.0	100.0	100.0	99.0	<b>95.0</b>	84.6	75.0	66.6
50%	100.0	100.0	100.0	100.0	<b>94.0</b>	82.5	71.4	62.5	55.5
60%	100.0	100.0	100.0	<b>92.5</b>	78.0	66.6	57.1	50.0	44.4
70%	100.0	100.0	<b>93.3</b>	72.5	60.0	50.0	42.8	37.5	33.3
80%	100.0	<b>90.0</b>	65.8	50.0	40.0	33.3	28.5	25.0	22.2
90%	<b>90.0</b>	50.0	33.3	25.0	20.0	16.6	14.2	12.5	11.1

**Table 3.** Median performance on Benchmark 3, shown as the percentage of inliers in the returned clean set  $C$ ; bold values represent the level of contamination where the number of detected inliers falls below 100%.

Outliers	$\rho$ value								
	$\rho = 10$	$\rho = 20$	$\rho = 30$	$\rho = 40$	$\rho = 50$	$\rho = 60$	$\rho = 70$	$\rho = 80$	$\rho = 90$
10%	100.0	100.0	100.0	100.0	100.0	100.0	99.2	97.5	<b>94.4</b>
20%	100.0	100.0	100.0	100.0	100.0	98.3	95.7	<b>90.9</b>	85.5
30%	100.0	100.0	100.0	100.0	99.0	94.5	<b>89.2</b>	81.8	74.4
40%	100.0	100.0	100.0	99.3	93.0	<b>87.5</b>	79.2	71.2	63.8
50%	100.0	100.0	100.0	93.1	<b>84.5</b>	75.4	66.4	58.7	52.7
60%	100.0	100.0	94.1	<b>80.0</b>	71.0	62.5	54.2	46.8	43.3
70%	100.0	100.0	<b>76.6</b>	63.7	55.0	45.4	40.0	35.6	32.2
80%	100.0	<b>75.0</b>	56.6	45.0	37.0	30.8	27.1	23.7	21.1
90%	<b>70.0</b>	40.0	28.3	21.2	18.0	15.0	13.5	12.1	10.5

method can identify outliers fairly easily using the proposed configuration. To better grasp these results, the numerical results for Benchmark 1 and Benchmark 3 are respectively shown in Tables 2 and 3. In each table, the bold value indicates when the median percentage of returned inliers falls below 100%.

## 5 Conclusions and Future Work

Dealing with outliers is a notoriously hard problem in regression. The algorithm presented in this work can effectively clean highly contaminated datasets. Standard regression techniques breakdown with even a single outlier in the training

set, while robust regression techniques fail when the contamination by outliers is greater than 50% on the training set. In such a cases, sampling techniques such as RANSAC are required, but the number of samples required grows rapidly with the percentage of outliers.

The proposed algorithm uses a random GP population to determine which training instances are inliers and which are not. It works under the assumption that outliers will be more difficult to model for randomly generated GP trees than inliers are; i.e. the residuals on outliers will be larger than on inliers. While robust regression methods also work under this assumption, this only holds after the model has been tuned, after learning has been performed. Moreover, this will only be possible if outliers represent a minority in the training set. On the other hand, the proposed algorithm does not perform any learning, basing its decision entirely on a random set of models. The proposed algorithm seems related to several other machine learning approaches. As stated above, it is obviously related to robust regression methods, particularly quantile regression, but without performing any model fitting. It is also related to RANSAC, since it performs a random sampling, but of models instead of training instances.

Results are encouraging, compared to other methods, only RANSAC can attempt to deal with problems where the level of contamination exceeds 50%. Take for instance the Hampel identifier, it would be useless since the median value in the dataset would be an outlier. Moreover, while RANSAC can deal with similar problems, its computational cost can become excessive and depends on the ability of the learning or modeling algorithm to extract relatively accurate models [2]. The proposed method is efficient, since it only requires generating and evaluating a single GP population.

Future work will focus on the following. First, extend the evaluation to real-world multi-variate problems, a more challenging scenario. Second, determine how specific parameters of the proposed algorithm affect performance, particularly the number of random models generated. Third, attempt to determine a *general* setting for  $\rho$ , at least experimentally. Fourth, clearly define how the proposed algorithm relates to other robust regression and learning algorithms. Finally, extend the method to deal with outliers in the input variables.

**Acknowledgments.** This research was funded by CONACYT (Mexico) Fronteras de la Ciencia 2015-2 Project No. FC-2015-2:944, BioISI R&D unit, UID/MULTI/04046/2013 funded by FCT/MCTES/PIDDAC, Portugal, and first author supported by CONACYT graduate scholarship No. 573397.

## References

1. Trujillo, L., et al.: Neat genetic programming: controlling bloat naturally. *Inf. Sci.* **333**, 21–43 (2016)
2. López, U., Trujillo, L., Martínez, Y., Legrand, P., Naredo, E., Silva, S.: RANSAC-GP: dealing with outliers in symbolic regression with genetic programming. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E., García-Sánchez, P. (eds.) *EuroGP 2017*. LNCS, vol. 10196, pp. 114–130. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55696-3\\_8](https://doi.org/10.1007/978-3-319-55696-3_8)

3. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **24**(6), 381–395 (1981)
4. Hartley, R.I., Zisserman, A.: *Multiple View Geometry in Computer Vision*, 2nd edn. Cambridge University Press, Cambridge (2004). ISBN 0521540518
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. *ACM Comput. Surv.* **41**(3), 15:1–15:58 (2009)
6. Hodge, V.J., Austin, J.: A survey of outlier detection methodologies. *Artif. Intell. Rev.* **22**, 85–126 (2004)
7. Pearson, R.: *Mining Imperfect Data: Dealing with Contamination and Incomplete Records*. Society for Industrial and Applied Mathematics. SIAM, Philadelphia (2005)
8. Kotanchek, M.E., Vladislavleva, E.Y., Smits, G.F.: Symbolic regression via genetic programming as a discovery engine: insights on outliers and prototypes. In: Riolo, R., O’Reilly, U.M., McConaghy, T. (eds.) *Genetic Programming Theory and Practice VII. Genetic and Evolutionary Computation*, pp. 55–72. Springer, Boston (2010). [https://doi.org/10.1007/978-1-4419-1626-6\\_4](https://doi.org/10.1007/978-1-4419-1626-6_4)
9. Miranda, L.F., Oliveira, L.O.V.B., Martins, J.F.B.S., Pappa, G.L.: How noisy data affects geometric semantic genetic programming. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017*, pp. 985–992. ACM, New York (2017)
10. Tran, C.T., Zhang, M., Andreae, P., Xue, B.: Genetic programming based feature construction for classification with incomplete data. In: *Proceedings of the Genetic and Evolutionary Computation Conference. GECCO 2017*, pp. 1033–1040. ACM, New York (2017)
11. Rousseeuw, P.J.: Least median of squares regression. *J. Am. Stat. Assoc.* **79**(388), 871–880 (1984)
12. Alfons, A., Croux, C., Gelper, S.: Sparse least trimmed squares regression for analyzing high-dimensional large data sets. *Ann. Appl. Stat.* **7**(1), 226–248 (2013)
13. Giloni, A., Padberg, M.: Least trimmed squares regression, least median squares regression, and mathematical programming. *Math. Comput. Model.* **35**(9), 1043–1060 (2002)
14. Bertsimas, D., Mazumder, R.: Least quantile regression via modern optimization. *ArXiv e-prints* (2013)
15. Meinshausen, N.: Quantile regression forests. *J. Mach. Learn. Res.* **7**, 983–999 (2006)
16. Hubert, M., Rousseeuw, P.J., Van Aelst, S.: *Statist. Sci. High-breakdown robust multivariate methods* **23**, 92–119 (2008)
17. Torr, P.H., Zisserman, A.: MLESAC: a new robust estimator with application to estimating image geometry. *Comput. Vis. Image Underst.* **78**(1), 138–156 (2000)
18. Hast, A., Nysj , J., Marchetti, A.: *Optimal RANSAC-towards a repeatable algorithm for finding the optimal set* (2013)
19. Gonalves, I., Silva, S.: Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A.Ş., Hu, B. (eds.) *EuroGP 2013. LNCS*, vol. 7831, pp. 73–84. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-37207-0\\_7](https://doi.org/10.1007/978-3-642-37207-0_7)
20. Spector, L.: Assessment of problem modality by differential performance of lexibase selection in genetic programming: a preliminary report. In: *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference Companion, GECCO Companion 2012*, pp. 401–408. ACM (2012)

21. Kotanchek, M., Smits, G., Vladislavleva, E.: Pursuing the pareto paradigm: tournaments, algorithm variations and ordinal optimization. In: Riolo, R., Soule, T., Worzel, B. (eds.) Genetic Programming Theory and Practice IV. Genetic and Evolutionary Computation. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-0-387-49650-4\\_11](https://doi.org/10.1007/978-0-387-49650-4_11)
22. Fortin, F.A.: DEAP: evolutionary algorithms made easy. *J. Mach. Learn. Res.* **13**, 2171–2175 (2012)



# GOMGE: Gene-Pool Optimal Mixing on Grammatical Evolution

Eric Medvet<sup>(✉)</sup>, Alberto Bartoli, Andrea De Lorenzo, and Fabiano Tarlao

Department of Engineering and Architecture, University of Trieste, Trieste, Italy  
emedvet@units.it

**Abstract.** Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) is a recent Evolutionary Algorithm (EA) in which the interactions among parts of the solution (i.e., the linkage) are learned and exploited in a novel variation operator. We present GOMGE, the extension of GOMEA to Grammatical Evolution (GE), a popular EA based on an indirect representation which may be applied to any problem whose solutions can be described using a context-free grammar (CFG). GE is a general approach that does not require the user to tune the internals of the EA to fit the problem at hand: there is hence the opportunity for benefiting from the potential of GOMEA to automatically learn and exploit the linkage. We apply the proposed approach to three variants of GE differing in the representation (original GE, SGE, and WHGE) and incorporate in GOMGE two specific improvements aimed at coping with the high degeneracy of those representations. We experimentally assess GOMGE and show that, when coupled with WHGE and SGE, it is clearly beneficial to both effectiveness and efficiency, whereas it delivers mixed results with the original GE.

**Keywords:** Genetic programming · Linkage · Family of Subsets Representation

## 1 Introduction

Evolutionary Algorithms (EAs) are a powerful tool for solving complex problems. One motivation for their wide adoption is that the user is not required to provide a model for the problem at hand: in most cases, it is up to the EA to figure out how the parts of the solution (w.r.t. the representation employed in that EA) interact in determining the solution quality. However, actually knowing the model and being able to exploit its knowledge may be crucial to determine the effectiveness of the EA.

A model-based EA has been recently proposed for achieving both goals, i.e., the ability to know and exploit the model without requiring any user-provided specification of the model itself. The Gene-pool Optimal Mixing Evolution Algorithm (GOMEA) [1] is a state-of-the-art approach for solving discrete optimization problems and has been carefully designed for exploiting the interactions

among parts of a solution, i.e., the *linkage*. GOMEA is based on several crucial contributions: an internal representation of the linkage; a method for deriving the linkage from the population; a novel genetic operator (Gene-pool Optimal Mixing, GOM) in which the individual iteratively receives from random donors some portions of the genetic material defined by the linkage.

We here present GOMGE, i.e., the extension of GOMEA to Grammatical Evolution (GE) [2]—a form of Genetic Programming (GP). GE particularly fits the two aforementioned goals pursued by GOMEA, because it is really a general purpose EA. In fact, key for GE success is that it can tackle any problem whose solutions may be described by means of a context-free grammar (CFG). The user is hence not required to know and tune the internals of the EA: he can obtain a solution for the problem at hand by simply providing the CFG and a fitness function. Indeed, GE has been widely used in many different applications: e.g., generation of string similarity indexes suitable for text extraction [3], road traffic rules synthesis [4], automatic design of analog electronic circuits [5], and even the design of other optimization algorithms [6].

Internally, GE operates on individuals described with an indirect representation: genetic operators are applied to bit-string genotypes; then, bit-strings are transformed into solutions (i.e., strings of the language defined by the problem-specific CFG) by means of a genotype-phenotype mapping function. The latter, which essentially defines the individual representation of GE, favored the adoption of this EA, since it allowed building on the vast knowledge about manipulation of bit-string genotypes. On the other hand, extensive research on the properties of GE representation showed that it has many drawbacks [7–9]. Indeed, beyond inspiring a large debate among scholars which also concerned about the aims and methods for designing an EA representation [10–12], the drawbacks of GE representation also stimulated the recent arising of two variants—Structured GE (SGE) [13] and Weighted Hierarchical GE (WHGE) [14]—mainly consisting in a different genotype-phenotype mapping function and, hence, a different representation.

We applied GOMGE to the three mentioned variants of GE (the original GE, SGE, and WHGE) and incorporated in GOMGE two small modifications motivated by the need of coping with the degeneracy of those representations, i.e., the tendency to map many genotypes to the same phenotype [7,8]. Our work has a twofold aim: (a) extend the benefit in effectiveness delivered by GOMEA to GE, hence further boosting its practical applicability, and (b) shed new lights on the three representations, in particular concerning their proneness to exhibit “good” linkage, i.e., a linkage which can actually be exploited to improve the effectiveness of the EA. The latter point is of particular interest for better understanding both GOMEA (and its linkage learning method) and GE representations: in fact, being based on an indirect representation, the linkage observed in GE is the result of the combination of interactions between genes which occur during the genotype-phenotype mapping and those related to the problem at hand.

We performed an extensive experimental evaluation considering three GE variants with four linkage models applied to four benchmark problems. The results show that GOMGE does improve the effectiveness and efficiency of both SGE and WHGE, whereas it delivers mixed results with GE.

The remainder of the paper is organized as follows. In Sect. 2, we briefly survey previous studies relevant to the present paper. In Sects. 3 and 3.1, we describe the standard search algorithm used in GE and GOMGE, respectively. In Sect. 4, we discuss the results of our experimental evaluation. Finally, in Sect. 5, we summarize the findings and draw the conclusions.

## 2 Related Works

GOMEA [1] has been extended to different EAs or macro-categories of problems: to GP in GOMEA-GP [15], where a novel approach is proposed for identifying and encapsulating relevant building blocks; to real-valued (RV) optimization in RV-GOMEA [16]; to multi-objective (MO) optimization problems in MO-GOMEA [17]. GOMGE is the first application of GOMEA to GE.

There have been several attempts to exploit some form of knowledge of the model underlying the problem for improving the effectiveness of GE. These efforts were usually aimed at discovering useful *building blocks*, i.e., reusable components of the solutions. Position-independent GE ( $\pi$ GE) [18] proposed a novel genotype-phenotype mapping in which the non-terminal symbol to be derived was chosen using the information in the genotype instead of following a left-to-right order. Decoupling non-terminal derivation from the non-terminal choice was expected to favor the emergence of building blocks, but no experimental evidence of the desired effect was provided. A different approach was instead proposed in [19] and, more recently, in [20]. In both cases, the aim is to modify the grammar to discover new problem-specific building blocks and hence improve the search effectiveness; the two cited papers, however, greatly differ in the way they pursue this goal. In [19] a user-defined “universal grammar” related to the class of considered problems (e.g., symbolic regression) is available and part of the genotype is devoted to encode a more specific grammar which describes the actual solution space. In [20], a two phases process is proposed: in a first phase, a probabilistic grammar-based model is learned during an evolution performed using the original user-provided CFG. In a second phase, the new learned model is used to evolve hopefully better solutions. In the present work, differently than the two cited works, we attempt to learn the model (i.e., the linkage) directly at the level of the genotype, instead of at the level of the phenotype (i.e., the grammar).

Another attempt of incorporating the knowledge of the model in GE has been proposed in [21] and further improved in [22]. The authors of the cited paper describe a rather complex theoretical framework in which a model can be obtained from a grammar by means of a deterministic algorithm: the model is a particular graph in which vertexes are partially derived strings of the language and edges are derivation rules. The genotype is not a bit-string, but instead a

sequence of derivation rules which, essentially, allows to move along the graph. Accordingly, the proposed EA uses specific genetic operators, making it rather different than the original GE. Finally, it is worth to mention that in several studies of model-based GP, the proposed model itself consisted in (probabilistic) grammars (e.g., [23]): we refer the reader to [24] for a broader overview of these approaches.

### 3 Grammatical Evolution

GE has been widely studied and several variants for the various EA components have been proposed. Here, we present the most widely used search algorithm and representation (for which we consider also the recent variants SGE and WHGE), because they are relevant to this study.

Algorithm 1 shows the search algorithm of GE. First, the initial population  $I$  consisting of  $n_{\text{pop}}$  individuals is built. In this work, we consider an initialization procedure in which genotypes of a given length  $l_g$  are generated at random, but more complex strategies may be employed. Then, the following steps are repeated until the termination criterion is met: (1) A new population  $I'$  (with  $|I'| = |I| = n_{\text{pop}}$ ) is built from  $I$  by applying the genetic operators (mutation and crossover chosen according to the probabilities  $p_{\text{mut}}, p_{\text{cross}}$ ) to parents selected from the population  $I$  using a predefined parent selection criterion (SELECTPARENT() in Algorithm 1). (2) The population  $I$  is updated by including the new population  $I'$  and then by removing the  $n_{\text{pop}}$  exceeding individuals using a predefined removal selection criterion (SELECTREMOVAL() in Algorithm 1).

Concerning the termination criterion, the most common option is to repeat the two steps above for a predefined number of times, usually called the number of generations. In this work, however, we chose a different criterion for enabling a fairer comparison with GOMGE which, differently than the  $m+n$  generational model of Algorithm 1, may perform a large number of fitness evaluations in each iteration of the main loop (see Sect. 3.1). We hence adopted for both search algorithms the following stopping criterion. The steps are iterated until at least one of the two following conditions is met: (a) the elapsed time  $T$  after the beginning of the evolution exceeds a predefined time limit  $T_{\text{max}}$  or (b) the population  $I$  includes an individual with perfect fitness  $f$  (the notion of perfect fitness being dependent, in general, on the problem).

The search algorithm defined in Algorithm 1 is agnostic to the specific selection criteria SELECTPARENT() and SELECTREMOVAL(): tournament selection and worst fitness selection (i.e., truncation selection) are, respectively, common choices. The genotype-phenotype mapping function MAP() is the component in which the GE variants mostly differ and essentially defines the individual representation. In this work, we consider the original representation and two recent variants: Structured GE (SGE) [13] and Weighted Hierarchical GE (WHGE) [14]: it is worth to note that, in both cases, the proposal of the representation variant was aimed at improving the poor properties of the original GE representation, in particular degeneracy and locality. Degeneracy concerns the degree to which



different genotypes are mapped to the same phenotype. Locality describes how well genotypic neighbors correspond to phenotypic neighbors. It has been shown that these properties are related to higher level properties of the EA, as, e.g., diversity [7] and evolvability [25]. It is foreseeable that degeneracy and locality may impact also on the learnability of the linkage and may interplay with the GOM operator.

---

**Algorithm 1. Standard GE.**


---

```

function EVOLVE()
   $I \leftarrow \text{INITPOPULATION}(n_{\text{pop}})$ 
  while  $\neg \text{TERMINATIONCRITERIONMET}()$  do
     $I' \leftarrow \emptyset$ 
    while  $|I'| < n_{\text{pop}}$  do
       $o \leftarrow \text{GETOPERATOR}()$ 
       $G_p \leftarrow \emptyset$ 
      while  $|G_p| \leq \text{ARITY}(o)$  do
         $(g_p, p_p, f_p) \leftarrow \text{SELECTPARENT}(I)$ 
         $G_p \leftarrow G_p \cup \{g_p\}$ 
      end while
       $G_c \leftarrow \text{APPLY}(o, G_p)$ 
      for all  $g_c \in G_c$  do
         $p_c \leftarrow \text{MAP}(g_c)$ 
         $f_c \leftarrow \text{FITNESS}(p_c)$ 
         $I' \leftarrow I' \cup \{(g_c, p_c, f_c)\}$ 
      end for
    end while
     $I \leftarrow I \cup I'$ 
    while  $|I| > n_{\text{pop}}$  do
       $I \leftarrow I \setminus \{\text{SELECTREMOVAL}(I)\}$ 
    end while
  end while
  return  $\text{BEST}(I)$ 
end function

```

---



---

**Algorithm 2. GOMGE.**


---

```

function EVOLVE()
   $I \leftarrow \text{INITPOPULATION}(n_{\text{pop}})$ 
   $i^* = \text{BEST}(I)$ 
  while  $\neg \text{TERMINATIONCRITERIONMET}()$  do
     $\mathcal{F} \leftarrow \text{LEARNLINKAGEMODEL}(I)$ 
     $I' \leftarrow \emptyset$ 
    for all  $(g, p, f) \in I$  do
       $(g_0, p_0, f_0) \leftarrow (g, p, f)$ 
      for all  $F \in \text{RNDPERM}(\mathcal{F})$  do
         $g_c \leftarrow g$ 
         $(g_d, p_d, f_d) \leftarrow \text{RANDOMDONOR}(I)$ 
         $g_c[F] \leftarrow g_d[F]$ 
         $p_c \leftarrow \text{MAP}(g_c)$ 
         $f_c \leftarrow \text{FITNESS}(p_c)$ 
        if  $f_c < f$  then
           $(g, p, f) \leftarrow (g_c, p_c, f_c)$ 
        end if
      end for
    end for
    while  $p = p_0$  do
       $g \leftarrow \text{APPLY}(\text{MUTATION}, \{g\})$ 
       $p \leftarrow \text{MAP}(g)$ 
       $f \leftarrow \text{FITNESS}(p)$ 
    end while
     $I' \leftarrow I' \cup \{(g, p, f)\}$ 
  end for
   $I \leftarrow I'$ 
   $i^* = \text{BEST}(I \cup \{i^*\})$ 
end while
return  $i^*$ 
end function

```

---

Due to space constraints, we do not describe in details the representation of GE, SGE, and WHGE: we provide a coarse overview of the underlying principles and refer the reader to the respective papers for further details. Being forms of grammar-based genetic programming, in GE, SGE, and WHGE the phenotype is a string of the language language  $\mathcal{L}(\mathcal{G})$  defined by a user-provided CFG  $\mathcal{G}$ , which is an implicit parameter of the mapping function  $\text{MAP}()$ . In the original GE [2], the genotype  $g$  is a bit-string. Groups of 8 consecutive bits in the genotype are called codons: each *codon* encodes an integer value and is consumed for deriving the leftmost non-terminal. SGE has been introduced in [13] by Lourenço et al. In SGE, the genotype  $g$  consists of a number of fixed-size lists (*genes*) of integers: each list corresponds to a non-terminal symbol of the CFG and each integer in the list (*codon*) determines a single derivation for that non-terminal. Finally, the most recent WHGE [14] is designed to consume the genotype hierarchically with the aim of reducing the degeneracy and increasing the locality. In WHGE, the genotype  $g$  is a bit-string, as in the original GE.

### 3.1 GOMGE: Gene-Pool Optimal Mixing EA for GE

Our GOMGE proposal consists on two localized modifications to the adaptation of GOMEA to GP [15], described below and motivated by explorative experiments and recent findings about (lack of) diversity in GE [7,26]. GOMGE is described in Algorithm 2. After the initialization of the population, the main loop is repeated until a termination criterion is met and consists in two steps: (i) learning the linkage from the current population and (ii) applying the Gene-pool Optimal Mixing (GOM) variation operator to each individual in the population. The linkage is expressed as a Family of Subsets (FOS)  $\mathcal{F} = \{F_1, F_2, \dots\}$  which is a set of sets of zero-based genotype indexes (*loci*): i.e.,  $F_i \subseteq \{0, \dots, l_g - 1\}$ , where  $l_g$  is the evolution-wise immutable size of the genotype. We experimented with 4 different way of obtaining the FOS described at the end of this section.

Applying the GOM operator to an individual  $(g, p, f)$  consists in repeating the following steps for each set  $F$  in a random permutation of  $\mathcal{F}$ : (i) a donor  $(g_d, p_d, f_d)$  is randomly chosen in the population and (ii) the portions of the genotype  $g$  defined by  $F$  are replaced with the corresponding portions coming from  $g_d$ ; (iii) the fitness  $\text{FITNESS}(\text{MAP}(g))$  of the new individual is computed and, (iv) if there is a strict improvement, the modification on the individual  $g$  is kept, otherwise, it is rolled back.

After preliminary experiments, we observed that this version of the GOM often resulted in no modifications being applied to the individual, since no fitness improvements were obtained. We think this finding is motivated by the degeneracy of the indirect representation of GE and its variants: the likelihood is non-negligible of obtaining the same individual after an iteration of GOM operator; as a consequence, the fitness does not improve and the evolution might stagnate. We hence modified the GOM operator by employing a *forced mutation* (performed with a mutation operator suitable to the specific representation being used) in case the phenotype did not change after the processing of all the sets in  $\mathcal{F}$ . The idea is borrowed from [15], where a phase called “forced improvement” eventually results in an individual with a better fitness than the input one, yet possibly equal to another individual in the population. Here, we instead simply apply a standard mutation, because otherwise the tendency of GE and variants to drastically reduce the diversity in the population during the evolution could have been further stimulated.

Since the forced mutation might apply to the best individual in the population (hence negatively affecting the results of the search obtained so far), we introduced a simple mechanism for keeping track of the best individual  $i^*$ , which is updated at each iteration of the main loop.

In GOMGE, as in GOMEA, the linkage is expressed using a FOS, which can either be learned from the population or being predefined. We considered 4 variants, two belonging to the former category (Linkage Tree and Random Tree) and two to the latter category (Univariate and Natural).

The Univariate FOS (later denoted by U) is the simplest FOS and assumes that there is no linkage between portions of the genotype. This FOS contains one singleton set for each possible locus:  $\mathcal{F}_U = \{\{0\}, \{1\}, \dots, \{l_g - 1\}\}$ .

The Natural FOS (later denoted by N) is a statically built FOS which tries to capture the representation-dependent linkage. We defined it only for GE, where it captures the fact that derivations are chosen using groups of 8 consecutive bits (i.e.,  $\mathcal{F}_{N,GE} = \{\{0, 1, \dots, 7\}, \{8, 9, \dots, 15\}, \dots\}$ ), and for SGE, where it captures the fact that integers are organized in lists, the size of each list being dependent on the grammar (i.e.,  $\mathcal{F}_{N,SGE} = \{\{0, \dots, |g_{s_0}| - 1\}, \{|g_{s_0}|, \dots, |g_{s_0}| + |g_{s_1}| - 1\}, \dots\}$ ).

The learnable variants are Linkage Tree and Random Tree, later denoted by LT and RT, respectively. LT was already considered in the seminal GOMEA paper and was later shown to be very beneficial to search effectiveness, in particular in black-box optimization problems [27]. LT models complex linkage structures using a hierarchy, i.e., a tree where nodes are sets of loci and a node is the set union of its children. The LT FOS  $\mathcal{F}_{LT}$  is built from the population as follows. Initially, a set of sets of loci  $\mathcal{F}_0$  is set to  $\mathcal{F}_U$  and  $\mathcal{F}_{LT} = \mathcal{F}_0$ . Then, the following steps are repeated until  $|\mathcal{F}_0| \geq 1$ : (i) the pair  $F, F' \in \mathcal{F}_0 \times \mathcal{F}_0$  of loci sets, with  $F \neq F'$ , is determined which exhibits the greatest mutual information; then (ii)  $F, F'$  are removed from  $\mathcal{F}_0$  and  $F \cup F'$  is added to  $\mathcal{F}_0$  and  $\mathcal{F}_{LT}$ . This procedure may be implemented efficiently using the algorithm described in [28], where the mutual information between sets of loci is estimated, rather than computed using the genotype values at  $F, F'$  loci observed in the population (we refer the reader to the cited paper for further details).

Finally, RT resembles LT since it also models the linkage as a hierarchy. Differently than LT, however, a random value instead of the mutual information is used at step 3.1 above when building  $\mathcal{F}_{RT}$ . The rationale is to allow, as for LT, the simultaneous modification at different loci of the genotype.

## 4 Experimental Evaluation

For assessing experimentally the effectiveness of GOMGE w.r.t. the standard GE search algorithm, we considered 4 benchmark problems: Parity (with  $n \in \{5, \dots, 9\}$ ), Nguyen7 [29], KLandscapes [30] (with  $k \in \{3, \dots, 7\}$ ), and Text [7]. These problems represent different domains, including boolean functions (Parity), symbolic regression (Nguyen7), and synthetic problems (KLandscapes and Text). Two of them have a tunable hardness (Parity and KLandscapes) and two are recommended as GP benchmarks in [31] (Nguyen7 and KLandscapes); one (Text) has been designed purposely for grammar-based GP and is based on a grammar with more derivation rules and more symbols than the other considered problems.

We performed the experimental evaluation using a prototype Java implementation of both standard GE and GOMGE. The implementation and the grammars for the benchmark problems are publicly available<sup>1</sup>. The prototype includes a two caches for the fitness function `FITNESS()` and the genotype-phenotype mapping function `MAP()`; both use a size-based eviction policy with a size limit of 200 000 entries.

<sup>1</sup> <https://github.com/ericmedvet/evolved-ge>.

We performed 30 runs for each of the five variants (standard GE, to be considered as the *baseline* and later denoted by Base., and GOMGE coupled with the 4 FOSs, U, N, RT, and LT) on each of the four problems. We executed each run on one node of the CINECA HPC cluster (Marconi-A1), the node having 2 Intel Xeon E5-2694 v4 CPUs (2.3mGHz) with 18 cores each and 128 GB of RAM.

We set the main evolutionary parameters as follows: genotype size  $l_g = 512$  for GE,  $l_g = 128$  for WHGE, or determined by  $d = 6$  (see [13]) for SGE; population size  $n_{\text{pop}} = 500$ ; two-points same crossover for GE, WHGE or SGE crossover for SGE with rate 0.8; bit-flip mutation with  $p_{\text{mut}} = 0.01$  for GE, WHGE or SGE mutation with  $p_{\text{mut}} = 0.01$  for SGE with rate 0.2; tournament selection of size 3; and max elapsed time  $T_{\text{max}}=60$  s.

Table 1 shows the mean and the standard deviation (across the 30 runs) of the final best fitness for each problem and variant. The table also shows, graphically and for each GOMGE variant and problem, the statistical significance ( $p$ -value with the Mann-Whitney U-test) of the null hypothesis that the final best fitness values have equal median of those obtained with the baseline.

**Table 1.** Mean and standard deviation of the final best fitness. The best mean for each problem is highlighted. The statistical significance (see text) is shown graphically: ‡ means  $p < 0.01$ , † means  $p < 0.05$ , and \* means  $p < 0.1$  (no markers for greater  $p$ -values).

	Var.	Parity-7	Nguyen7	KLand.-5	Text
GE	Base.	$0.5 \pm 0.02$	$0.39 \pm 0.25$	$0.61 \pm 0.09$	$4.9 \pm 1.2$
	U	$0.5 \pm 0.01$	$0.49 \pm 0.19^{\ddagger}$	$0.63 \pm 0.06^{\ddagger}$	$3.5 \pm 0.7^{\ddagger}$
	N	$0.5 \pm 0$	$0.4 \pm 0.2$	$0.61 \pm 0.09$	<b><math>3.1 \pm 0.8^{\ddagger}</math></b>
	RT	$0.49 \pm 0.02$	$0.68 \pm 0.6^{\ddagger}$	$0.68 \pm 0.04^{\ddagger}$	$4.5 \pm 0.6^{\ddagger}$
	LT	$0.49 \pm 0.03$	$0.68 \pm 0.16^{\ddagger}$	$0.68 \pm 0.04^{\ddagger}$	$4.6 \pm 0.5^{\ddagger}$
WHGE	Base.	$0.17 \pm 0.13$	$0.52 \pm 0.19$	$0.4 \pm 0.08$	$5.7 \pm 0.8$
	U	$0.16 \pm 0.07^*$	$0.31 \pm 0.15^{\ddagger}$	$0.6 \pm 0.05^{\ddagger}$	$4.9 \pm 0.5^{\ddagger}$
	RT	<b><math>0 \pm 0^{\ddagger}</math></b>	<b><math>0.18 \pm 0.11^{\ddagger}</math></b>	$0.29 \pm 0.04^{\ddagger}$	$4 \pm 0^{\ddagger}$
	LT	<b><math>0 \pm 0^{\ddagger}</math></b>	$0.21 \pm 0.12^{\ddagger}$	<b><math>0.25 \pm 0.07^{\ddagger}</math></b>	$4 \pm 0^{\ddagger}$
SGE	Base.	$0.08 \pm 0.12$	$0.7 \pm 0.12$	$0.54 \pm 0.14$	$6.3 \pm 0.5$
	U	<b><math>0 \pm 0^{\ddagger}</math></b>	$0.35 \pm 0.23^{\ddagger}$	$0.34 \pm 0^{\ddagger}$	$5.4 \pm 0.5^{\ddagger}$
	N	<b><math>0 \pm 0^{\ddagger}</math></b>	$0.29 \pm 0.24^{\ddagger}$	$0.34 \pm 0^{\ddagger}$	$5.1 \pm 0.3^{\ddagger}$
	RT	<b><math>0 \pm 0^{\ddagger}</math></b>	$0.65 \pm 1.14^{\ddagger}$	$0.34 \pm 0^{\ddagger}$	$5 \pm 0.2^{\ddagger}$
	LT	<b><math>0 \pm 0^{\ddagger}</math></b>	$0.54 \pm 0.21^{\ddagger}$	$0.34 \pm 0^{\ddagger}$	$5 \pm 0.2^{\ddagger}$

The foremost finding is that GOMGE outperforms the baseline with WHGE and SGE in almost all cases (i.e., FOS and problem), with the single exception of U with WHGE on the KLandscapes-5 problem, for which the baseline performs

better (0.4 vs. 0.6). The difference is always significant. GOMGE improvement becomes evident on the Parity-7 problem, for which both WHGE and SGE obtain the perfect fitness in all the runs only with GOMGE. Concerning the FOS, it can be seen that the largest improvement is delivered by RT and LT, for WHGE, and by N, for SGE (see also the next tables): the facts that LT and RT lead to the same good performance with WHGE and that N with SGE resembles the SGE crossover operator (see [13]) suggest that the improvement is related to the reiterated application of the GOM operator, rather than to the possibility of learning the linkage.

Differently, coupling GOMGE with GE representation leads to mixed results: no significant difference are visible on one problem (Parity-7), a decrease in the fitness is visible on two problems (Nguyen7 and KLandscapes-5), and an improvement is visible for the last problem (Text). Overall, N is the best FOS for GE.

For better understanding the results in terms of final best fitness, we analyzed also three other relevant metrics: the elapsed time  $T$ , the number of actual fitness evaluations  $N$  (corresponding to the fitness cache miss count), and the final phenotypical diversity  $D$ . We measured the latter as the ratio between the number of different phenotypes in the population and the population size.

**Table 2.** Elapsed time  $T$  (in s), number  $N$  of actual fitness evaluations (in thousands), and final phenotype diversity  $D$  (in percentage).

	Var.	Elapsed time $T$ [s]				Act. fitness ev. $N$ [ $\times 10^3$ ]				Pheno. div. $D$ [%]			
		Par.-7	Ng.7	KL.-5	Text	Par.-7	Ng.7	KL.-5	Text	Par.-7	Ng.7	KL.-5	Text
GE	Base.	85	66	65	59	0.2	6.8	5	5	6	4	7	2
	U	114	57	59	59	0.4	12.8	19.2	192.8	23	35	68	93
	N	123	59	55	60	0.2	37.8	32.1	133.7	28	54	69	97
	RT	145	65	65	68	0.7	7.9	14.7	67.9	27	53	69	96
	LT	149	79	72	69	0.6	8	14.6	53.1	28	54	69	96
WHGE	Base.	68	64	64	61	10.7	9.4	10.5	6.5	11	4	31	12
	U	71	64	71	62	585.2	299.8	160.2	197.6	92	89	88	92
	RT	14	86	61	77	353.7	548.7	352.7	591.9	79	56	89	48
	LT	15	85	61	83	328.7	360.7	321.4	457.1	71	45	54	51
SGE	Base.	43	61	62	62	1.2	3.3	2.6	1.2	5	7	5	4
	U	20	60	64	60	39.6	50.1	52.1	45.1	98	71	62	91
	N	1	63	63	59	38.8	89.9	53.3	62	98	71	38	82
	RT	2	57	76	58	47.3	81.2	54.8	46.3	92	52	27	50
	LT	4	53	68	59	57.9	21.8	52	33.4	64	27	27	32

Table 2 shows the mean (across the 30 runs) of the three metrics  $T$ ,  $N$ ,  $D$  for each problem and variant. Two main observations may be made. First, the number of actual fitness evaluation increases with GOMGE, the increment being remarkable for WHGE and SGE (up to  $20\times$ ). This figure is also reflected in the elapsed time  $T$ , when a perfect fitness value is not found. We recall that one of

the two termination criteria is the elapsed time, with a time limit of  $T_{\max} = 60\text{s}$ : however, since the condition is evaluated once per main loop, the limit may be exceeded, in particular for GOMGE. It can also be noted that GE, in all the 5 variants, performs a very low number (hundreds, on average) of actual fitness evaluations for the Parity-7 problem: this is mainly due to high degeneracy, which has already been shown to hamper this representation in particular in problems where the phenotype should be large [7], and, to a lesser extent, to invalidity, i.e., the tendency of generating a null phenotype after exceeding the maximum number of wrappings [2].

Second, Table 2 shows that the final phenotypical diversity is in general much larger with GOMGE than with the baseline, in all cases: values are around 10% with the latter and often exceed 90% with GOMGE. This finding can be explained by the fact that GOMGE does not select best individuals for applying the GOM operator, but rather replaces a parent with a child only upon a fitness improvement, a mechanism resembling the established diversity promotion scheme known as deterministic crowding [32]. Together, the two observations suggest that GOMGE is at the same time more effective and more efficient than the baseline in exploring the search space: indeed, it can be noted from Tables 1 and 2 that the largest fitness improvements are obtained in sync with improvements in the metrics  $N$  and  $D$ . Significantly, the only problem in which GOMGE outperforms the baseline with GE is the one (Text) in which the increment of  $N$  and  $D$  is the greatest.

## 5 Concluding Remarks

We presented GOMGE, an application of the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) to Grammatical Evolution (GE), a form of general-purpose Genetic Programming which is widely used by practitioners because it easily applies to any problem whose solutions may be described by a context-free grammar. We incorporated in GOMGE two specific improvements for coping with the degeneracy (i.e., the tendency to map many genotypes to the same phenotype) of GE indirect representations. We applied GOMGE to three variants of GE (original GE, SGE, and WHGE), essentially differing in the individual representation, a key component of any EA which has been shown to impact on many higher-level EA properties (e.g., evolvability), and eventually on its effectiveness.

We performed an extensive experimental evaluation of 4 GOMGE variants, differing in the way of obtaining a linkage model, on 4 benchmark problems. The results show that GOMGE is significantly beneficial to both effectiveness and efficiency of the search with SGE and WHGE, whereas it delivers mixed results with the original GE. At a deeper analysis, the experimental results suggest that the drastic increase in the phenotypical diversity and in the number of actual fitness evaluation are key factors for explaining the performance gap between GOMGE and standard GE.

We think that our proposal further boosts the applicability of GE to practical problems and sheds new light on the possibility of GE representations to exhibit “good” and learnable linkage.

## References

1. Thierens, D., Bosman, P.A.: Optimal mixing evolutionary algorithms. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011, pp. 617–624. ACM, New York (2011)
2. 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. 83–96. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055930>
3. Bartoli, A., De Lorenzo, A., Medvet, E., Tarlao, F.: Syntactical similarity learning by means of grammatical evolution. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 260–269. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_24](https://doi.org/10.1007/978-3-319-45823-6_24)
4. Medvet, E., Bartoli, A., Talamini, J.: Road traffic rules synthesis using grammatical evolution. In: Squillero, G., Sim, K. (eds.) EvoApplications 2017. LNCS, vol. 10200, pp. 173–188. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55792-2\\_12](https://doi.org/10.1007/978-3-319-55792-2_12)
5. Castejón, F., Carmona, E.J.: Automatic design of analog electronic circuits using grammatical evolution. *Appl. Soft Comput.* **62**, 1003–1018 (2018)
6. Miranda, P.B., Prudêncio, R.B.: Generation of particle swarm optimization algorithms: an experimental study using grammar-guided genetic programming. *Appl. Soft Comput.* **60**, 281–296 (2017)
7. Medvet, E., Bartoli, A., Talamini, J.: Road traffic rules synthesis using grammatical evolution. In: Squillero, G., Sim, K. (eds.) EvoApplications 2017. LNCS, vol. 10200, pp. 173–188. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55792-2\\_12](https://doi.org/10.1007/978-3-319-55792-2_12)
8. Thorhauer, A.: On the non-uniform redundancy in grammatical evolution. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 292–302. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_27](https://doi.org/10.1007/978-3-319-45823-6_27)
9. Thorhauer, A., Rothlauf, F.: On the locality of standard search operators in grammatical evolution. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) PPSN 2014. LNCS, vol. 8672, pp. 465–475. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10762-2\\_46](https://doi.org/10.1007/978-3-319-10762-2_46)
10. Medvet, E., Bartoli, A.: On the automatic design of a representation for grammar-based genetic programming. In: Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S., García-Sánchez, P. (eds.) EuroGP 2018. LNCS, vol. 10781, pp. 101–117. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77553-1\\_7](https://doi.org/10.1007/978-3-319-77553-1_7)
11. Whigham, P.A., Dick, G., Maclaurin, J.: On the mapping of genotype to phenotype in evolutionary algorithms. *Genet. Program. Evol. Mach.* **18**, 1–9 (2017)
12. Spector, L.: Introduction to the peer commentary special section on “on the mapping of genotype to phenotype in evolutionary algorithms” by peter a. whigham, grant dick, and james maclaurin. *Genet. Program. Evol. Mach.* **18**(3), 351–352 (2017)

13. Lourenço, N., Pereira, F.B., Costa, E.: SGE: a structured representation for grammatical evolution. In: Bonnevay, S., Legrand, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) EA 2015. LNCS, vol. 9554, pp. 136–148. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-31471-6\\_11](https://doi.org/10.1007/978-3-319-31471-6_11)
14. Medvet, E.: Hierarchical grammatical evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2017, pp. 249–250. ACM, New York (2017)
15. Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.N.: Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1041–1048. ACM (2017)
16. Bouter, A., Alderliesten, T., Witteveen, C., Bosman, P.A.: Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 705–712. ACM (2017)
17. Luong, N.H., La Poutré, H., Bosman, P.A.: Multi-objective gene-pool optimal mixing evolutionary algorithms. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 357–364. ACM (2014)
18. O’Neill, M., Brabazon, A., Nicolau, M., Garraghy, S.M., Keenan, P.:  $\pi$ grammatical evolution. In: Deb, K. (ed.) GECCO 2004. LNCS, vol. 3103, pp. 617–629. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24855-2\\_70](https://doi.org/10.1007/978-3-540-24855-2_70)
19. O’Neill, M., Ryan, C.: Grammatical evolution by grammatical evolution: the evolution of grammar and genetic code. In: Keijzer, M., O’Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 138–149. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24650-3\\_13](https://doi.org/10.1007/978-3-540-24650-3_13)
20. Wong, P.-K., Wong, M.-L., Leung, K.-S.: Hierarchical knowledge in self-improving grammar-based genetic programming. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 270–280. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_25](https://doi.org/10.1007/978-3-319-45823-6_25)
21. He, P., Johnson, C.G., Wang, H.: Modeling grammatical evolution by automaton. *Sci. China Inf. Sci.* **54**(12), 2544–2553 (2011)
22. He, P., Deng, Z., Gao, C., Chang, L., Hu, A.: Analyzing grammatical evolution and  $\pi$ Grammatical evolution with grammar model. In: Balas, V.E., Jain, L.C., Zhao, X. (eds.) Information Technology and Intelligent Transportation Systems. AISC, vol. 455, pp. 483–489. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-38771-0\\_47](https://doi.org/10.1007/978-3-319-38771-0_47)
23. Shan, Y., McKay, R.I., Baxter, R., Abbass, H., Essam, D., Nguyen, H.: Grammar model-based program evolution. In: Congress on Evolutionary Computation, CEC2004, Volume 1, pp. 478–485. IEEE (2004)
24. Shan, Y., McKay, R., Essam, D., Abbass, H.: A survey of probabilistic model building genetic programming. In: Pelikan, M., Sastry, K., CantÚPaz, E. (eds.) Scalable Optimization via Probabilistic Modeling, pp. 121–160. Springer, Heidelberg (2006). [https://doi.org/10.1007/978-3-540-34954-9\\_6](https://doi.org/10.1007/978-3-540-34954-9_6)
25. Medvet, E., Daolio, F., Tagliapietra, D.: Evolvability in grammatical evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 977–984. ACM (2017)
26. Medvet, E., Bartoli, A., Squillero, G.: An effective diversity promotion mechanism in grammatical evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 247–248. ACM (2017)



27. Thierens, D., Bosman, P.A.N.: Hierarchical problem solving with the linkage tree genetic algorithm. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, pp. 877–884. ACM (2013)
28. Gronau, I., Moran, S.: Optimal implementations of UPGMA and other common clustering algorithms. *Inf. Process. Lett.* **104**(6), 205–210 (2007)
29. Uy, N.Q., Hoai, N.X., O’Neill, M., McKay, R.I., Galván-López, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genet. Program. Evol. Mach.* **12**(2), 91–119 (2011)
30. Vanneschi, L., Castelli, M., Manzoni, L.: The k landscapes: a tunably difficult benchmark for genetic programming. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 1467–1474. ACM (2011)
31. White, D.R., Mcdermott, J., Castelli, M., Manzoni, L., Goldman, B.W., Kronberger, G., Jaśkowski, W., O’Reilly, U.M., Luke, S.: Better gp benchmarks: community survey results and proposals. *Genet. Program. Evol. Mach.* **14**(1), 3–29 (2013)
32. Squillero, G., Tonda, A.: Divergence of character and premature convergence: a survey of methodologies for promoting diversity in evolutionary optimization. *Inf. Sci.* **329**, 782–799 (2016)



# Self-adaptive Crossover in Genetic Programming: The Case of the Tartarus Problem

Thomas D. Griffiths<sup>(✉)</sup> and Anikó Ekárt

Aston Lab for Intelligent Collectives Engineering (ALICE), Aston University,  
Aston Triangle, Birmingham B4 7ET, UK  
{grifftd1,a.ekart}@aston.ac.uk

**Abstract.** The runtime performance of many evolutionary algorithms depends heavily on their parameter values, many of which are problem specific. Previous work has shown that the modification of parameter values at runtime can lead to significant improvements in performance. In this paper we discuss both the ‘*when*’ and ‘*how*’ aspects of implementing self-adaptation in a Genetic Programming system, focusing on the crossover operator. We perform experiments on Tartarus Problem instances and find that the runtime modification of crossover parameters at the individual level, rather than population level, generate solutions with superior performance, compared to traditional crossover methods.

**Keywords:** Self-adaption · Crossover · Tartarus problem

## 1 Introduction

In the field of Evolutionary Algorithms and specifically Genetic Programming, it is widely accepted that the on-the-fly modification and adaptation of parameters values at runtime can lead to improvements in performance [1]. This process of modifying parameter values can be conceptualised into two distinct processes, the first: ‘*when*’ to modify and the second: ‘*how*’ to modify.

A common approach for deciding ‘*when*’ to trigger the parameter modifications, whether they be deterministic [2] or probabilistic [3], is decided by use of a pre-determined schedule or fixed time interval; we refer to these as *episodic modifications*. The primary benefit of episodic methods is that they allow for a regular and predictable sequence of parameter modifications to be performed over time without the need for any further interaction.

However, the rigid nature of this approach presents several drawbacks when utilised on dynamic or multi-dimensional optimisation problems, such as the Tartarus Problem (TP). An alternative to *episodic modification* is to create a mechanism which provides a continual opportunity to modify parameter values at any time; we refer to this as *continuous modification*. We therefore propose the introduction of a self-adaptive crossover bias method, allowing for the continual modification of individual crossover parameters at runtime.

The process of deciding ‘*how*’ the parameter value is to be modified is often more complex, this can be divided into two smaller, sequential sub-tasks:

- Deciding the mechanism by which the parameter values are modified,
- Calculating the magnitude of the parameter value modifications.

This division between the mechanism and the magnitude allows for the methods by which the modifications are made and the impact of those modifications, to be tuned and controlled separately at runtime. There exist several different approaches to deciding ‘*how*’ the parameter values should be modified that are utilised in Genetic Programming, these can be classified as either *deterministic*, *adaptive* or *self-adaptive*. An outline and comparative taxonomy of these approaches is presented in Sect. 2.

It is hypothesised that allowing the Genetic Programming system to trigger the parameter modifications ‘*as and when they are required*’ will reduce the number of ineffective adaptations being executed, increasing efficiency and allowing for convergence to an optimal solution. The proposed self-adaptive crossover bias will create a more *continuous* parameter value modification process, which is more flexible compared to the rigid, traditional *episodic* approach, leading to an increase in solution performance.

In this paper we discuss the differences between adaptive and self-adaptive parameter modification and implement a self-adaptive crossover bias method in genetic programming. The paper is organised as follows: Sect. 2 discusses and defines the differences between adaptation and self-adaptation, presenting a taxonomy of the two parameter modification approaches. Section 3 describes the Tartarus Problem and the experimental setup. Section 4 presents the proposed self-adaptive crossover operator and compares the performance with that of individuals utilising a standard crossover operator. Finally, Sect. 5 addresses conclusions and future research aspirations on the topic.

## 2 Parameter Modification Approaches

The parameter modification approaches utilised in genetic programming can be generally classified into one of three categories [1], being *deterministic*, *adaptive* or *self-adaptive*<sup>1</sup> in nature. The characteristics, flexibility and complexity varies widely between the three categories of approach.

**Deterministic Parameter Modification** – The parameter value is modified on a *global level* according to a fixed, pre-determined rule. The modification receives no feedback from, and is not influenced by, the current status of the search [4, 5].

**Adaptive Parameter Modification** – The parameter value is modified on a *global level* according to a mechanism, which receives input from, and is at least partly influenced by, the status of the search [6].

---

<sup>1</sup> The descriptive terms ‘Adaptive’ and ‘Self-Adaptive’ are used in the broad general context of Evolutionary Computation. These terms have distinct meanings in fields such as Artificial Life; based on strict Ecological and Psychological definitions.

**Self-adaptive Parameter Modification** – The parameter value is modified on an *individual level*, where the parameters are encoded into the genome of an individual in some form. The parameters undergo the same processes of mutation and recombination as the individuals themselves. The modification of these parameter values is coupled with the status of the search [7].

In Adaptive Parameter Modification (APM) the mechanism by which the parameter values are modified is defined in advance, leading to explicit exogenous parameter modification. The performance of APM is only as good as the information that it receives from the environment, care must be taken to ensure that the information received is applicable to the selected parameters.

Conversely, in the Self-Adaptive Parameter Modification (SAPM) the way in which the parameter values are modified is entirely implicit. In this approach the mutation and recombination processes of the evolutionary cycle itself are used and exploited. The parameter values are embedded in the representation [8], leading to an implicit endogenous parameter modification. The performance of SAPM is closely linked to the choice of evolutionary operators, therefore effective operator choice is essential. Table 1 outlines a taxonomy of approaches comparing the three methods of *deterministic*, *adaptive* and *self-adaptive* parameter modification.

**Table 1.** Taxonomy of parameter modification approaches. (× indicates a relationship.)

		Deterministic	APM	SAPM
Affected by	Explicitly-defined mechanisms	×	×	
	State of the search		×	×
	Operator selection			×
Modifies	Population level parameters	×	×	
	Individual level parameters			×

The taxonomy outlined in Table 1 allows for the comparison of the different parameter modification approaches to be made. Each approach is *affected by* a selection of factors, both internal and external, which influence the overall effectiveness and performance. The *self-adaptive* approach leads to *modifications* to be made at the *individual* level, in contrast the *adaptive* and *deterministic* approaches both lead to *modifications* to be made at the *global* level.

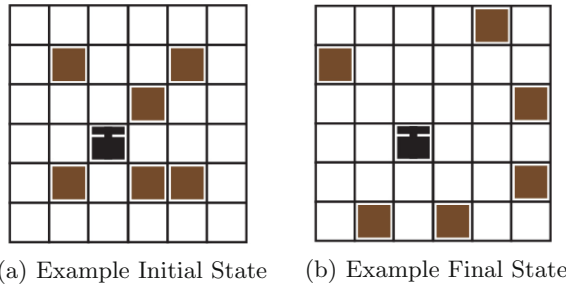
### 3 The Tartarus Problem

The Tartarus problem is a grid-based optimisation problem [9], which we introduced as a genetic programming benchmark [10]. The problem was chosen due to the fact that it satisfies many of the desirable benchmark characteristics outlined

by White et al. [11]. One of the most important characteristics of an effective benchmark problem is *tunable difficulty* [12], the ability to create several problem instances with a tunable and predictable level of difficulty.

A Tartarus instance comprises of an enclosed, non-toroidal  $n \times n$  grid, a set number of movable blocks B and a controllable agent, as shown in Fig. 1(a). Unlike in other grid based problems, such as the Lawnmower problem [13], the agent is initially unaware of its location and orientation within the environment. The agent receives input from eight sensors, allowing it to detect both blocks and the environment boundary in the surrounding eight grid-squares. The goal is to locate and move the blocks to the environment boundary, as shown in Fig. 1(b). At the end of a run, the environment is analysed and the agent is awarded a score, the fitness score, based on its progress in achieving the goal. The agent is able to change its state by executing a finite number of actions  $m$ , chosen from the following three actions:

- (1) *turn left*,    (2) *turn right*,    (3) *move forwards one square*.

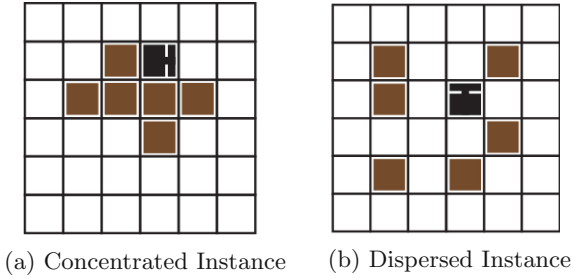


**Fig. 1.** Example states for the canonical  $6 \times 6$  Tartarus instance.

### 3.1 Improved State Evaluation

We previously suggested that the original method of evaluating the state of Tartarus instances was insufficient to capture the progress of the agent [10]. The original method of state evaluation only rewarded individuals who had pushed blocks all the way to the edges of the grid. This binary success or fail approach works well for many benchmark problems where the absolute score achieved by a candidate solution is the only desired success measure. However, for GP, rewarding part-way solutions is essential during evolution, so that better solutions can evolve.

For example, the concentrated instance in Fig. 2(a) is very different from the dispersed instance in Fig. 2(b). However, under the original evaluation method [9] both of these states would have the same fitness score of zero. The blocks in the dispersed instance are visibly closer to the edge of the grid when compared to the blocks in the concentrated instance. Specifically, it would take a total of 32 movement actions to move the blocks to the edge of the grid in the concentrated instance (a), but only 27 actions to move the blocks in the dispersed instance (b).



**Fig. 2.** Comparison of concentrated and dispersed instances

We proposed an improved method for evaluating the state of a Tartarus instance that utilises a more granular approach, rewarding blocks which have moved part-way as well as blocks which have been moved completely to the edge. This is done by calculating how close each block is to the edge of the environment, resulting in the following state evaluation  $SE$  [10]:

$$SE = 6 - \frac{12 \sum_{i=1}^B d_i}{B(n-1)}, \quad (1)$$

where  $B$  is the total number of blocks,  $n$  is the size of the grid and  $d_i$  is the distance of block  $i$  from an edge in the given instance. The value range of  $SE$  is consistent with the range of the original evaluation method; 0–6, allowing for direct comparison between the canonical  $6 \times 6$  grid and larger instance sizes.

A score near 0 would indicate that the agent has made no progress towards moving the blocks to the edge of the environment, or in some cases moved blocks closer to the centre in a counterintuitive manner. A score of 6 would indicate a state where all of the blocks in the instance have been successfully moved to the edges of the environment by the agent. At the end of each generation the agents use their resultant  $SE$  value as their fitness score.

## 4 Self-adaptive Crossover Operator

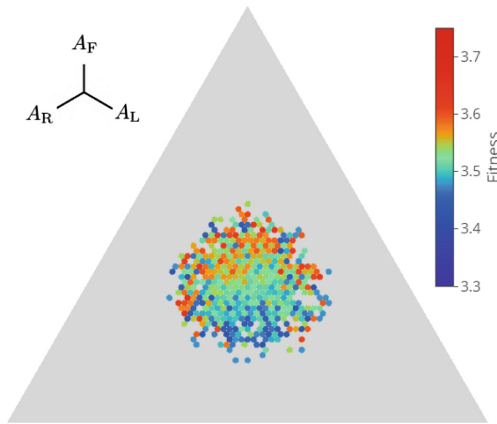
For a TP instance of size  $n = 6$ , an agent at the standard level of difficulty,  $D = 1$ , is allowed  $m = 80$  movement operations [10]. For linear GP these operations are encoded as a genome containing  $m$  alleles, with each allele corresponding to one of the three possible agent actions outlined in Sect. 3.

For each individual genome, the aggregate number of *move forward one square* ( $A_F$ ), *turn left* ( $A_L$ ) and *turn right* ( $A_R$ ) alleles are counted, these values make up the genome composition. It is important to note that this composition of the genome does not take into consideration the sequential order of the alleles, but only the aggregate number of each type of allele present. *We hypothesise that for each Tartarus instance there exist optimal compositions of agent actions,*

which, when used to seed future individuals, will likely lead to an increase in solution performance.

As the composition of the individual genome is made up of three primary components, it can be viewed on a ternary plot in order to visualise the magnitude of the components present in the composition. A population of 1000 individuals were generated, corresponding to 697 unique genome compositions. The population was executed across 100 different TP instances of size  $n = 6$ , and the resultant fitness scores averaged.

Analysis of the data showed there to be a clear divide in the average fitness scores between individuals who have an approximately equal composition, from the central region of the ternary plot, and those individuals with an uneven composition, who lie on the periphery. 80% of the compositions fall within the central region; here the variation in average fitness scores is low, with values ranging from 3.3–3.75, as shown in Fig. 3.



**Fig. 3.** The central 80% of compositions

However, for individuals who have an uneven composition, who fall outside of this central region; the variation in average fitness scores is high, with values ranging from 2.6–4.6. This is highlighted most clearly in Fig. 4, showing the bottom 10% and the top 10% of individual compositions in terms of averaged fitness score. It can be seen that the top 10% and bottom 10% of compositions exist in two defined bands surrounding the central region.

Upon further investigation, it was found that increasing the number of *move forward* instructions in the genome, relative to number of *turn left* and *turn right* instructions, leads to a noticeable increase in fitness score. This can be seen most notably in Fig. 3; there is a defined change in fitness scores between the compositions in the uppermost section of the plot, with higher  $A_F$ , and the compositions in the lower section of the plot, with lower  $A_F$ .

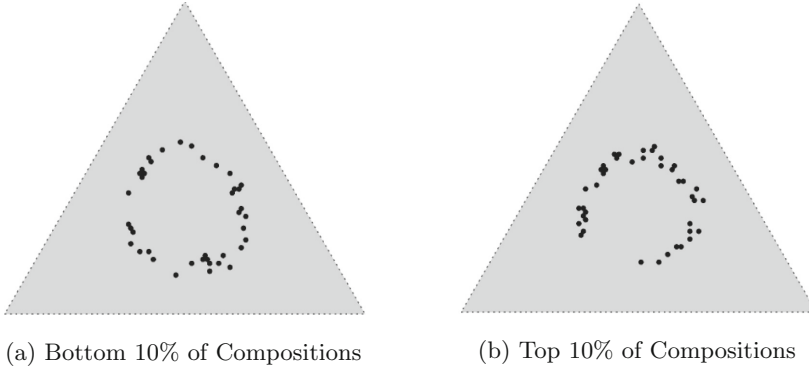


Fig. 4. Top and bottom 10% of compositions

This is expected behaviour, it is intuitive that compositions containing a high proportion of *turn left* or *turn right* instructions would simply spin around and not move far from the initial grid location, therefore having a lower score. In a similar manner, compositions containing a lower but approximately equal number of *turn left* and *turn right* instructions, the impact of these would effectively be cancelled out, resulting in a lower score.

We postulated that it would be possible to use this information to design a self-adaptive crossover bias in order to exploit the changes in expected fitness for different areas of the composition space. This would allow for the introduction of bias in the generation of new individuals by favouring offspring with certain compositions. As it is the output of the chosen crossover operator that is affected, the process of generating new individuals, the proposed self-adaptations can be incorporated and utilised alongside any traditional crossover approach.

In order to do this, the crossover operator was parameterised at the individual level. Each individual was assigned a random target  $A_F$  value  $T'_g$  during initialisation, in the range  $A_F = \frac{2}{5}m - \frac{4}{5}m$ , from where the value can adapt during evolution. The process of adapting the target value is divided into two stages. In the first stage, the ‘how’ stage, the target value  $T'_g$  is updated at the end of generation  $g$ , during the evaluation step, according to the performance of the individual in comparison to previous evaluations:

$$T'_g = \begin{cases} T_g & \text{if } F_g > F_{g-1} \\ T_g + R_g & \text{if } F_g \leq F_{g-1}, \end{cases} \tag{2}$$

where  $T_g$  is the current target value,  $F_g$  and  $F_{g-1}$  are the current and previous fitness scores of the individual and  $R_g$  is a uniformly distributed random value in the interval:

$$\left[ -\frac{A_{F_g}}{T_g}, \frac{A_{F_g}}{T_g} \right],$$



where  $A_{F_g}$  is the current  $A_F$  value in generation  $g$ . In the second stage, the ‘when’ stage, the probability of triggering the self-adaptation and implementing the new target value  $T'_g$  into the crossover parameters of the individual is calculated:

$$P(T'_g) = \frac{T_g}{G \cdot B \cdot T'_g}, \quad (3)$$

where  $G$  is the number of generations without an improvement in the fitness score of the individual and  $B$  is the number of blocks present in the instance.

The probability  $P(T'_g)$  is influenced by both the number of generations  $G$  since the actions of the individual led to an improvement in fitness score and the change between the target values  $T_g$  and  $T'_g$ . As  $G$  increases or the difference between  $T_g$  and  $T'_g$  increases, the chance that the self-adaptation will be triggered becomes greater. If the self-adaptation is triggered, at the start of the next generation,  $T_{g+1}$  will be initialised with the current value  $T'_g$ .

A population of 100 individuals was generated, each with a genome containing a random mixture of  $m = 142$  alleles. These individuals were tested on 100 instances of size  $n = 8$ . The target  $A_F$  values  $T$  chosen by the individual at each generation  $g$  were averaged. As shown in Fig. 5, over time, the target values chosen by the individuals within the population stabilise and converge to a small range of values. Figure 5 also shows the maximum and minimum  $T$  values within the population, over generations, until they converge.

It can be seen that by generation 18 the target values of all the individuals within the population have converged to approximately  $A_F = 95$ , for an instance of size  $n = 8$ . This indicates that allowing for the self-adaptation of the target value  $T$  leads to the creation of a crossover operator favouring individuals with compositions with close to optimal  $A_F$  values. From Fig. 5 we can conclude that an  $A_F$  value close to the optimal value is found.

The utilisation of the proposed self-adaptive crossover bias leads to an increase in both the *overall solution performance* and the *rate of solution improvement* in the Tartarus Problem. In Fig. 6 the performance of the self-adaptive crossover bias, averaged over 20 different TP instances of size  $n = 8$ , is plotted against the performance using standard canonical crossover. The range in fitness values present in the population at each generation is shown by the shaded areas, with the average score shown as solid lines.

The occurrences of self-adaptations being triggered within a population plotted against the changes in maximum fitness score, on a generation by generation basis is shown in Fig. 7. The plot shows that there is a strong correlation between the occurrence of self-adaptations within the population and an increase in the maximum fitness score achieved. We can conclude that the mechanisms by which the self-adaptation is calculated and triggered are effective, improving the performance of individuals in the population through the modification and manipulation of evolutionary pressures.

Between generation 11 and generation 12, 32% of the individuals in the population triggered self-adaptations of their target  $A_F$  value  $T_n$ . This led to an increase of 0.5 in the maximum fitness score of the population, bringing it

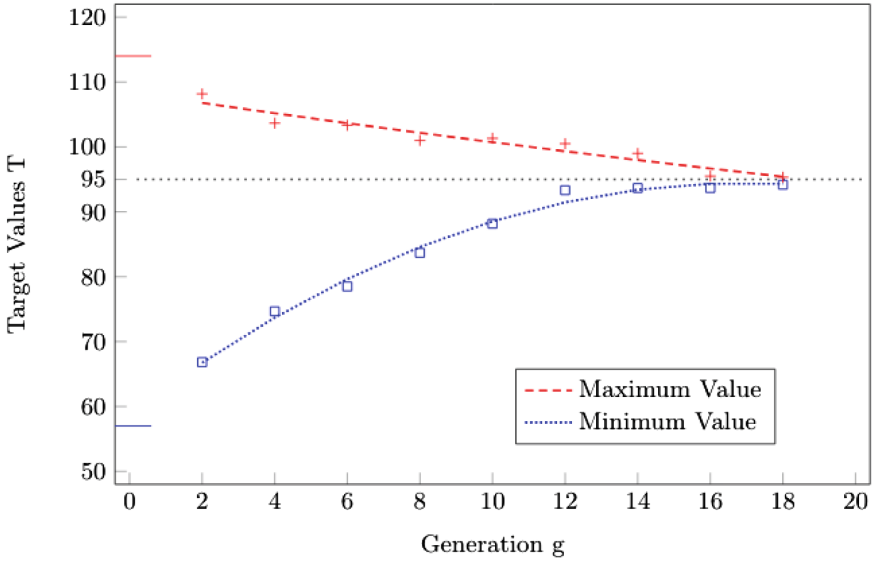


Fig. 5. Convergence of target  $A_F$  value T within the population

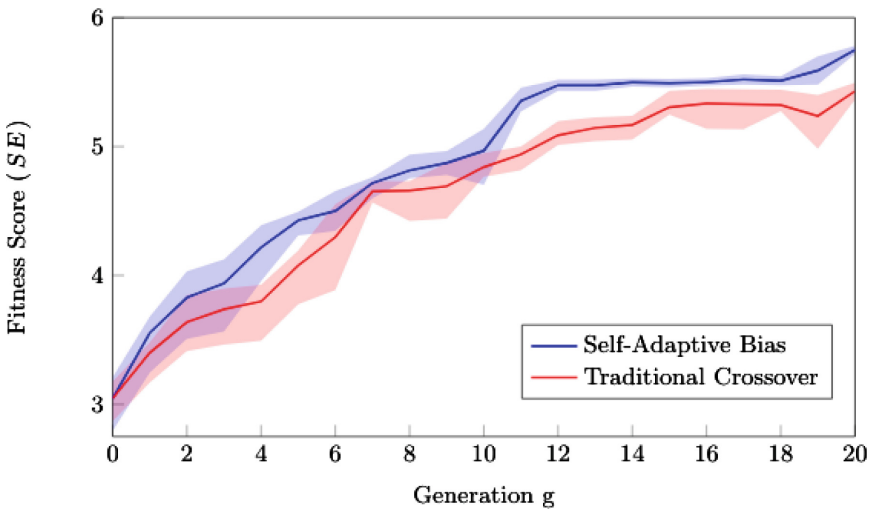


Fig. 6. Comparison between self-adaptive bias and traditional crossover

from 4.5 to 5.0. This is a substantial increase in the maximum fitness score of the population, a direct consequence of the self-adaptations carried out by the individuals.

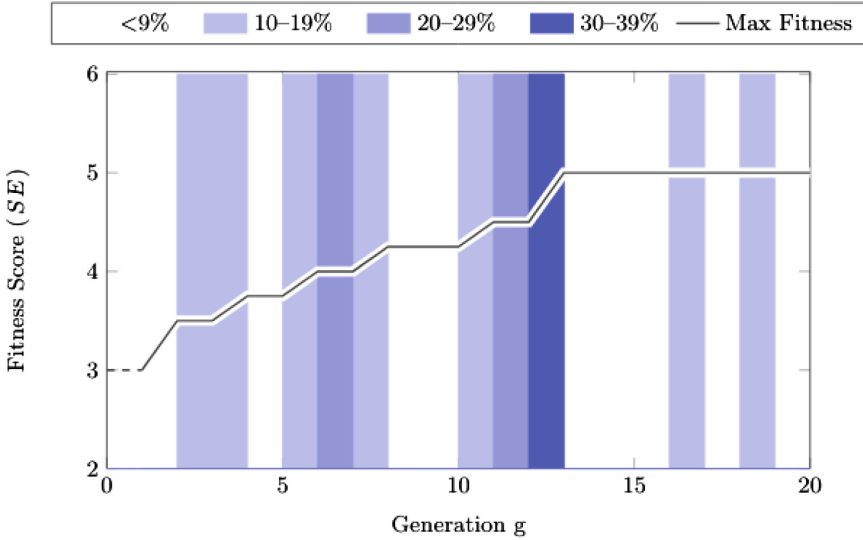


Fig. 7. Occurrences of self-adaptation and the maximum fitness score.

## 5 Conclusion

In this paper we outlined a novel approach to introducing *self-adaptation* into a crossover operator bias at the *individual level*.

The self-adaptation is triggered by the individual *as and when* required on a continual basis, rather than according to a pre-defined schedule or episodic time interval. The introduction of bias into the crossover operator, favouring offspring with certain compositions leads to convergence to solutions with higher average fitness scores.

We demonstrated that the individuals within the population were able to converge on a target parameter to be used by the crossover operator bias. This crossover bias was successfully utilised in order to generate solutions with higher average fitness score, when compared to solutions utilising traditional crossover operators.

The next step is to concentrate on testing the robustness of the proposed self-adaptation mechanism. Work will be conducted to test the applicability of the mechanism on other benchmark problems in order to ensure that it is generalisable and flexible. The long-term aim is to adapt and improve the self-adaptive mechanism so that it may be used on real world problems and applications.

## References

1. Eiben, A.E., Michalewicz, Z., Schoenauer, M., Smith, J.E.: Parameter control in evolutionary algorithms. In: Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.) *Parameter Setting in Evolutionary Algorithms*. Studies in Computational Intelligence, vol. 54, pp. 19–46. Springer, Berlin (2007). [https://doi.org/10.1007/978-3-540-69432-8\\_2](https://doi.org/10.1007/978-3-540-69432-8_2)
2. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. *Science* **220**, 671–680 (1983)
3. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1785–1791. IEEE (2005)
4. Hesser, J., Männer, R.: Towards an optimal mutation probability for genetic algorithms. In: Schwefel, H.-P., Männer, R. (eds.) *PPSN 1990*. LNCS, vol. 496, pp. 23–32. Springer, Heidelberg (1991). <https://doi.org/10.1007/BFb0029727>
5. Hansen, N, Ostermeier, A., Gawelczyk, A.: On the adaptation of arbitrary normal mutation distributions in evolution strategies: the generating set adaptation. In: Eshelman, L.J. (ed.) *Proceedings of the 6th International Conference on Genetic Algorithms, ICGA 1995*, pp. 57–64. Morgan Kaufmann (1995)
6. Hinterding, R., Michalewicz, Z., Peachey, T.C.: Self-adaptive genetic algorithm for numeric functions. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) *PPSN 1996*. LNCS, vol. 1141, pp. 420–429. Springer, Heidelberg (1996). [https://doi.org/10.1007/3-540-61723-X\\_1006](https://doi.org/10.1007/3-540-61723-X_1006)
7. Bäck, T.: The interaction of mutation rate, selection and self-adaptation within a genetic algorithm. In: *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature, PPSN II*, pp. 85–94 (1992)
8. Dang, D.-C., Lehre, P.K.: Self-adaptation of mutation rates in non-elitist populations. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) *PPSN 2016*. LNCS, vol. 9921, pp. 803–813. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_75](https://doi.org/10.1007/978-3-319-45823-6_75)
9. Teller, A.: The evolution of mental models. In: Kinnear Jr, K.E. (ed.) *Advances in Genetic Programming*, pp. 199–217 (1994)
10. Griffiths, T.D., Ekárt, A.: Improving the Tartarus problem as a benchmark in genetic programming. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E., García-Sánchez, P. (eds.) *EuroGP 2017*. LNCS, vol. 10196, pp. 278–293. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55696-3\\_18](https://doi.org/10.1007/978-3-319-55696-3_18)
11. White, D.R., et al.: Better GP benchmarks: community survey results and proposals. *Genet. Program. Evolvable Mach.* **14**(1), 3–29 (2013)
12. McDermott, J., et al.: Genetic programming needs better benchmarks. In: Soule, T., et al. (eds.) *Proceedings of the 14th International Conference on Genetic and Evolutionary Computation, GECCO 2012*, pp. 791–798 (2012)
13. Koza, J.R.: Scalable learning in genetic programming using automatic function definition. In: Kinnear Jr, K.E. (ed.) *Advances in Genetic Programming*, pp. 99–117 (1994)

# **Multi-objective Optimization**



# A Decomposition-Based Evolutionary Algorithm for Multi-modal Multi-objective Optimization

Ryoji Tanabe<sup>(✉)</sup> and Hisao Ishibuchi

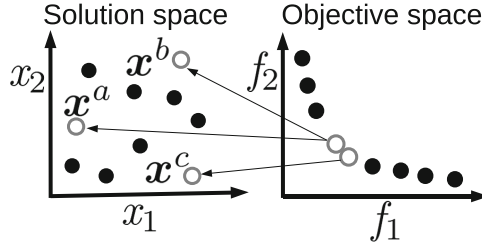
Shenzhen Key Laboratory of Computational Intelligence,  
Department of Computer Science and Engineering,  
Southern University of Science and Technology, Shenzhen, China  
rt.ryoji.tanabe@gmail.com, hisao@sustc.edu.cn

**Abstract.** This paper proposes a novel decomposition-based evolutionary algorithm for multi-modal multi-objective optimization, which is the problem of locating as many as possible (almost) equivalent Pareto optimal solutions. In the proposed method, two or more individuals can be assigned to each decomposed subproblem to maintain the diversity of the population in the solution space. More precisely, a child is assigned to a subproblem whose weight vector is closest to its objective vector, in terms of perpendicular distance. If the child is close to one of individuals that have already been assigned to the subproblem in the solution space, the replacement selection is performed based on their scalarizing function values. Otherwise, the child is newly assigned to the subproblem, regardless of its quality. The effectiveness of the proposed method is evaluated on seven problems. Results show that the proposed algorithm is capable of finding multiple equivalent Pareto optimal solutions.

## 1 Introduction

A multi-objective optimization problem (MOP) is the problem of finding a solution  $\mathbf{x} = (x_1, \dots, x_D)^T \in \mathbb{S}$  that minimizes an objective function vector  $\mathbf{f} : \mathbb{S} \rightarrow \mathbb{R}^M$ . Here,  $\mathbb{S}$  is the  $D$ -dimensional solution space, and  $\mathbb{R}^M$  is the  $M$ -dimensional objective space. Usually,  $\mathbf{f}$  consists of  $M$  conflicting objective functions. A solution  $\mathbf{x}^1$  is said to dominate  $\mathbf{x}^2$  iff  $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2)$  for all  $i \in \{1, \dots, M\}$  and  $f_i(\mathbf{x}^1) < f_i(\mathbf{x}^2)$  for at least one index  $i$ . If there exists no  $\mathbf{x}$  in  $\mathbb{S}$  such that  $\mathbf{x}$  dominates  $\mathbf{x}^*$ ,  $\mathbf{x}^*$  is called a Pareto optimal solution. The set of all  $\mathbf{x}^*$  is the Pareto optimal solution set, and the set of all  $\mathbf{f}(\mathbf{x}^*)$  is the Pareto front. The goal of MOPs is usually to find a set of nondominated solutions that approximates the Pareto front well in the objective space.

An evolutionary multi-objective optimization algorithm (EMOA) is an efficient population-based optimization method to approximate the Pareto front of a given MOP in a single run [1]. Although several paradigms of EMOAs (e.g., dominance-based EMOAs) have been proposed, decomposition-based EMOAs



**Fig. 1.** Illustration of a situation where three solutions are identical or close to each other in the objective space but are far from each other in the solution space. This figure was made using [8,13] as reference.

are recently popular in the EMO community. In particular, MOEA/D [18] is one of the most representative decomposition-based EMOAs [14].

There are multiple equivalent Pareto optimal solutions in some real-world problems (e.g., space mission design problems [12], rocket engine design problems [7], and path-planning problems [17]). Figure 1 explains such a situation. Diverse solutions are helpful for decision-making [3, 11–13]. If two or more solutions having (almost) the same objective vector are found, users can make a final decision according to their preference which cannot be represented by the objective functions. For example, in Fig. 1, if  $\mathbf{x}^a$  becomes unavailable for some reasons (e.g., materials shortages and traffic accidents),  $\mathbf{x}^b$  and  $\mathbf{x}^c$  can be candidates for the final solution instead of  $\mathbf{x}^a$ .

A multi-modal MOP (MMOP) [3, 8, 17] is the problem of locating as many as possible (almost) equivalent Pareto optimal solutions. Unlike the general MOPs, the goal of MMOPs is to find a good approximation of the Pareto-optimal solution set. For example, in Fig. 1, it is sufficient to find one of  $\mathbf{x}^a$ ,  $\mathbf{x}^b$ , and  $\mathbf{x}^c$  for MOPs, because their objective vectors are almost the same. In contrast, EMOAs need to find all of  $\mathbf{x}^a$ ,  $\mathbf{x}^b$ , and  $\mathbf{x}^c$  for MMOPs. Some EMOAs for MMOPs have been proposed in the literature (e.g., [3, 6, 12, 13, 17]).

While MMOPs can be found in real-world problems [7, 12, 17], it is likely that most MOEA/D-type algorithms [14] are not capable of locating multiple equivalent Pareto optimal solutions. This is because they do not have any explicit mechanism to maintain the diversity of the population in the solution space. If the ability to keep solution space diversity in the population is incorporated into MOEA/D, an efficient multi-modal multi-objective optimizer may be realized.

This paper proposes a novel MOEA/D algorithm with addition and deletion operators (MOEA/D-AD) for multi-modal multi-objective optimization. In MOEA/D-AD, the population size  $\mu$  is dynamically changed during the search process. Multiple individuals that are far from each other in the solution space can be assigned to the same decomposed single-objective subproblem. Only similar individuals in the solution space are compared based on their scalarizing function values. Thus, MOEA/D-AD maintains the diversity in the population by performing environmental selection for each subproblem among individuals that are close to each other in the solution space.

---

**Algorithm 1.** The procedure of MOEA/D-AD

---

```

1  $t \leftarrow 1$ , initialize the population  $\mathbf{P} = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ ;
2 for  $i \in \{1, \dots, N\}$  do Assign  $\mathbf{x}^i$  to the  $i$ -th subproblem;
3 while The termination criteria are not met do
4      $\mu \leftarrow |\mathbf{P}|$ ;
5     Randomly select  $r_1$  and  $r_2$  from  $\{1, \dots, \mu\}$  such that  $r_1 \neq r_2$ ;
6     Generate the child  $\mathbf{u}$  by recombining  $\mathbf{x}^{r_1}$  and  $\mathbf{x}^{r_2}$ ;
7     Apply the mutation operator to  $\mathbf{u}$ ;
8     for  $i \in \{1, \dots, N\}$  do  $d_i \leftarrow \text{PD}(\mathbf{f}'(\mathbf{u}), \mathbf{w}^i)$ ;
9      $j \leftarrow \arg \min_{i \in \{1, \dots, N\}} \{d_i\}$ ;
10     $b^{\text{winner}} \leftarrow \text{FALSE}$  and  $b^{\text{explorer}} \leftarrow \text{TRUE}$ ;
11    for  $\mathbf{x} \in \mathbf{P} \mid \mathbf{x}$  has been assigned to the  $j$ -th subproblem do
12        if  $\text{isNeighborhood}(\mathbf{u}, \mathbf{x}) = \text{TRUE}$  then
13             $b^{\text{explorer}} \leftarrow \text{FALSE}$ ;
14            if  $g(\mathbf{u}|\mathbf{w}^j) \leq g(\mathbf{x}|\mathbf{w}^j)$  then
15                 $\mathbf{P} \leftarrow \mathbf{P} \setminus \{\mathbf{x}\}$  and  $b^{\text{winner}} \leftarrow \text{TRUE}$ ;
16    if  $b^{\text{winner}} = \text{TRUE}$  or  $b^{\text{explorer}} = \text{TRUE}$  then
17         $\mathbf{P} \leftarrow \mathbf{P} \cup \{\mathbf{u}\}$  and assign  $\mathbf{u}$  to the  $j$ -th subproblem;
18     $t \leftarrow t + 1$ ;
19  $\mathbf{A} \leftarrow \text{selectSparseSolutions}(\mathbf{P})$ ;
20 return  $\mathbf{A}$ ;

```

---

This paper is organized as follows. Section 2 introduces MOEA/D-AD. Section 3 describes experimental setup. Section 4 presents experimental results of MOEA/D-AD, including performance comparison and its analysis. Section 5 concludes this paper with discussions on directions for future work.

## 2 Proposed MOEA/D-AD

MOEA/D decomposes a given  $M$ -objective MOP into  $N$  single-objective subproblems using a scalarizing function  $g : \mathbb{R}^M \rightarrow \mathbb{R}$  and a set of uniformly distributed weight vectors  $\mathbf{W} = \{\mathbf{w}^1, \dots, \mathbf{w}^N\}$ , where  $\mathbf{w}^i = (w_1^i, \dots, w_M^i)^T$  for each  $i \in \{1, \dots, N\}$ , and  $\sum_{j=1}^M w_j^i = 1$ . One individual in the population  $\mathbf{P}$  is assigned to each decomposed subproblem. Thus, the population size  $\mu$  of MOEA/D always equals  $N$  (i.e.,  $\mu = N$  and  $|\mathbf{P}| = |\mathbf{W}|$ ).

Algorithm 1 shows the procedure of the proposed MOEA/D-AD for multimodal multi-objective optimization. While the number of subproblems  $N$  is still constant in MOEA/D-AD,  $\mu$  is nonconstant. Although it is ensured that  $\mu \geq N$ ,  $\mu$  is dynamically changed during the search process (i.e.,  $\mu \neq N$  and  $|\mathbf{P}| \neq |\mathbf{W}|$ ) unlike MOEA/D. After the initialization of the population (lines 1–2), the following steps are repeatedly applied until a termination condition is satisfied.



---

**Algorithm 2.** The  $\text{isNeighborhood}(\mathbf{u}, \mathbf{x})$  function

---

```

/* The population  $P = \{\mathbf{y}^1, \dots, \mathbf{y}^\mu\}$  */
1 for  $i \in \{1, \dots, \mu\}$  do  $d_i^E \leftarrow \text{NED}(\mathbf{y}^i, \mathbf{u})$ ;
2 Sort individuals based on their distance values such that  $d_1^E \leq d_2^E \leq \dots \leq d_\mu^E$ ;
3 for  $i \in \{1, \dots, L\}$  do
4   if  $\mathbf{y}^i = \mathbf{x}$  then return TRUE;
5 return FALSE;
```

---

At the beginning of each iteration, parent individuals are selected from the whole population  $\mathbf{P}$  (line 5). Unlike the original MOEA/D, the mating selection is not restricted to neighborhood individuals to generate diverse new solutions. Then, a child  $\mathbf{u}$  is generated by applying the variation operators (lines 6–7).

After  $\mathbf{u}$  has been generated, the environmental selection is performed (lines 8–17). Note that our subproblem selection method (described below) was derived from MOEA/D-DU [16]. However, unlike MOEA/D-DU, only one subproblem is updated for each iteration in MOEA/D-AD to preserve the diversity. First, the perpendicular distance  $d_i$  between the normalized objective vector  $\mathbf{f}'(\mathbf{u})$  of  $\mathbf{u}$  and  $\mathbf{w}^i$  is calculated for each  $i \in \{1, \dots, N\}$  (line 8), where PD represents the perpendicular distance between two input vectors. Here,  $\mathbf{f}'(\mathbf{u})$  is obtained as follows:  $f'_k(\mathbf{u}) = (f_k(\mathbf{u}) - f_k^{\min}) / (f_k^{\max} - f_k^{\min})$ , where  $f_k^{\min} = \min_{\mathbf{y} \in \mathbf{P}} \{f_k(\mathbf{y})\}$ , and  $f_k^{\max} = \max_{\mathbf{y} \in \mathbf{P}} \{f_k(\mathbf{y})\}$  for each  $k \in \{1, \dots, M\}$ . Then, the  $j$ -th subproblem having the minimum  $d$  value is selected (line 9). The environmental selection is performed only on the  $j$ -th subproblem.

The child  $\mathbf{u}$  is compared to all the individuals that have been assigned to the  $j$ -th subproblem (line 11–15). Two Boolean variables  $b^{\text{winner}} \in \{\text{TRUE}, \text{FALSE}\}$  (line 10) are used for the addition operation of MOEA/D-AD. More precisely,  $b^{\text{winner}}$  represents whether  $\mathbf{u}$  outperforms at least one individual belonging to the  $j$ -th subproblem regarding the scalarizing function value, and  $b^{\text{explorer}}$  indicates whether  $\mathbf{u}$  is far from all the individuals assigned to the  $j$ -th subproblem in the solution space. If at least one of  $b^{\text{winner}}$  and  $b^{\text{explorer}}$  is TRUE,  $\mathbf{u}$  enters the population  $\mathbf{P}$  (lines 16–17).

In line 12 of Algorithm 1, the  $\text{isNeighborhood}(\mathbf{u}, \mathbf{x})$  function returns TRUE if  $\mathbf{u}$  is close to  $\mathbf{x}$  in the solution space (otherwise, it returns FALSE). Algorithm 2 shows details of the function, where the NED function returns the normalized Euclidean distance between two input vectors using the upper and lower bounds for each variable of a given problem. In Algorithm 2,  $L$  ( $1 \leq L \leq \mu$ ) is a control parameter of MOEA/D-AD. First, the normalized Euclidean distance between each individual in  $\mathbf{P}$  and  $\mathbf{u}$  is calculated. Then, all the  $\mu$  individuals are sorted based on their distance values in descending order. Finally, if  $\mathbf{x}$  is within the  $L$  nearest individuals from  $\mathbf{u}$  among the  $\mu$  individuals, the function returns TRUE.

If  $\mathbf{x}$  is in the neighborhood of  $\mathbf{u}$  in the solution space (line 12), they are compared based on their scalarizing function values (lines 14–15). The environmental selection is performed only among similar individuals in the solution space in

---

**Algorithm 3.** The selectSparseSolutions( $\mathbf{P}$ ) function

---

```

/* The population  $P = \{x^1, \dots, x^\mu\}$  */
1  $\mathbf{P} \leftarrow$  selectNondominatedSolutions( $\mathbf{P}$ ),  $\mu \leftarrow |\mathbf{P}|$ ,  $\mathbf{A} \leftarrow \emptyset$ ;
2 for  $i \in \{1, \dots, \mu\}$  do  $b_i^{\text{selected}} \leftarrow$  FALSE,  $D_i \leftarrow \infty$ ;
3 Randomly select  $j$  from  $\{1, \dots, \mu\}$ ;
4  $\mathbf{A} \leftarrow \mathbf{A} \cup \{x^j\}$ ,  $b_j^{\text{selected}} \leftarrow$  TRUE;
5 for  $i \in \{1, \dots, \mu\}$  do
6   if  $b_i^{\text{selected}} =$  FALSE then  $D_i \leftarrow \min(\text{NED}(x^i, x^j), D_i)$ ;
7 while  $|\mathbf{A}| < N$  do
8    $j \leftarrow \arg \max_{i \in \{1, \dots, \mu\} | b_i^{\text{selected}} = \text{FALSE}} D_i$ ,  $\mathbf{A} \leftarrow \mathbf{A} \cup \{x^j\}$ ,  $b_j^{\text{selected}} \leftarrow$  TRUE;
9   for  $i \in \{1, \dots, \mu\}$  do
10    if  $b_i^{\text{selected}} =$  FALSE then  $D_i \leftarrow \min(\text{NED}(x^i, x^j), D_i)$ ;
11 return  $\mathbf{A}$ , which is the  $N$  or fewer nondominated solutions selected from  $\mathbf{P}$ ;

```

---

order to maintain the diversity of the population. If  $x$  is worse than  $u$ ,  $x$  is removed from  $\mathbf{P}$  (line 15). This is the deletion operation of MOEA/D-AD.

Since  $\mu$  is not bounded,  $\mathbf{P}$  may include a large number of solutions at the end of the search. This is undesirable in practice because decision-makers are likely to want to examine only a small number of nondominated solutions that approximate the Pareto front and the Pareto solution set [18]. To address this issue, a method of selecting  $N$  nondominated solutions is applied to the final population (line 19). Recall that  $N$  denotes the number of subproblems.

Algorithm 3 shows details of the selectSparseSolutions function, which returns  $N$  or less nondominated solutions  $\mathbf{A}$ . First, nondominated solutions are selected from  $\mathbf{P}$ . Then, one individual is randomly selected from  $\mathbf{P}$  and inserted into  $\mathbf{A}$ . Then, a solution having the maximum distance to solutions in  $\mathbf{A}$  is repeatedly stored into  $\mathbf{A}$ . It is expected that a set of nondominated solutions being far from each other in the solution space are obtained by this procedure.

### 3 Experimental Settings

**Test Problems.** We used the following seven two-objective MMOPs: the Two-On-One problem [10], the Omni-test problem [3], the three SYM-PART problems [11], and the two SSUF problems [9]. The number of variables  $D$  is five for the Omni-test problem and two for the other problems. In the Two-On-One and SSUF1 problems, there are two symmetrical Pareto optimal solutions that are mapped to the same objective vector. In the other problems, Pareto optimal solutions are regularly distributed. The number of equivalent Pareto optimal solutions is two for the SSUF3 problem, nine for the three SYM-PART problems, and 45 for the Omni-test problem.

**Performance Indicators.** We used the inverted generational distance (IGD) [20] and IGDX [19] for performance assessment of EMOAs. Below,  $\mathbf{A}$  denotes a set of nondominated solutions of the final population of an EMOA. The IGD and IGDX metrics require a set of reference points  $\mathbf{A}^*$ . For  $\mathbf{A}^*$  for each problem, we used 5 000 solutions which were selected from randomly generated 10 000 Pareto-optimal solutions by using the `selectSparseSolutions` function (Algorithm 3).

The IGD value is the average distance from each reference solution in  $\mathbf{A}^*$  to its nearest solution in  $\mathbf{A}$  in the objective space as follows:

$$\text{IGD}(\mathbf{A}) = \frac{1}{|\mathbf{A}^*|} \left( \sum_{z \in \mathbf{A}^*} \min_{x \in \mathbf{A}} \{ \text{ED}(f(x), f(z)) \} \right),$$

where  $\text{ED}(x^1, x^2)$  represents the Euclidean distance between  $x^1$  and  $x^2$ .

Similarly, the IGDX value of  $\mathbf{A}$  is given as follows:

$$\text{IGDX}(\mathbf{A}) = \frac{1}{|\mathbf{A}^*|} \left( \sum_{z \in \mathbf{A}^*} \min_{x \in \mathbf{A}} \{ \text{ED}(x, z) \} \right).$$

While IGD measures the quality of  $\mathbf{A}$  in terms of both convergence to the Pareto front and diversity in the objective space, IGDX evaluates how well  $\mathbf{A}$  approximates the Pareto-optimal solution set in the solution space. Thus, EMOAs that can find  $\mathbf{A}$  with small IGD and IGDX values are efficient multi-objective optimizers and multi-modal multi-objective optimizers, respectively. It should be noted that small IGD values do not always mean small IGDX values.

**Setup for EMOAs.** We compared MOEA/D-AD with the following five methods: MO\_Ring\_PSO\_SCD [17], Omni-optimizer [3], NSGA-II [2], MOEA/D [18], and MOEA/D-DU [16]. Omni-optimizer is a representative EMOA for MMOPs. MO\_Ring\_PSO\_SCD is a recently proposed PSO algorithm for MMOPs. NSGA-II and MOEA/D are widely used EMOAs for MOPs. Since the selection method of the subproblem to be updated in MOEA/D-AD was derived from MOEA/D-DU, we also included it in our experiments.

Available source code through the Internet were used for algorithm implementation. For the implementation of MOEA/D-AD, we used the jMetal framework [4]. Source code of MOEA/D-AD can be downloaded from the first author's website (<https://ryojitanabe.github.io/>). The population size  $\mu$  and the number of weight vectors  $N$  were set to 100 for all the methods. In MOEA/D-AD,  $\mu$  is dynamically changed as shown in Fig. 4(a). For a fair comparison, we used a set of nondominated solutions of the size  $N = 100$  selected from the final population by using the `selectSparseSolutions` function (Algorithm 3). Thus, the EMOAs were compared using the obtained solution sets of the same size (100). For all the six EMOAs, the number of maximum function evaluations was set to 30 000, and 31 runs were performed. The SBX crossover and the polynomial mutation were used in all the EMOAs (except for MO\_Ring\_PSO\_SCD). Their control parameters were set as follows:  $p_c = 1$ ,  $\eta_c = 20$ ,  $p_m = 1/D$ , and  $\eta_m = 20$ .

**Table 1.** Results of the six EMOAs on the seven MMOPs. The tables (a) and (b) show the mean IGD and IGDX values, respectively. The best and second best data are represented by the bold and italic font. The numbers in parenthesis indicate the ranks of the EMOAs. The symbols +, −, and  $\approx$  indicate that a given EMOA performs significantly better (+), significantly worse (−), and not significantly better or worse ( $\approx$ ) compared to MOEA/D-AD according to the Wilcoxon rank-sum test with  $p < 0.05$ .

	MOEA/ D-AD	MO_Ring_ PSO_SCD	Omni- optimizer	NSGA-II	MOEA/D	MOEA/ D-DU
(a) IGD						
Two-On-One	0.0637 (5)	0.0606 $\approx$ (4)	<i>0.0489+</i> (2)	0.0490+ (3)	<b>0.0450+</b> (1)	0.0709− (6)
Omni-test	0.0755 (5)	0.1814− (6)	<i>0.0303+</i> (2)	<b>0.0297+</b> (1)	0.0517+ (4)	0.0458+ (3)
SYM-PART1	0.0302 (4)	0.0283+ (3)	<i>0.0236+</i> (2)	<b>0.0210+</b> (1)	0.0467− (5)	0.0478− (6)
SYM-PART2	0.0305 (3)	0.0312 $\approx$ (4)	<i>0.0284+</i> (2)	<b>0.0229+</b> (1)	0.0466− (5)	0.0474− (6)
SYM-PART3	<i>0.0307</i> (2)	0.0323− (3)	0.0343− (4)	<b>0.0228+</b> (1)	0.0455− (5)	0.0470− (6)
SSUF1	0.0075 (6)	0.0065+ (5)	0.0060+ (4)	<i>0.0055+</i> (2)	0.0055+ (3)	<b>0.0042+</b> (1)
SSUF3	0.0190 (5)	0.0106+ (3)	0.0170+ (4)	<b>0.0073+</b> (1)	0.0629− (6)	<i>0.0082+</i> (2)
(b) IGDX						
Two-On-One	<b>0.0353</b> (1)	<i>0.0369−</i> (2)	0.0383− (3)	0.1480− (4)	0.2805− (6)	0.2067− (5)
Omni-test	<b>1.3894</b> (1)	2.2227− (3)	<i>2.0337−</i> (2)	2.5664− (4)	4.3950− (6)	2.9251− (5)
SYM-PART1	<b>0.0686</b> (1)	<i>0.1482−</i> (2)	3.8027− (3)	7.9287− (5)	9.1551− (6)	5.0426− (4)
SYM-PART2	<b>0.0783</b> (1)	<i>0.1610−</i> (2)	1.0863− (3)	5.3711− (5)	9.4834− (6)	5.1610− (4)
SYM-PART3	<b>0.1480</b> (1)	<i>0.4909−</i> (2)	1.3620− (3)	5.8410− (5)	7.3969− (6)	4.6767− (4)
SSUF1	<b>0.0761</b> (1)	<i>0.0860−</i> (2)	0.0899− (3)	0.1323− (5)	0.2443− (6)	0.1143− (4)
SSUF3	<i>0.0302</i> (2)	<b>0.0198+</b> (1)	0.0541− (3)	0.0710− (5)	0.3083− (6)	0.0599− (4)

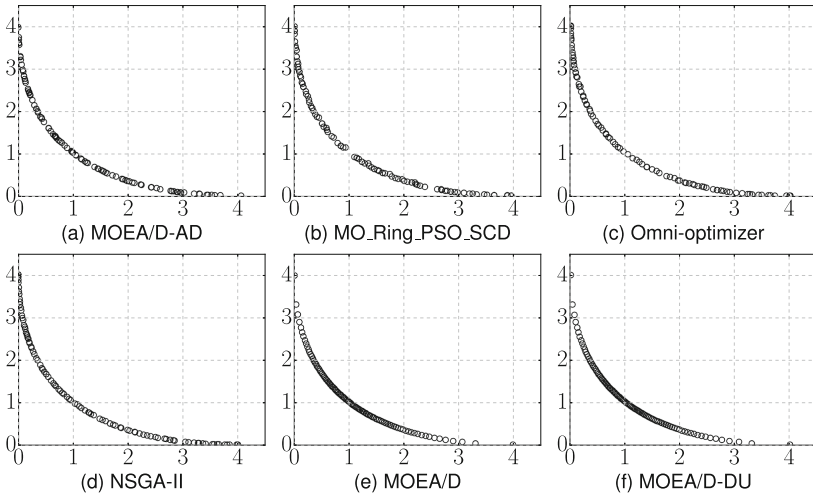
We used the Tchebycheff function [18] for MOEA/D and MOEA/D-AD as the scalarizing function. The control parameter  $L$  of MOEA/D-AD was set to  $L = \lfloor 0.1\mu \rfloor$  (e.g.,  $L = 201$  when  $\mu = 2018$ ). According to [16, 18], the neighborhood size  $T$  of MOEA/D and MOEA/D-DU was set to  $T = 20$ . All other parameters of MOEA/D-DU and MO\_Ring\_PSO\_SCD were set according to [16, 17].

## 4 Experimental Results

### 4.1 Performance Comparison

**IGD Metric.** Table 1 shows the comparison of the EMOAs on the seven problems. The IGD and IGDX values are reported in Table 1(a) and (b), respectively.

Table 1(a) shows that the performance of NSGA-II regarding the IGD metric is the best on five problems. MOEA/D and MOEA/D-DU also perform best on the Two-On-One and SSUF1 problems, respectively. In contrast, MOEA/D-AD and MO\_Ring\_PSO\_SCD perform poorly on most problems. Note that such a poor performance of multi-modal multi-objective optimizers for multi-objective optimization has already been reported in [13, 17]. Since multi-modal multi-objective optimizers try to locate all equivalent Pareto optimal solutions, their ability to find a good approximation of the Pareto front is usually worse than



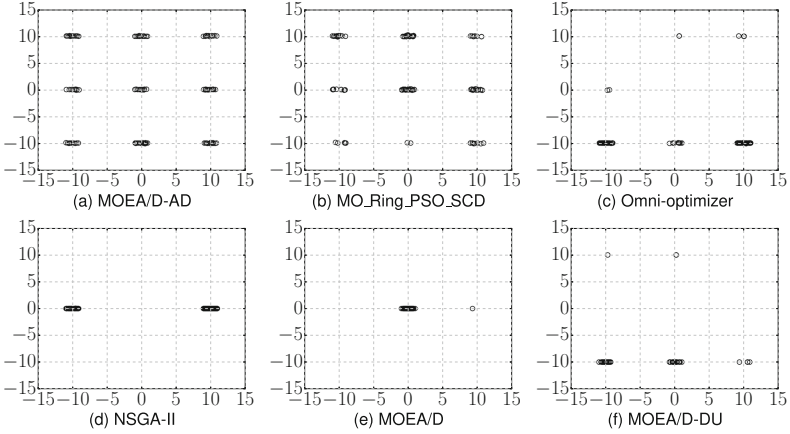
**Fig. 2.** Distribution of nondominated solutions in the final population of each EMOA in the objective space on the SYM-PART1 problem. The horizontal and vertical axis represent  $f_1$  and  $f_2$ , respectively.

that of multi-objective optimizers, which directly approximate the Pareto front. However, the IGD values achieved by MOEA/D-AD are only 1.3–2.6 times worse than the best IGD values on all the problems.

**IGDX Metric.** Table 1(b) indicates that the three multi-modal multi-objective optimizers (MOEA/D-AD, MO\_Ring\_PSO\_SCD, and Omni-optimizer) have good performance, regarding the IGDX indicator. In particular, MOEA/D-AD performs the best on the six MMOPs. MOEA/D-AD shows the second best performance only on the SSUF3 problem. In contrast, the performance of MOEA/D and MOEA/D-DU regarding the IGDX metric is quite poor. The IGDX values obtained by MOEA/D are 3.2–121.1 times worse than those by MOEA/D-AD. Thus, the new mechanism that maintains the solution space diversity in the population mainly contributes to the effectiveness of MOEA/D-AD.

**Distribution of Solutions Found.** Figures 2 and 3 show the distribution of nondominated solutions in the final population of each EMOA in the objective and solution spaces on the SYM-PART1 problem. Again, we emphasize that only  $N = 100$  nondominated solutions selected from the final population by using the selectSparseSolutions function (Algorithm 3) are shown for MOEA/D-AD in Figs. 2 and 3. Results of a single run with median IGD and IGDX values among 31 runs are shown in Figs. 2 and 3, respectively.

As shown in Fig. 2, the Pareto front of the SYM-PART1 problem is convex. While the distribution of nondominated solutions found by MOEA/D and MOEA/D-DU in the objective space is biased to the center of the Pareto front,



**Fig. 3.** Distribution of nondominated solutions in the final population of each EMOA in the solution space on the SYM-PART1 problem. The horizontal and vertical axis represent  $x_1$  and  $x_2$ , respectively.

that by NSGA-II is uniform. Compared to the result of NSGA-II, nondominated solutions obtained by MOEA/D-AD and MO\_Ring\_PSO\_SCD are not uniformly distributed in the objective space. This is because they also take into account the diversity of the population in the solution space.

The Pareto optimal solutions are on the nine lines in the SYM-PART1 problem. Figure 3 shows that Omni-optimizer, NSGA-II, MOEA/D, and MOEA/D-DU fail to locate all the nine equivalent Pareto optimal solution sets. Solutions obtained by the four methods are only on a few lines. In contrast, MOEA/D-AD and MO\_Ring\_PSO\_SCD successfully find nondominated solutions on all the nine lines. In particular, solutions obtained by MOEA/D-AD are more evenly distributed on the nine lines. Similar results to Figs. 2 and 3 are observed in other test problems. In summary, our results indicate that MOEA/D-AD is an efficient method for multi-modal multi-objective optimization.

## 4.2 Analysis of MOEA/D-AD

**Influence of  $L$  on the Performance of MOEA/D-AD.** MOEA/D-AD has the control parameter  $L$ , which determines the neighborhood size of the child in the solution space. Generally speaking, it is important to understand the influence of control parameters on the performance of a novel evolutionary algorithm. Here, we investigate how  $L$  affects the effectiveness of MOEA/D-AD.

Table 2 shows results of MOEA/D-AD with six  $L$  values on the seven test problems. Due to space constraint, only aggregations of statistical testing results to MOEA/D-AD with  $L = [0.1\mu]$  are shown here. Intuitively, MOEA/D-AD with a large  $L$  value should perform well regarding the IGD metric because large  $L$  values relax the restriction for the environmental selection and may

**Table 2.** Results of MOEA/D-AD with various  $L$  values on the seven MMOPs. The tables (a) and (b) show aggregations of statistical testing results (+, −, and  $\approx$ ) of the IGD and IGDX metrics. Each entry in the table shows the number of problems where the performance of MOEA/D-AD with each value of  $L$  is significantly better (worse) or has no significant difference from that of MOEA/D-AD with  $L = \lfloor 0.1\mu \rfloor$ .

	$0.1\mu$	$0.05\mu$	$0.2\mu$	$0.3\mu$	$0.4\mu$	$0.5\mu$
(a) IGD						
+ (better)	1	0	0	0	0	0
− (worse)	0	3	5	5	5	5
$\approx$ (no sig.)	6	4	2	2	2	2
(b) IGDX						
+ (better)	2	1	0	0	0	0
− (worse)	3	2	5	6	7	7
$\approx$ (no sig.)	2	4	2	1	0	0

improve its ability for multi-objective optimization. However, Table 2(a) shows that the performance of MOEA/D-AD with a large  $L$  value is poor, regarding the IGD indicator. As pointed out in [15], the solution space diversity may help MOEA/D-AD to approximate the Pareto front well.

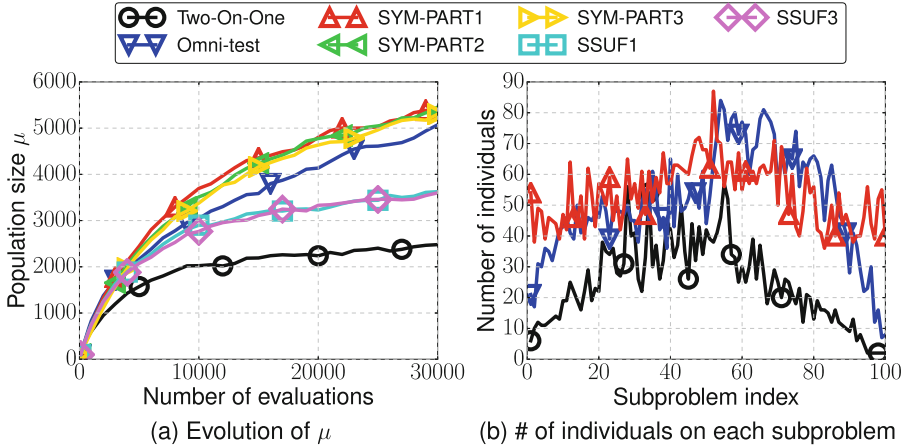
Table 2(b) indicates that the best IGDX values are obtained by  $L = \lfloor 0.05\mu \rfloor$  and  $L = \lfloor 0.2\mu \rfloor$  on two problems and one problem, respectively. Thus, the performance of MOEA/D-AD depends on the  $L$  value. A control method of  $L$  is likely to be beneficial for MOEA/D-AD. However, MOEA/D-AD with  $L = \lfloor 0.1\mu \rfloor$  performs well on most problems. Therefore,  $L = \lfloor 0.1\mu \rfloor$  can be the first choice.

**Adaptive Behavior of MOEA/D-AD.** Unlike other MOEA/D-type algorithms, the population size  $\mu$  and the number of individuals belonging to each subproblem are adaptively adjusted in MOEA/D-AD.

Figure 4(a) shows the evolution of  $\mu$  of MOEA/D-AD on the seven problems. In Fig. 4(a),  $\mu$  is increased as the search progresses. This is because the diverse individuals in the solution space are iteratively added in the population. Recall that the number of equivalent Pareto optimal solutions  $n^{\text{same}}$  is 45 for the Omni-test problem, nine for the three SYM-PART problems, and two for other problems. Figure 4(a) indicates that the trajectory of  $\mu$  is problem-dependent. Ideally,  $\mu$  should equal  $n^{\text{same}} \times N$  so that  $n^{\text{same}}$  Pareto optimal solutions are assigned to each of  $N$  subproblems. However, the actual  $\mu$  values are significantly larger than the expected values. For example, while the ideal  $\mu$  value is 200 ( $2 \times 100$ ) on the Two-On-One problem, the actual  $\mu$  value at the end of the search is 2480.

To analyze the reason, we show the number of individuals assigned to each subproblem at the end of the search on the Two-On-One, Omni-test, and

SYM-PART1 problems in Fig. 4(b). Figure 4(b) indicates that the distribution of individuals is not even. More extra individuals are allocated to subproblems whose indices are close to 50. That is, MOEA/D-AD allocates unnecessary individuals to most subproblems. If individuals can be evenly assigned to each subproblem, the performance of MOEA/D-AD may be improved. An in-depth analysis of the adaptive behavior of MOEA/D-AD is needed.



**Fig. 4.** (a) Evolution of the population size  $\mu$  of MOEA/D-AD. (b) Number of individuals assigned to the  $j$ -th subproblem ( $j \in \{1, \dots, N\}$ ) at the end of the search, where  $N = 100$ . Results of a single run with a median IGDX value among 31 runs are shown.

## 5 Conclusion

We proposed MOEA/D-AD, which is a novel MOEA/D for multi-modal multi-objective optimization. In order to locate multiple equivalent Pareto optimal solutions, MOEA/D-AD assigns one or more individuals that are far from each other in the solution space to each subproblem. We examined the performance of MOEA/D-AD on the seven two-objective problems having equivalent Pareto optimal solutions. Our results indicate that MOEA/D-AD is capable of finding multiple equivalent Pareto optimal solutions. The results also show that MOEA/D-AD performs significantly better than Omni-optimizer and MO\_Ring\_PSO\_SCD, which are representative multi-modal multi-objective optimizers.

Several interesting directions for future work remain. Any neighborhood criterion (e.g., sharing and clustering [8]) can be introduced in MOEA/D-AD. Although we used the relative distance-based neighborhood decision (Algorithm 2) in this study, investigating the performance of MOEA/D-AD with other neighborhood criteria is one future research topic. Also, the effectiveness of MOEA/D-AD could be improved by using an external archive that stores diverse



solutions [12]. A more efficient search can be performed by utilizing a decision-maker's preference [5]. Incorporating the decision-maker's preference into the search process of MOEA/D-AD is an avenue for future work. Since the existing multi-modal multi-objective test problems are not scalable in the number of objectives and variables, this paper dealt with only two-objective problems with up to five variables. Designing scalable test problems is another research topic.

**Acknowledgments.** This work was supported by the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284).

## References

1. Deb, K.: *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley, Hoboken (2001)
2. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE TEVC* **6**(2), 182–197 (2002)
3. Deb, K., Tiwari, S.: Omni-optimizer: a generic evolutionary algorithm for single and multi-objective optimization. *EJOR* **185**(3), 1062–1087 (2008)
4. Durillo, J.J., Nebro, A.J.: jMetal: a Java framework for multi-objective optimization. *Adv. Eng. Softw.* **42**(10), 760–771 (2011)
5. Gong, M., Liu, F., Zhang, W., Jiao, L., Zhang, Q.: Interactive MOEA/D for multi-objective decision making. In: *GECCO*, pp. 721–728 (2011)
6. Kramer, O., Danielsiek, H.: DBSCAN-based multi-objective niching to approximate equivalent pareto-subsets. In: *GECCO*, pp. 503–510 (2010)
7. Kudo, F., Yoshikawa, T., Furuhashi, T.: A study on analysis of design variables in Pareto solutions for conceptual design optimization problem of hybrid rocket engine. In: *IEEE CEC*, pp. 2558–2562 (2011)
8. Li, X., Epitropakis, M.G., Deb, K., Engelbrecht, A.P.: Seeking multiple solutions: an updated survey on niching methods and their applications. *IEEE TEVC* **21**(4), 518–538 (2017)
9. Liang, J.J., Yue, C.T., Qu, B.Y.: Multimodal multi-objective optimization: a preliminary study. In: *IEEE CEC*, pp. 2454–2461 (2016)
10. Preuss, M., Naujoks, B., Rudolph, G.: Pareto set and EMOA behavior for simple multimodal multiobjective functions. In: Runarsson, T.P., Beyer, H.-G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) *PPSN 2006*. LNCS, vol. 4193, pp. 513–522. Springer, Heidelberg (2006). [https://doi.org/10.1007/11844297\\_52](https://doi.org/10.1007/11844297_52)
11. Rudolph, G., Naujoks, B., Preuss, M.: Capabilities of EMOA to detect and preserve equivalent pareto subsets. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) *EMO 2007*. LNCS, vol. 4403, pp. 36–50. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-70928-2\\_7](https://doi.org/10.1007/978-3-540-70928-2_7)
12. Schütze, O., Vasile, M., Coello, C.A.C.: Computing the set of epsilon-efficient solutions in multiobjective space mission design. *JACIC* **8**(3), 53–70 (2011)
13. Shir, O.M., Preuss, M., Naujoks, B., Emmerich, M.: Enhancing decision space diversity in evolutionary multiobjective algorithms. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) *EMO 2009*. LNCS, vol. 5467, pp. 95–109. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-01020-0\\_12](https://doi.org/10.1007/978-3-642-01020-0_12)
14. Trivedi, A., Srinivasan, D., Sanyal, K., Ghosh, A.: A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE TEVC* **21**(3), 440–462 (2017)

15. Ulrich, T., Bader, J., Zitzler, E.: Integrating decision space diversity into hypervolume-based multiobjective search. In: GECCO, pp. 455–462 (2010)
16. Yuan, Y., Xu, H., Wang, B., Zhang, B., Yao, X.: Balancing convergence and diversity in decomposition-based many-objective optimizers. *IEEE TEVC* **20**(2), 180–198 (2016)
17. Yue, C., Qu, B., Liang, J.: A multi-objective particle swarm optimizer using ring topology for solving multimodal multi-objective problems. *IEEE TEVC* (2017, in press)
18. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE TEVC* **11**(6), 712–731 (2007)
19. Zhou, A., Zhang, Q., Jin, Y.: Approximating the set of pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. *IEEE TEVC* **13**(5), 1167–1189 (2009)
20. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. *IEEE TEVC* **7**(2), 117–132 (2003)



# A Double-Niched Evolutionary Algorithm and Its Behavior on Polygon-Based Problems

Yiping Liu<sup>1</sup>, Hisao Ishibuchi<sup>2</sup>, Yusuke Nojima<sup>1</sup>(✉), Naoki Masuyama<sup>1</sup>,  
and Ke Shang<sup>2</sup>

<sup>1</sup> Department of Computer Science and Intelligent Systems,  
Graduate School of Engineering, Osaka Prefecture University, Sakai,  
Osaka 599-8531, Japan

yiping0liu@gmail.com, {nojima,masuyama}@cs.osakafu-u.ac.jp

<sup>2</sup> Department of Computer Science and Engineering,  
Southern University of Science and Technology,  
Shenzhen 518055, Guangdong, China

hisao@sustc.edu.cn, kshang@foxmail.com

**Abstract.** Multi-modal multi-objective optimization problems are commonly seen in real-world applications. However, most existing researches focus on solving multi-objective optimization problems without multi-modal property or multi-modal optimization problems with single objective. In this paper, we propose a double-niched evolutionary algorithm for multi-modal multi-objective optimization. The proposed algorithm employs a niche sharing method to diversify the solution set in both the objective and decision spaces. We examine the behaviors of the proposed algorithm and its two variants as well as three other existing evolutionary optimizers on three types of polygon-based problems. Our experimental results suggest that the proposed algorithm is able to find multiple Pareto optimal solution sets in the decision space, even if the diversity requirements in the objective and decision spaces are inconsistent or there exist local optimal areas in the decision space.

**Keywords:** Evolutionary computation  
Multi-objective optimization · Multi-modal optimization · Niche  
Diversity

## 1 Introduction

There are many multi-objective optimization problems in real-world applications. Due to the conflicting nature of objectives, there is typically no single optimal solution to these problems, rather a Pareto optimal solution set. The image of the Pareto optimal solution set in the objective space is referred to as the Pareto front. The general task (in *a posteriori* situations) of a multi-objective optimizer is to find an approximate solution set not only close to but also well distributed on the Pareto front.

In view of this, a large number of multi-objective evolutionary algorithms (MOEAs) are designed to solve multi-objective optimization problems over the past two decades. The most typical MOEAs are the Pareto-based ones, in which the Pareto dominance relationship is adopted as the first selection criterion to distinguish well converged solutions, while a density-based second selection criterion is used to promote diversity in the objective space. The widely adopted density-based selection criteria are the crowding distance [1] and niche sharing [4] methods, to name a few.

On the other hand, the objective(s) of an optimization problem may have multi-modal property. For such an objective, there exist different optimal solutions which have the same objective value. This requires evolutionary algorithms to maintain diversity among solutions in the decision space to provide more options for the decision maker. Most existing researches focus on multi-modal single-objective optimization, where niche techniques, e.g., the fitness sharing [3] and crowding [12] methods, are usually employed to diversify the solution set.

Up to now, there are only a few researches on multi-modal multi-objective evolutionary optimization. How to maintain diversity in both the objective and decision spaces is a crucial issue for evolutionary algorithms to solve multi-modal multi-objective optimization problems. In this paper, we propose a Double-Niched Evolutionary Algorithm (DNEA), in which the niche sharing method is adopted in both the objective and decision spaces. We compared the proposed DNEA with three state-of-the-art designs on polygon-based problems, where the performance of the achieved solution sets in the objective space can be visually examined in the decision space. Besides a basic type of the polygon-based problems, we also adopted two other types to further investigate and discuss the behaviors of the competing algorithms on multi-modal multi-objective optimization.

The remainder of this paper is organized as follows. In Sect. 2, the related works on multi-modal multi-objective optimization problems and techniques for diversity maintenance are reviewed for the completeness of the presentation. The proposed DNEA is then described in detail in Sect. 3. Section 4 presents the experimental results and relevant discussions. Section 5 concludes the paper and provides future research directions.

## 2 Related Works

### 2.1 Multi-modal Multi-objective Optimization Problems

As defined in [7] recently, a multi-modal multi-objective optimization problem has more than one Pareto optimal solution sets. In other word, there are at least two similar feasible regions in the decision space corresponding to the same region of the objective space. Later, [13] gave a simple real-world example in the path-planning problem. The traveling time and the number of transfer stations are two objectives in this example. There may exist two different paths that have the same objective values. In such a situation, if an optimizer can provide both

of the paths, the decision maker will have more options for other considerations (e.g. gas station).

Actually, before the concept of multi-modal multi-objective optimization problems is proposed, there have been some researches on this topic. For instance, a map-based problem is proposed in [5], where the goal is find a location nearest to elementary school, junior-high school, convenience store, and railway station on a real-world map. Clearly, it is a four-objective optimization problem. Since the numbers of the aforementioned places are usually more than one on the map, there may exist several optimal locations that have the same objective values. In addition, a few real-world multi-objective optimization problems are also identified to multi-modal property in the literature [11].

In this study, we adopt the polygon-based problems [5] as test problems in the experiments. The polygon-based problems can be termed as an ideal version of the aforementioned map-based problems. The Pareto optimal sets of these problems are located in several regular polygons, which is relatively easy for investigating the behavior of an optimizer at the early stage of the research on multi-modal multi-objective optimization. Moreover, there have not been a widely accepted metric to simultaneously measure the convergence and diversity performances in both the objective and decision spaces of a solution set for multi-modal multi-objective optimization, whereas these performances in the polygon-based problems can be visually examined in a two-dimensional space. This is another important reason of adopting the polygon-based problems in this study.

## 2.2 Diversity Maintenance in the Objective and Decision Spaces

In early 70s and 80s, some classic niche techniques, e.g., the fitness sharing [3] and crowding [12] methods, have been proposed to manipulate the distribution of solutions in the decision space for multi-modal evolutionary optimization. In the fitness sharing method, individuals in the same neighborhood will degrade the fitness of each other, thereby discouraging the others occupying the same niche. In crowding methods, an offspring and its close parents compete with each other, and individuals with better fitness in the sparse areas are favored. There are a lot of other niche methods developed in the last two decades, e.g., clearing [10] and speciation [6]. However, all of above methods can only deal with single-objective optimization problems.

On the other hand, MOEA are developed to provide a diverse solution set in the objective space for multi-objective optimization. Non-dominated Sorting Genetic Algorithm II (NSGA-II) [1] is one of the most representative Pareto-based MOEAs. In NSGA-II, solutions with large crowding distances in the objective space are preferred in the environmental selection. Niche Pareto Genetic Algorithm (NPGA) [4] is another classic Pareto-based MOEA, where the fitness sharing method [3] is termed as the niche sharing method to promote diversity in the objective space. MOEA Based on Decomposition (MOEA/D) [14] is also found a promising alternative to solve multi-objective optimization problems. In MOEA/D, a number of scalarizing functions based on a set of well distributed

reference vectors are used to guide the evolution. The diversity of solutions is ensured by the distribution of the reference vectors. In addition, indicator-based MOEAs [8, 15] and reference points-based MOEAs [9] are theoretically well-supported options.

There have been a few works on maintaining diversity in the decision space for multi-objective optimization. In [2], the Omni-optimizer was proposed by applying the crowding distance in the decision space. A decision space-based niching NSGA-II (DN-NSGA-II) in [7] was developed to search multiple Pareto optimal solution sets, which is similar to omni-optimizer. Very recently, a multi-objective particle swarm optimization algorithm with ring topology and special crowding distance [13] is proposed to obtain good distributions among the population.

In this paper, we propose a double-niched evolutionary algorithm for multimodal multi-objective optimization. In the proposed algorithm, the niche sharing method is simultaneously employed for diversity maintenance in both the objective and decision spaces. We describe the proposed algorithm in detail in the next section.

### 3 A Double-Niched Evolutionary Algorithm

The general framework of DNEA is similar to other generational evolutionary algorithms. What makes DNEA special is its environmental selection operator, which is detailed in Algorithm 1.

---

#### Algorithm 1. Environmental Selection of DNEA

---

**Require:**  $N$  (population size),  $Q$  (candidate solution set),  $\sigma_{\text{obj}}$  (niche radius in the objective space),  $\sigma_{\text{var}}$  (niche radius in the decision space)

- 1:  $F = F_1 \cup F_2 \cup \dots \cup F_k = \text{Nondominated\_sort}(Q)$
- 2:  $P = F_1 \cup F_2 \cup \dots \cup F_{k-1}$
- 3:  $N' = N - |P|$
- 4: **while**  $|F_k| > N'$  **do**
- 5:     **for all**  $\mathbf{x}_i \in F_k$  **do**
- 6:         calculate  $f_{DS}(\mathbf{x}_i)$  according to  $\sigma_{\text{obj}}$  and  $\sigma_{\text{var}}$
- 7:     **end for**
- 8:      $\mathbf{x}_{\text{max}} = \arg \max_{\mathbf{x}_i \in F_k} f_{DS}(\mathbf{x}_i)$
- 9:      $F_k = F_k / \{\mathbf{x}_{\text{max}}\}$
- 10: **end while**
- 11:  $P = P \cup F_k$
- 12: **return**  $P$

---

In Algorithm 1, the solutions in the candidate solution set,  $Q$ , are first sorted to form several nondominated fronts,  $F_1 \cup F_2 \cup \dots \cup F_k$ , where  $k$  in  $F_k$  is the minimal value such that  $|F_1| + |F_2| + \dots + |F_k| > N$  ( $N$  is the population size) (Line 1). This procedure is similar to that in NSGA-II [1]. Then, the first  $F_{k-1}$  nondominated fronts are combined into the new population,  $P$  (Line 2).  $N' = N - |P|$  is the

number of solutions remain to be chosen into  $P$  (Line 3). While  $|F_k| > N'$ , the double-sharing function,  $f_{DS}$ , of each solution in  $F_k$  is calculated as follows (Line 6):

$$f_{DS}(\mathbf{x}_i) = \sum_{\mathbf{x}_j \in F_k} Sh_{obj}(i, j) + Sh_{var}(i, j) \quad (1)$$

In this formulation,  $Sh_{obj}(i, j) = \max\{0, 1 - d_{obj}(i, j)/\sigma_{obj}\}$  and  $Sh_{var}(i, j) = \max\{0, 1 - d_{var}(i, j)/\sigma_{var}\}$ , where  $d_{obj}(i, j)$  and  $\sigma_{obj}$  are the Euclidean distance between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  and the niche radius in the objective space, respectively, and  $d_{var}(i, j)$  and  $\sigma_{var}$  have the similar meanings in the decision space. Then, the solution with the maximum value of the double-sharing function,  $\mathbf{x}_{max}$ , is deleted from  $F_k$  (Line 9). Finally, the remaining solutions in  $F_k$  (where  $|F_k| = N'$ ) are merged into  $P$  (Line 11).

Note that the settings of  $\sigma_{obj}$  and  $\sigma_{var}$  are non-trivial. Generally, the higher dimension of the objective (decision) space and the smaller population size, the larger value of  $\sigma_{obj}$  ( $\sigma_{var}$ ). If  $\sigma_{obj}$  ( $\sigma_{var}$ ) is too large (e.g. larger than the distance between any pair of solutions), boundary solutions are more likely to be selected. Conversely, if  $\sigma_{obj}$  ( $\sigma_{var}$ ) is too small (e.g. smaller than the distance between any pair of solutions), then the solutions to be discarded are selected at random as the double-sharing function would assign zero to every solution. In both of the above situations, the algorithm would encounter diversity maintenance issues. In this study, since it is easy to choose the above values for polygon-based problems, we handle them as pre-specified fixed parameters. Developing a method to adaptively tune  $\sigma_{obj}$  and  $\sigma_{var}$  is an interesting future work.

It can be seen from Algorithm 1 and Eq. (1) that solutions located in sparse regions either in the objective space or in the decision space are preferred. A solution that is very close to others in the objective (decision) space but far away from others in the decision (objective) space still has a chance to be selected. This means that DNEA has a great potential to maintain diversity in both the objective and decision spaces.

In the following section, we investigate the performance of DNEA on the polygon-based problems to demonstrate its effectiveness. We also test two variants of DNEA as competing algorithms. The first is termed as DNEA<sub>obj</sub>, where any  $Sh_{var}$  is set to zero. This means that DNEA<sub>obj</sub> only has the ability to maintain diversity in the objective space. In this situation, DNEA<sub>obj</sub> is almost equal to NPGA. Conversely, setting  $Sh_{obj}$  to zero, the second is termed as DNEA<sub>var</sub>, which only focuses on diversity in the decision space.

## 4 Experiments

In this section, three types of polygon-based problems are first introduced. Then, the competing algorithms and the parameter settings are given. Finally, the performance of the competing algorithms are empirically evaluated and discussed.

#### 4.1 Polygon-Based Problems

We adopt three types of polygon-based problems with 3 and 4 objectives in the experiments. There are four polygons in each problem. The details of them are described as follows.

**Type I:** The first type is a very basic one, where all the polygons have the same shape and size. The vertexes of triangles in the 3-objective problem of Type I are

$$\begin{aligned} \{A_1 = (20, 30), B_1 = (30, 10), C_1 = (10, 10), \\ A_2 = (80, 30), B_2 = (90, 10), C_2 = (70, 10), \\ A_3 = (80, 90), B_3 = (90, 70), C_3 = (70, 70), \\ A_4 = (20, 90), B_4 = (30, 70), C_4 = (10, 70)\}. \end{aligned}$$

$A_i B_i C_i, i = 1, 2, 3, 4$  is the  $i$ th triangle. The three objectives to be minimized are formulated as follows:

$$\begin{aligned} f_1(\mathbf{x}) &= \min\{d(\mathbf{x}, A_i), i = 1, 2, 3, 4\} \\ f_2(\mathbf{x}) &= \min\{d(\mathbf{x}, B_i), i = 1, 2, 3, 4\} \\ f_3(\mathbf{x}) &= \min\{d(\mathbf{x}, C_i), i = 1, 2, 3, 4\} \end{aligned} \quad (2)$$

where  $d(\mathbf{x}, X)$  is the Euclidean distance from a solution  $\mathbf{x}$  to  $X$  ( $X$  is a vertex) in the decision space. Similarly, the objectives of the 4-objective problem of Type I can be defined. There are four rectangles with size of  $20 \times 20$  in the 4-objective problem. Each polygon in these problems is a Pareto optimal region, and all the regions are mapped to the same Pareto front. Finding a uniformly distributed solution set in a polygon will lead to a well distributed approximate Pareto front.

**Type II:** The vertexes of polygons in Type II are the same as those in Type I. The difference is that  $d(\mathbf{x}, X)$  is transformed into  $d(\mathbf{x}, X)^{0.01}$  in the objectives in Type II. By such transformation, uniformly distributed solutions in the objective space are actually nonuniformly distributed in the decision space, and vice versa. By using the problems in Type II, we intend to investigate the behavior of each competing algorithm when the diversities in the objective and decision spaces are inconsistent.

**Type III:** For the problems in Type III, the size of polygons sequentially increases. To be specific, the vertexes of triangles in the 3-objective problem in Type III are

$$\begin{aligned} \{A_1 = (20, 30), B_1 = (30, 10), C_1 = (10, 10), \\ A_2 = (80, 30.02), B_2 = (90.01, 10), C_2 = (69.99, 10), \\ A_3 = (80, 90.2), B_3 = (90.1, 70), C_3 = (69.9, 70), \\ A_4 = (20, 92), B_4 = (31, 70), C_4 = (9, 70)\}. \end{aligned}$$

The vertexes in the 4-objective problem are

$$\begin{aligned} \{A_1 = (10, 30), B_1 = (30, 30), C_1 = (30, 10), D_1 = (10, 10), \\ A_2 = (69.99, 30.01), B_2 = (90.01, 30.01), C_2 = (90.01, 9.99), D_2 = (69.99, 9.99), \\ A_3 = (69.9, 90.1), B_3 = (90.1, 90.1), C_3 = (90.1, 69.9), D_3 = (69.9, 69.9), \\ A_4 = (9, 91), B_4 = (31, 91), C_4 = (31, 69), D_4 = (9, 69)\}. \end{aligned}$$



For the problems in Type III, only the first polygon is the true Pareto optimal region and all the other polygons are local optimal regions. This means that any solution located in the other polygons is dominated by a solution in the first polygon. By testing each competing algorithm on the problems in Type III, we expect to observe that whether the algorithm is trapped into the local optimal regions while maintaining diversity in the decision space.

## 4.2 Competing Algorithms and Parameter Settings

Besides the proposed DNEA and its two variants,  $DNEA_{obj}$  and  $DNEA_{var}$ , we applied three other algorithms, i.e., DN-NSGA-II, NSGA-II, and MOEA/D, to each test problem 30 times using the following specifications:

- Population size: 210 and 220 for 3- and 4-objective problems, respectively
- Population initialization: random values in  $[0, 100]$  for each decision variable
- Termination condition: 300 generations
- Crossover probability: 1.0 (SBX with  $\eta_c = 20$ )
- Mutation probability: 0.5 (Polynomial mutation with  $m = 20$ )
- Niche radius in DNEA and its variants:  $\sigma_{obj} = 0.06$  and  $\sigma_{var} = 0.02$
- Neighborhood size in MOEA/D: 10% of the population size
- Crowding factor in DN-NSGA-II: half of the population size

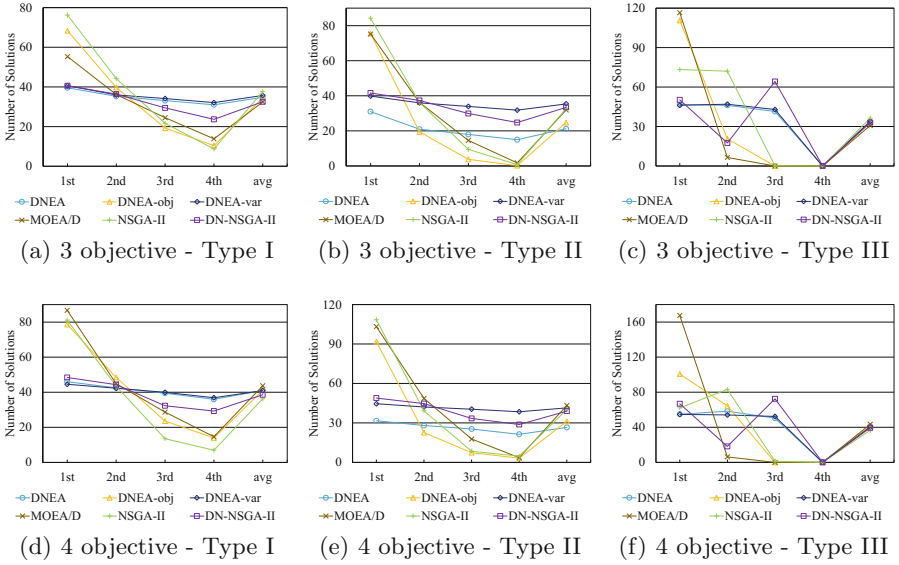
It is interesting to note that the competing algorithms can be classified into three categories. The first one is DNEA, which is designed to maintain diversity in both the objective and decision spaces. The second one includes  $DNEA_{obj}$  and the classic multi-objective optimizers, i.e., NSGA-II and MOEA/D. They only focus on diversity maintenance in the objective space. On the contrary,  $DNEA_{var}$  and DN-NSGA-II fall into the third one.

## 4.3 Results and Discussions

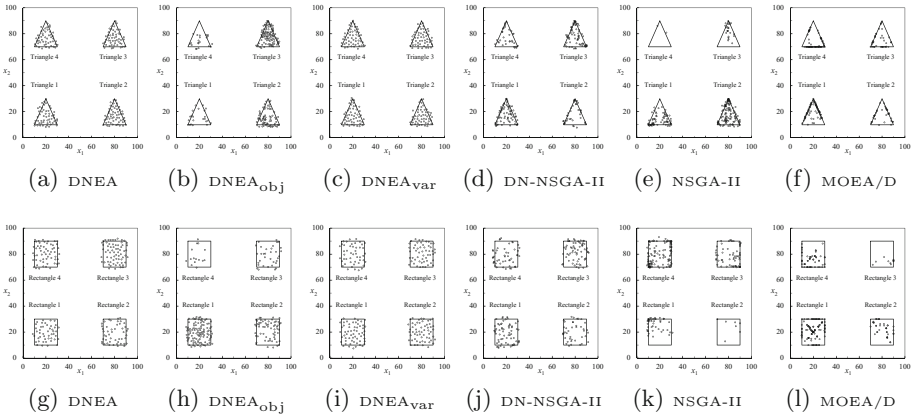
In this part, the performances of the competing algorithms are evaluated and discussed on the three types of polygon-based problems.

**Results on Type I:** In Fig. 1, we show the average number of solutions in the Pareto optimal regions achieved by each competing algorithms over 30 runs. In Fig. 1(a), (b), (d) and (e), ‘1st’ represents the average number of solutions in the polygon which contains the most solutions in each run. ‘2nd’ represents that in the polygon which contains the second most solutions, and ‘3rd’ and ‘4th’ have the similar meanings. ‘avg’ indicates the average number of solutions in all the four polygons. Figure 2 shows the final solution sets of each algorithm in a typical run in the decision space. In the typical run, the number of solutions in each polygon is the nearest to the average number over 30 runs. Note that the results in most other runs are similar to the typical one.

From Fig. 1(a) and (d), we can see that the difference between ‘1st’ and ‘4th’ obtained by  $DNEA_{obj}$ , NSGA-II, and MOEA/D is larger than the others, which means that most of the solutions achieved by these algorithms concentrate on one



**Fig. 1.** The average number of solutions in each polygon.

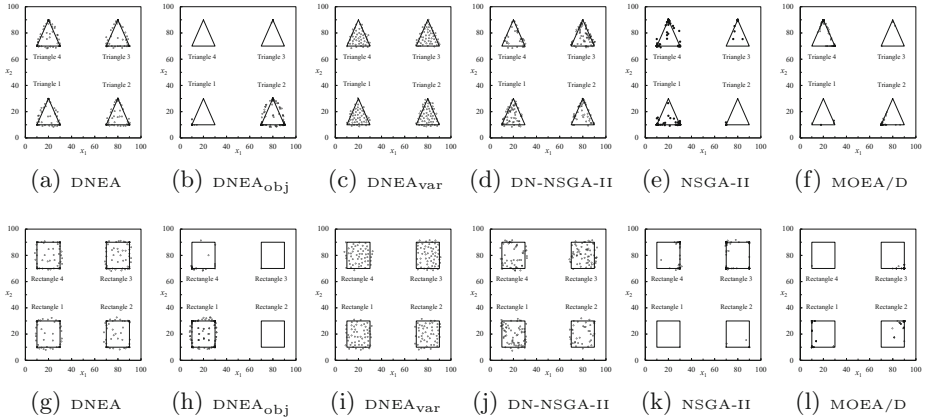


**Fig. 2.** The final solution sets in the decision space on the polygon-based problem in type I (a-f and g-l show the results on the 3- and 4-objective problems, respectively).

or two polygons. This can be also visually observed from the distribution of solutions in the decision space in Fig. 2. Thus,  $DNEA_{obj}$ , NSGA-II, and MOEA/D fail to get multiple Pareto optimal solution sets. On the other hand, the difference between ‘1st’ and ‘4th’ obtained by DNEA,  $DNEA_{var}$ , and DN-NSGA-II in Fig. 1 are relatively small. This suggests that the solutions are almost equally assigned to each polygon, which can be also observed in Fig. 2. From these observations, we can conclude that DNEA,  $DNEA_{var}$ , and DN-NSGA-II have a good

ability to maintain diversity in the decision space. It is worth noting that the difference between “1st” and “4th” of DN-NSGA-II is a bit larger than DNEA and DNEA<sub>var</sub> in Fig. 1(a) and (d), and the distribution of solutions of DN-NSGA-II is not as good as those of DNEA and DNEA<sub>var</sub> in Fig. 2. This indicates that the niche sharing method could perform better than the crowding distance method in maintaining diversity.

**Results on Type II:** Similar to Figs. 2 and 3 shows the results of each competing algorithm on the polygon-based problems in Type II.

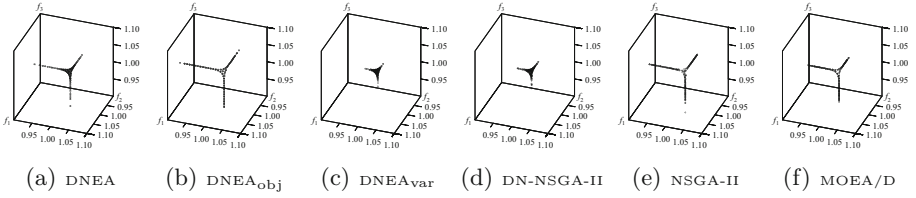


**Fig. 3.** The final solution sets in the decision space on the polygon-based problem in type II (a-f and g-l show the results on the 3- and 4-objective problems, respectively).

The results in Fig. 1(b) and (e) are similar to those in Fig. 1(a) and (d), however, the average numbers of solutions in the polygons achieved by DNEA and DNEA<sub>obj</sub> are smaller than the others. We speculate that the reason is the deterioration of the convergence ability for the complicated Pareto fronts by the enhancement of the diversification ability in those algorithms. From Fig. 3, we can see that only DNEA find all vertexes of all polygons. The solutions achieved by DNEA<sub>obj</sub>, NSGA-II, and MOEA/D only concentrate on several vertexes due to the same reason when handling with the problems in Type I. The behaviors of DNEA<sub>var</sub> and DN-NSGA-II are much the same as those in Fig. 2, since they only consider diversity in the decision space.

For further investigation, we show the non-dominated solutions in the objective space obtained by each algorithm on the 3-objective problem in the typical run in Fig. 4. It can be seen from Fig. 4 that the solutions obtained by DNEA<sub>var</sub> and DN-NSGA-II focus on small areas. This observation suggests that they cannot maintain a good diversity in the objective space for the problems in Type II, although the distribution of their solutions looks uniform in the decision space in Fig. 3. The solutions obtained by DNEA, DNEA<sub>obj</sub>, NSGA-II, and MOEA/D are widely spread in the objective space. However, as we have observed in Fig. 3,

only DNEA can achieve solution sets with large diversity in the decision space. Similar results can be also observed on the 4-objective problem, where they are not presented due to space limits.



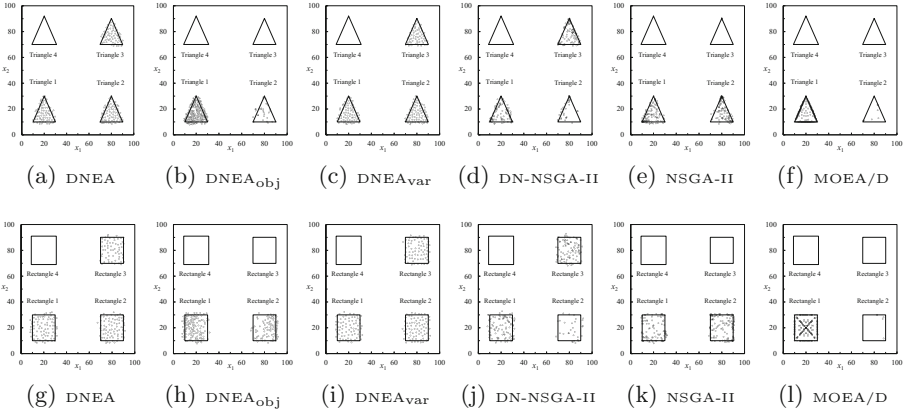
**Fig. 4.** The Pareto fronts on the 3-objective polygon-based problem in type II shown by the 3D coordinates.

From the above-mentioned observations, we can conclude that maintaining diversity in both the objective and decision spaces is necessary for solving the problems in Type II. This motivates us to think that when the requirements of diversity in the objective and decision spaces are conflict, should we consider them equally, or make a trade-off between them? The proposed DNEA in this study belongs to the former way. Developing methods in the latter way will be an interesting future work.

**Results on Type III:** In the same manner as in the previous two subsections, the results on the polygon-based problems in Type III are shown in Figs. 1(c) and (f) and 5. The meaning of the results in Fig. 1(c) and (f) is a little different from those in Figs. 1(a), (b), (d), and (e). In Fig. 1(c) and (f), ‘1st’, ‘2nd’, ‘3rd’, and ‘4th’ indicate the first, second, third, and fourth polygon, respectively (only the first polygon is the true Pareto optimal solution set). Since the polygons in the Type III problems have different sizes, it is better to count the solutions in each polygon separately.

It can be seen from Figs. 1(c) and (f) and 5 that most of the solutions achieved by  $DNEA_{objj}$ , NSGA-II, and MOEA/D locate in the first and second polygons. Especially, almost all of the solutions achieved by MOEA/D are in the first polygon. The reason is that the scalarizing function employed in MOEA/D provides a much larger selection pressure towards the Pareto front than the Pareto dominance criterion used in the other algorithms. The behaviors of DNEA and  $DNEA_{var}$  are nearly the same, where the solutions are equally assigned to the first three polygons. The solutions achieved by DN-NSGA-II also locate in the first three polygons, however, the number of solutions in the second polygon is smaller than those in the first and third polygons for unknown reason.

These observations indicate that maintaining diversity in the decision space can lead to more solutions in the local optimal areas than that in the objective space. However, such algorithms like DNEA,  $DNEA_{var}$ , and DN-NSGA-II are not trapped in these local optimal areas. They can also provide a well-distributed Pareto optimal solution set in the first polygon (i.e., the true Pareto optimal solution set). The question is that whether the solutions in the local optimal areas are necessary in a real-world application. If such solutions are actually



**Fig. 5.** The final solution sets in the decision space on the polygon-based problem in type III (a-f and g-l show the results on the 3- and 4-objective problems, respectively).

needed for the decision maker, how to achieve them is another question. For example, the solutions in the fourth polygon may be needed in some situations, however, none of the algorithms can achieve them. Controlling the number of solutions in each local optimal region is another interesting future work.

### 5 Conclusions

In this paper, we proposed a double-niched evolutionary algorithm, i.e., DNEA, for multi-modal multi-objective optimization. In DNEA, a double sharing function is employed to estimate the density of a solution in both the objective and decision spaces. We introduced three types of polygon-based problems and applied DNEA, its variants, DN-NSGA-II, NSGA-II, and MOEA/D to them. In computational experiments, we have the following observations: (1) Diversity maintenance in the decision space is necessary to find multiple Pareto optimal solution sets. (2) Diversities in the objective and decision spaces should be simultaneously considered if they are inconsistent. (3) Promoting diversity in the decision space leads to more solutions in local Pareto optimal regions. Besides the future works mentioned in Subsection 4.3, balance between convergence and diversity in the decision space is certainly interesting for our future research.

**Acknowledgments.** This work was supported by the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284).

### References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
2. Deb, K., Tiwari, S.: Omni-optimizer: a generic evolutionary algorithm for single and multi-objective optimization. *Eur. J. Oper. Res.* **185**(3), 1062–1087 (2008)

3. Goldberg, D.E., Richardson, J., et al.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms Genetic algorithms and their applications, pp. 41–49. Lawrence Erlbaum, Hillsdale (1987)
4. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niche Pareto genetic algorithm for multiobjective optimization. In: Proceedings of IEEE World Congress on Computational Intelligence, pp. 82–87. IEEE (1994)
5. Ishibuchi, H., Akedo, N., Nojima, Y.: A many-objective test problem for visually examining diversity maintenance behavior in a decision space. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary computation, pp. 649–656. ACM (2011)
6. Li, J.P., Balazs, M.E., Parks, G.T., Clarkson, P.J.: A species conserving genetic algorithm for multimodal function optimization. *Evol. Comput.* **10**(3), 207–234 (2002)
7. Liang, J., Yue, C., Qu, B.: Multimodal multi-objective optimization: a preliminary study. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 2454–2461. IEEE (2016)
8. Liu, Y., Gong, D., Sun, J., Jin, Y.: A many-objective evolutionary algorithm using a one-by-one selection strategy. *IEEE Trans. Cybern.* **47**(9), 2689–2702 (2017)
9. Liu, Y., Gong, D., Sun, X., Zhang, Y.: Many-objective evolutionary optimization based on reference points. *Appl. Soft Comput.* **50**(1), 344–355 (2017)
10. Pérowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 798–803. IEEE (1996)
11. Preuss, M., Kausch, C., Bouvy, C., Henrich, F.: Decision space diversity can be essential for solving multiobjective real-world problems. In: Ehrgott, M., Naujoks, B., Stewart, T., Wallenius, J. (eds.) *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, vol. 634, pp. 367–377. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-04045-0\\_31](https://doi.org/10.1007/978-3-642-04045-0_31)
12. Thomsen, R.: Multimodal optimization using crowding-based differential evolution. In: IEEE Congress on Evolutionary Computation, vol. 2, pp. 1382–1389. IEEE (2004)
13. Yue, C., Qu, B., Liang, J.: A multi-objective particle swarm optimizer using ring topology for solving multimodal multi-objective problems. *IEEE Trans. Evol. Comput.* (2017, early access)
14. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
15. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X. (ed.) *Parallel Problem Solving from Nature-PPSN VIII*, vol. 3242, pp. 832–842. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30217-9\\_84](https://doi.org/10.1007/978-3-540-30217-9_84)



# Artificial Decision Maker Driven by PSO: An Approach for Testing Reference Point Based Interactive Methods

Cristóbal Barba-González<sup>1</sup>, Vesa Ojalehto<sup>2</sup>, José García-Nieto<sup>1</sup>(✉),  
Antonio J. Nebro<sup>1,2</sup>, Kaisa Miettinen<sup>2</sup>, and José F. Aldana-Montes<sup>1</sup>

<sup>1</sup> Dep. Lenguajes y Ciencias de la Computación,  
Ada Byron Research Building, University of Málaga, 29071 Málaga, Spain  
{cbarba,jnieto,antonio,jfam}@lcc.uma.es

<sup>2</sup> Faculty of Information Technology, University of Jyväskylä,  
P.O. Box 35, 40014 Agora, Finland  
{vesa.ojalehto,kaisa.miettinen}@jyu.fi

**Abstract.** Over the years, many interactive multiobjective optimization methods based on a reference point have been proposed. With a reference point, the decision maker indicates desirable objective function values to iteratively direct the solution process. However, when analyzing the performance of these methods, a critical issue is how to systematically involve decision makers. A recent approach to this problem is to replace a decision maker with an artificial one to be able to systematically evaluate and compare reference point based interactive methods in controlled experiments. In this study, a new artificial decision maker is proposed, which reuses the dynamics of particle swarm optimization for guiding the generation of consecutive reference points, hence, replacing the decision maker in preference articulation. We use the artificial decision maker to compare interactive methods. We demonstrate the artificial decision maker using the DTLZ benchmark problems with 3, 5 and 7 objectives to compare R-NSGA-II and WASF-GA as interactive methods. The experimental results show that the proposed artificial decision maker is useful and efficient. It offers an intuitive and flexible mechanism to capture the current context when testing interactive methods for decision making.

**Keywords:** Multiobjective optimization · Preference articulation  
Multiple criteria decision making · Particle swarm optimization

## 1 Introduction

Interactive multiobjective optimization methods based on a reference point are very popular techniques [1–3] not only in current research, but also in industry, as they allow decision makers (DMs) to specify information about their preferences in an intuitive manner to direct the operation of the optimization algorithms. As a consequence, the DM is able to learn progressively (at each iteration) about the

set of (approximated) solutions in the Pareto front of a complex problem, hence reducing one's cognitive load [2]. A second advantage of applying interactive multiobjective optimization methods is that they only need to generate those solutions interesting for the DM, i.e., that are in the region of interest.

Nevertheless, a critical issue arises when testing and comparing interactive methods [1,4], since they require the DMs to be involved in the solution process. Therefore, this involvement makes experiments much more costly than testing by computational means. In addition, other human factors take part, such as inconsistency and variability among decisions, learning curve when facing problems, and different times in solution processes.

In order to cope with this deficiency, a useful approach is to use artificial DMs (ADMs) as mechanisms to generate preference information when comparing interactive methods. Because interactive methods utilize different types of preference information [3,5], appropriate ADMs are demanded for each type. Indeed, in comparison with the amount and diversity of existing interactive methods, the number of ADMs is limited [1]. Interactive methods can be divided into non ad hoc and ad hoc methods depending on whether the DM can be replaced by a value function or not, respectively (see, e.g., [1,6]). Reference point based methods belong to the latter group. However, many popular interactive methods are based on reference points [2,3], where the DM represents the region of interest as a vector of desirable objective values.

Recently, in [7] a new ADM has been developed for testing reference point based interactive methods. It is able to adjust reference points based on information about solutions derived so far. The adjustment involves randomness and the amount of noise decreases during the interactive solution process. The overall procedure is based on a pre-defined neighborhood of a most preferred solution.

Following this line of research, a novel ADM is proposed here that reuses the dynamics of particle swarm optimization (PSO) to guide the generation of reference points, hence, replacing the DM in preference articulation. The idea is to derive reference points by particle's movements in the swarm, which evolves in the objective space. The main contributions of the proposed ADM in this paper are as follows:

- It offers an intuitive, bio-inspired and flexible mechanism to capture the current context in interactive solution processes when tackling multiobjective optimization problems. At each iteration of the process, nondominated solutions derived so far can be used in generating the new reference point.
- It avoids dependence on the pre-defined target levels for objectives.
- It allows different parameter settings to enhance diversification/intensification in the generation of new reference points.

The new ADM is tailored for comparing interactive evolutionary reference point based methods. We demonstrate it on the DTLZ benchmark problems with 3, 5 and 7 objectives and two reference point evolutionary methods R-NSGA-II [8] and WASF-GA [9]. Thus, we use them as examples of interactive EMOs (iEMOs). The experimental results show that the proposed ADM is useful and efficient when compared to the previous one.



The rest of this paper is organized as follows. Section 2 contains background concepts and related work. The proposed ADM is described in Sect. 3. Section 4 summarizes experimental results, analysis and discussions. Finally, conclusions and lines of future work are outlined in Sect. 5.

## 2 Background

Evolutionary multiobjective optimization methods have been shown to perform successfully when finding a set of trade-off solution approximations representing a Pareto front to complex multiobjective optimization problems. Nevertheless, a common requirement in real-world problems arises in solution process where not only Pareto front approximations are demanded, but it is desirable to find preferred solutions or regions that reflect human DM's desires or tendencies.

Interactive methods are able to focus on an area of interest in the objective space, in order to find preferred solutions [1]. Examples of ways how a DM can provide preference information are comparisons of small sets of solutions, classification or indicating desired trade-offs [1, 3, 6]. Furthermore, as mentioned in the introduction, an intuitive type of preference articulation in interactive methods is based on reference points [2, 3], which consist of desirable objective function values.

The difficulty arises when trying to evaluate and compare interactive methods based on reference points, since a human DM is required to take part in the solution process to specify reference points. On the other hand, as stated in [4], there exists a strong necessity of creating automatic DMs to facilitate the comparison of different methods.

We consider multiobjective optimization problems of the form

$$\begin{aligned} &\text{minimize} && \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))^T \\ &\text{subject to} && \mathbf{x} = (x_1, \dots, x_n)^T \in S, \end{aligned} \tag{1}$$

where we minimize<sup>1</sup>  $k$  ( $k \geq 2$ ) objective functions  $f_i : S \rightarrow \mathbb{R}$  on the set  $S \subset \mathbb{R}^n$  of feasible solutions (decision vectors). The elements in the objective space  $\mathbb{R}^k$  are the objective (function) values  $\mathbf{z} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))^T$ , usually called *objective vectors*. We denote the set of feasible objective vectors by  $Z = \mathbf{f}(S)$ . The so-called *Pareto optimal set* of solutions to the problem is defined as:

$$E = \left\{ \mathbf{x} \in S : \nexists \mathbf{x}' \in S \mid f_i(\mathbf{x}') \leq f_i(\mathbf{x}), i = 1, \dots, k \text{ and } \mathbf{f}(\mathbf{x}') \neq \mathbf{f}(\mathbf{x}) \right\} \tag{2}$$

and the corresponding objective vectors form a Pareto front.

**Artificial Decision Maker:** In what follows, we refer to the ADM proposed in [7] as the original ADM. It consists of three main components: steady part, current context and preference information. We need the concepts of *ideal* ( $\mathbf{z}^*$ ) and *nadir* ( $\mathbf{z}^{\text{nad}}$ ) objective vectors of the problem to find reference points.

---

<sup>1</sup> Without loss of generality, we use minimization in definitions.

The former is defined as  $\mathbf{z}^* = (z_1^*, \dots, z_k^*)^T$ , where  $z_i^* = \min_{x \in S} f_i(\mathbf{x})$  for  $i = 1, \dots, k$ , whereas the later is defined as  $\mathbf{z}^{\text{nad}} = (z_1^{\text{nad}}, \dots, z_k^{\text{nad}})^T$ , where  $z_i^{\text{nad}} = \max_{x \in E} f_i(x)$  for  $i = 1, \dots, k$ . If these vectors are not known a priori, the ideal objective vectors can be calculated and the nadir estimated [3]. When applying iEMOS, they can e.g. be estimated from the current population. The three main components of the ADM are:

- *Steady part*: This part includes experience and knowledge available at the beginning of the solution process and remains unchanged in the solution process. As an example, the steady part can consist of a region of interest or of target levels specific to objective functions that are desired to be achieved.
- *Current context*: This part includes all the knowledge about the problem which is gained during the solution process by the ADM, for instance, shape of the Pareto front, trade-offs between the objectives, obtainable objective function values (e.g.,  $\mathbf{z}^*$  and  $\mathbf{z}^{\text{nad}}$ ), etc.
- *Preference information*: With this part, the ADM expresses its knowledge during the solution process in order to guide the method towards solutions that are more preferred by the ADM. Preference information is method-specific and in this research we consider reference points  $\mathbf{q} = (q_1, \dots, q_k)^T$ .

### 3 Artificial Decision Maker Driven by PSO

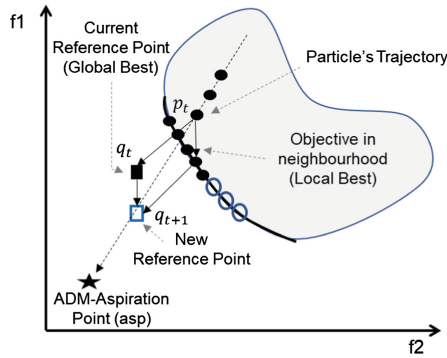
As mentioned before, we propose an ADM that enables testing interactive methods, where preference information is given in the form of a reference point. The proposed ADM utilizes PSO in modifying the current context of the original ADM and we call it ADM-PSO.

Given an iteration counter  $t$ , a *reference point* is denoted by  $\mathbf{q}_t = (q_{t,1}, \dots, q_{t,k})^T$ . It is said to be *achievable* for problem (1), if  $\mathbf{q}_t \in Z + \mathbb{R}_+^k$  (where  $\mathbb{R}_+^k = \{\mathbf{y} \in \mathbb{R}^k \mid y_i \geq 0 \text{ for } i = 1, \dots, k\}$ ), that is, if either  $\mathbf{q}_t \in Z$  or if  $\mathbf{q}_t$  is dominated by a Pareto optimal objective vector in  $Z$ . Otherwise, the reference point is said to be *unachievable*, that is, not all of its components can be achieved simultaneously.

By using a reference point  $\mathbf{q}_t$  in the iteration  $t$ , an ADM is able to feed an interactive multiobjective optimization method with preferences. Then the method can direct the solution process accordingly. If the method is evolutionary, it can generate approximations of Pareto optimal solutions oriented to this specific region of interest. This new set of nondominated solutions can be in turn used to generate a new reference point  $\mathbf{q}_{t+1}$  for the next iteration of the method. This process can be repeated until a stopping criterion is valid. In our case, we use a pre-defined point  $\mathbf{asp}$  to be called ADM-aspiration point and stop once we get a reference point close enough to it. Intuitively, an additional (single objective) optimization problem arises in this process, since the new reference point is to be generated, with a minimum distance to  $\mathbf{asp}$ . In our case, the current Pareto front approximation is used as a population to train the implicit learning model of the optimization method, i.e., the ADM. In this way, the new ADM is able to operate on the objective space by taking advantage of all the information provided by the interactive method.

Keeping this idea in mind, the proposed approach focuses on the use of a canonical PSO to carry out the generation of new reference points, hence acting as an ADM which is able to interact with the underlying interactive multiobjective optimization method. As mentioned earlier, in this study we consider iEMO methods. The aim is to reuse the biological inspiration modeling a particle's dynamics in PSO, to replace DMs when managing their preferences.

A conceptual sketch of this approach is illustrated in Fig. 1, where the new reference point  $\mathbf{q}_{t+1}$  is generated in one movement step of the PSO. It takes into account the previous reference point  $\mathbf{q}_t$  as well as the objective vectors of the nondominated solutions in the current Pareto front approximation, provided by the underlying iEMO.



**Fig. 1.** Conceptual sketch of an ADM-PSO operation. The new reference point  $\mathbf{q}_{t+1}$  is generated by means of PSO particle's movement operators.

Among the many existing PSO variants, for simplicity, ADM-PSO is based on the standard version 2007 [10]. It provides the canonical equations to model the particle's movements, which have been adapted to cope with the reference point generation as follows: Each particle's position vector  $\mathbf{p}$  (codifying an objective vector) is updated at each iteration  $t$  as

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_{t+1}, \tag{3}$$

where  $\mathbf{p}_{t+1}$  is a new candidate reference point ( $\mathbf{p}_{t+1} = \mathbf{q}_{t+1}$ ) and  $\mathbf{v}_{t+1}$  is the velocity vector of the particle given by

$$\mathbf{v}_{t+1} = \omega \cdot \mathbf{v}_t + U^t[0, \varphi_1] \cdot (\mathbf{l}_t - \mathbf{p}_t) + U^t[0, \varphi_2] \cdot (\mathbf{b}_t - \mathbf{p}_t). \tag{4}$$

In (4),  $\mathbf{l}_t$  is the local best position the particle  $\mathbf{p}_t$  has ever stored and  $\mathbf{b}_t$  is the position found by the member of its neighborhood that has had the best performance so far. In ADM-PSO,  $\mathbf{b}_t = \mathbf{q}_t$ , i.e., it is set as the reference point.

Acceleration coefficients  $\varphi_1$  and  $\varphi_2$  control the relative effect of the personal and social best particles and  $U^t$  is a diagonal matrix with elements distributed

in the interval  $[0, \varphi_i]$ , uniformly at random. Finally,  $\omega \in (0, 1)$  is called the inertia weight and influences the trade-off between exploitation and exploration. These parameters can be used to induce additional preference information into the ADM. In particular, ADM-PSO is able to set the *current context* (defined in Sect. 2) by using not only the nearest point to **asp** (as done by the original ADM), but all the points (objective vectors of nondominated solutions) in the Pareto front approximation provided by the iEMO. Consequently, this allows the ADM-PSO to explore thoroughly the Pareto front in the objective space.

In order to assess the adequacy of the new generated reference points, the following single objective fitness function is used by ADM-PSO:

$$d(\mathbf{x}_q) = \sqrt{\sum_{i=1}^k (f_i(\mathbf{x}_q) - asp_i)^2}. \quad (5)$$

In short, the function  $d(\mathbf{x}_q)$  calculates the Euclidean distance between the nearest point (solution  $\mathbf{x}_q$ ) of the Pareto front approximation obtained with the reference point  $\mathbf{q}$ , and the point **asp**, where  $k$  is the number of objectives. As commented before, ADM-PSO aims at minimizing this distance.

### Algorithm of ADM-PSO

For the sake of a better understanding, the pseudo-code of ADM-PSO is shown in Algorithm 1. The first phase corresponds to initialization of parameters, populations and initial Pareto set approximations (from line 1 to 11). In this phase, an initial reference point is also generated (line 12) as done in the original ADM (see Sect. 3 in [7]). After this, the iterative solution process (line 13) starts with multiple rounds of the interactive multiobjective optimization method (line 14) and the corresponding generation of new reference points, by means of PSO (line 16). Each ADM round (lines 13-18) entails a maximum number of iterations ( $I_{max}$ ) in which the iEMO algorithm in question is run until reaching a maximum number of generations  $G_{max}$  (line 14). The PSO is then invoked to obtain a new reference point, which uses the last obtained Pareto set approximation from the previous step. Before that, an intermediate step (line 15) is computed to “accommodate” objective vectors in the Pareto front approximation (or nondominated points) to the swarm ( $S_{t+1}$ ). At the end, the approximation of the region of interest found is returned (line 19) and the whole algorithm ends.

ADM-PSO has been developed in the jMetal library of EMOs and following its architectural style [11] with the aim of taking advantage of all the functionalities provided in this framework: solution types, operators, algorithms, problems, etc. It is worth noting that the core algorithm has been designed to provide a general (software) template, so that iEMOs to be tested can be easily configured. As mentioned, the current configuration contains iEMOs R-NSGA-II and WASF-GA. In this way, a framework for the evaluation and comparison of iEMOs is available<sup>2</sup>.

<sup>2</sup> <https://github.com/KhaosResearch/admpso>.

**Algorithm 1.** Pseudo-code of ADM-PSO

---

```

1:  $I_{max}$  // Maximum number of ADM iterations
2:  $G_{max}$  // Maximum number of iEMO generations
3:  $c, m$  // Genetic operators
4:  $t \leftarrow 0$  // ADM iteration counter
5:  $A$  // Multiobjective optimization problem
6:  $S$  // Maximum swarm size of ADM-PSO
7:  $\varphi_1, \varphi_2, \omega$  // PSO specific parameters
8:  $M$  // iEMO algorithm(s) tested
9:  $P_t \leftarrow initializePopulation(N)$  // where  $N$  is Population size
10:  $evaluate(P_t, A)$ 
11:  $E_t \leftarrow initializeParetoSet(P_t)$ 
12:  $\mathbf{q}_t \leftarrow initializeReferencePoint(\mathbf{asp}, \mathbf{z}^*, \mathbf{z}^{nad}, w_r, p_r)$  // As in original ADM
13: while ( $t < I_{max}$ ) AND ( $\mathbf{asp} \neq \mathbf{q}_t$ ) do
14:   ( $P_{t+1}, E_{t+1}$ )  $\leftarrow compute_{iEMO}(M, \mathbf{q}_t, c, m, P_t, A, G_{max})$  // Evolves iEMO
15:    $S_{t+1} \leftarrow setNewSwarm(S, E_{t+1})$  // Generate new swarm from  $E_{t+1}$ 
16:    $\mathbf{q}_{t+1} \leftarrow compute_{PSO}(\mathbf{asp}, \mathbf{q}_t, S_{t+1}, \varphi_1, \varphi_2, \omega)$  // Generate new reference point
17:    $t \leftarrow t + 1$ 
18: end while
19: return  $E_{t+1}$  // Notify Pareto front approximation

```

---

## 4 Experimental Results

In order to demonstrate the validity of the proposed approach, a series of experiments has been conducted to test two iEMOs called WASF-GA [9] and R-NSGA-II [8]. In the experiments, ADM-PSO generates reference points for the methods, hence enabling automatic tests and comparisons. For these experiments, a common framework has been used that comprises of a family of seven DTLZ benchmark problems [12] with 3, 5 and 7 objectives, summing up to 21 different problems. For each combination of algorithms and problems, 31 independent runs were performed.

In these experiments, a set of fixed ADM-aspiration points (*asp*) was configured for each problem. They are all achievable and calculated by taking into account the estimated ideal and nadir objective vectors for each problem as  $asp_i = 2/3 \times z_i^{nad} + z_i^*$ . It is worth noting that for these problems the ideal objective vectors are always at the origin  $(0, \dots, 0)$ , whereas nadir objective vectors were obtained from the worst solutions (ranges) found so far in preliminary experiments, where algorithmic parameters were tuned as described below. In this regard, Table 1 shows the nadir objective vectors used with the corresponding *asp* for each problem, as well as the number of objective functions ( $k$ ).

In order to enable fair comparisons, WASF-GA and R-NSGA-II were set using a common parameter setting that consists of a population size  $N = 100$ , external archive size  $E = 100$ , a maximum number of (iEMO) generations  $G_{max} = 20,000$ , a crossover SBX with a probability  $c = 0.9$  and a distributional index 20, a polynomial mutation with a probability  $m = 0.1$ , a mutation distributional index 20, and a binary tournament selection. In the case of R-NSGA-II, the epsilon parameter was set to 0.0045.

**Table 1.** Achievable ADM-aspiration points (*asp*) and nadir objective vectors used.

Problem	k	<i>asp</i>	<i>z<sup>nad</sup></i>
DTLZ1	3	(6.7, 26.7, 133.4)	(10.0, 40.0, 200.0)
	5	(7.0, 26.7, 133.7, 33.6, 100.4)	(10.0, 40.0, 200.5, 50.5, 150.5)
	7	(7.0, 26.7, 133.7, 33.6, 100.4, 31.0, 67.7)	(10.0, 40.0, 200.5, 50.5, 150.5, 46.5, 101.5)
DTLZ2	3	(2.6, 1.4, 1.4)	(4.0, 2.0, 2.0)
	5	(2.6, 1.4, 1.4, 1.4, 1.4)	(4.0, 2.0, 2.0, 2.0, 2.0)
	7	(2.6, 1.4, 1.4, 1.4, 1.4, 1.4, 1.4)	(4.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0)
DTLZ3	3	(17.0, 122.0, 38.7)	(25.5, 183.0, 58.0)
	5	(1.0, 19.0, 1.0, 667.0, 668)	(1.5, 28.5, 1.5, 999.0, 999.5)
	7	(17.0, 122.0, 40.0, 40.0, 667.0, 668.0, 667.0)	(25.5, 183.0, 60.0, 60.0, 999.0, 999.5, 999.0)
DTLZ4	3	(1.4, 1.4, 1.4)	(2.0, 2.0, 2.0)
	5	(1.4, 1.4, 1.4, 1.4, 1.4)	(2.0, 2.0, 2.0, 2.0, 2.0)
	7	(1.4, 1.4, 1.4, 1.4, 1.4, 1.4, 1.4)	(2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0)
DTLZ5	3	(1.4, 1.4, 1.4)	(2.0, 2.0, 2.0)
	5	(1.4, 1.4, 3.0, 3.0, 1.7)	(2.0, 2.0, 3.0, 3.0, 2.5)
	7	(1.4, 1.4, 1.4, 1.4, 1.0)	(2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 1.5)
DTLZ6	3	(3.0, 2.7, 4.4)	(4.5, 4.0, 6.5)
	5	(3.0, 2.7, 4.4, 4.4, 5.4)	(4.5, 4.0, 6.5, 6.5, 8.0)
	7	(3.0, 2.7, 4.4, 4.4, 4.4, 3.7, 3.4)	(4.5, 4.0, 6.5, 6.5, 6.5, 5.5, 5.0)
DTLZ7	3	(1.4, 1.4, 13.4)	(2.0, 2.0, 20.0)
	5	(1.4, 1.4, 1.4, 1.4, 21.7)	(2.0, 2.0, 2.0, 2.0, 32.5)
	7	(1.4, 1.4, 1.4, 1.4, 1.4, 1.4, 42.7)	(2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 55.0)

For ADM-PSO, parameters were set by following the previous work and standard settings of PSO 2007 [10]. It comprised of a maximum number of iterations  $I_{max} = 11$ , an objective consideration probability  $p = 0.5$ , a weight  $w = 1/k$  and a tolerance  $\theta = 10^{-3}$  (see [7] for a further explanation of ADM parameters). For PSO, we set  $\varphi_1 = \varphi_2 = 1/(2 + \log(2))$  and inertia  $\omega = 1/(2 \cdot \log(2))$ . Since the swarm is fed with those non-dominated points of the external archive of the iEMO, the maximum swarm size was set accordingly, i.e.,  $S = E = 100$  each time the ADM-PSO is started. It conducted only 3 generations to assure that particles are able to move accordingly with new data.

In addition, with the aim of keeping track of the original ADM [7], it was also applied by following the same procedure. The results of ADM and ADM-PSO are arranged in Tables 2 and 3, respectively. In these tables, WASF-GA and R-NSGA-II are compared using the DTLZ problems, where for brevity the number of objectives is limited to 3, 5 and 7. The mean, standard deviation (STD) and minimum (MIN) distances of the nearest point (in the resulting Pareto front approximation) to the ADM-aspiration point *asp* are reported, together with the number of iterations (ITER) the ADM used on the average.

The first observation can be made from Tables 2 and 3 with regards to ADM and ADM-PSO. They performed in a similar way when guiding the underlying iEMOs (WASF-GA and R-NSGA-II) to find solutions close to the ADM-aspiration point. In this sense, no statistical differences could be found when comparing the mean distance distributions of all the combinations of ADMs with iEMOs. To be more specific, according to Friedman’s test [13] with  $\chi^2$  and 3 degrees of freedom, a value of 6.07 was obtained ( $< 7.81$  from a  $\chi^2$  distribution table  $\alpha = 0.05$ ), so the null hypothesis could not be rejected.

**Table 2.** WASF-GA versus R-NSGA-II with the original ADM.

Pro blem	k	WASF-GA				R-NSGA-II			
		MEAN	STD	MIN	ITER	MEAN	STD	MIN	ITER
DTLZ1	3	3.54E-02	3.33E-02	1.08E-02	7.70E+00	4.53E-01	1.57E-01	2.46E-01	8.40E+00
DTLZ1	5	1.60E+00	4.82E-01	4.83E-01	8.10E+00	2.15E+00	2.48E-01	1.56E+00	8.00E+00
DTLZ1	7	3.15E-01	1.48E-01	8.95E-02	9.00E+00	7.33E-02	2.82E-02	4.16E-02	7.50E+00
DTLZ2	3	1.75E-01	3.84E-01	2.42E-02	9.40E+00	7.11E-02	3.33E-02	1.82E-02	8.40E+00
DTLZ2	5	4.24E-01	3.14E-01	2.23E-01	7.30E+00	3.90E-01	6.22E-02	3.01E-01	7.90E+00
DTLZ2	7	1.68E-01	4.22E-02	1.10E-01	6.60E+00	1.43E-01	9.09E-02	4.07E-02	8.80E+00
DTLZ3	3	4.78E+00	1.23E+00	3.41E+00	7.00E+00	1.29E+01	2.33E+00	1.02E+01	7.00E+00
DTLZ3	5	3.74E+01	4.41E+00	2.91E+01	7.70E+00	2.77E+01	2.17E+01	1.06E+00	9.50E+00
DTLZ3	7	2.23E+02	3.33E+01	1.73E+02	6.60E+00	3.25E+02	2.78E+01	2.81E+02	8.10E+00
DTLZ4	3	6.01E-01	5.16E-04	6.01E-01	9.20E+00	6.06E-01	3.92E-03	6.02E-01	8.00E+00
DTLZ4	5	6.97E-01	1.17E-01	5.49E-01	7.30E+00	3.70E-01	6.40E-02	3.18E-01	6.90E+00
DTLZ4	7	7.03E-01	2.64E-02	6.51E-01	8.10E+00	6.25E-01	2.81E-02	5.80E-01	8.20E+00
DTLZ5	3	1.70E-01	3.16E-04	1.70E-01	9.10E+00	1.62E-01	5.86E-03	1.51E-01	7.90E+00
DTLZ5	5	9.51E-03	5.27E-03	4.92E-03	9.30E+00	1.29E-01	2.69E-02	9.67E-02	7.10E+00
DTLZ5	7	1.51E-01	2.54E-02	1.15E-01	7.70E+00	1.13E-01	3.74E-02	4.85E-02	7.00E+00
DTLZ6	3	4.52E-01	4.72E-01	1.60E-01	7.20E+00	1.48E+00	1.29E-01	1.25E+00	8.10E+00
DTLZ6	5	1.46E+00	1.25E+00	1.52E-01	7.50E+00	2.83E+00	1.37E+00	9.00E-01	5.20E+00
DTLZ6	7	4.87E+00	1.52E-01	4.63E+00	7.70E+00	4.60E+00	1.72E-01	4.20E+00	8.60E+00
DTLZ7	3	2.16E-02	1.96E-02	4.12E-03	9.90E+00	4.85E-01	1.10E-01	2.92E-01	7.60E+00
DTLZ7	5	2.90E+00	6.02E-01	1.88E+00	7.10E+00	2.80E+00	7.16E-01	1.75E+00	8.80E+00
DTLZ7	7	2.05E+01	1.52E+00	1.78E+01	7.10E+00	1.24E+01	9.65E-01	1.06E+01	4.70E+00

Nevertheless, ADM-PSO was able to obtain solutions in a lower number of iterations, which means an advantage in the computational effort. This can be observed in columns ITER of Tables 2 and 3, where ADM-PSO with WASF-GA used a lower number of iterations than the original ADM with WASF-GA in 15 out of 21 problems. Furthermore, ADM-PSO with R-NSGA-II required fewer iterations than the original ADM with R-NSGA-II for all the problems except for DTLZ7 with 7 objectives. Overall, the number of iterations can be used as an indicator for the solution process even though a smaller number does not directly mean a good performance. The ADM may be e.g. tailored to focus first on learning where very different reference points are used for scanning the Pareto front.

In this sense, it is worth noting that the aim of iEMOs is not to obtain a complete coverage of the Pareto front, but to focus on a specific region of interest relevant for a DM. WASF-GA and R-NSGA-II usually generate a set of solutions in that region, so that ADM-PSO uses them when forming the swarm. In this way, it is able to take advantage of information in the current context while experimenting a fast convergence (typical in PSO) to the ADM-aspiration point, i.e., to generate better reference points.

A special case was registered for problem DTLZ3 in Tables 2 and 3 since the performances of the iEMOs involved usually deteriorated as the ADM did not achieve the ADM-aspiration points consistently. Probably, the heterogeneity in the ranges observed when calculating the nadir objective vectors for this problem made the ADMs to generate the points *asp* close to unachievable regions, hence leading the iEMO to require extra effort to reach it.

In general, the performance of the iEMOs got worse as the number of objective functions increased. This is not surprising, since the complexity of DTLZ

**Table 3.** WASF-GA versus R-NSGA-II with ADM-PSO.

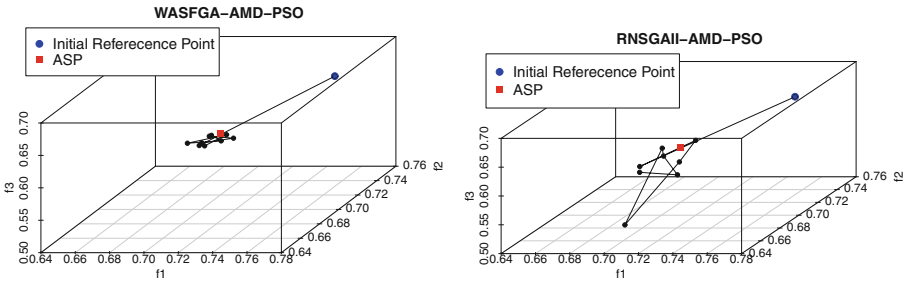
Pro blem	k	WASF-GA				R-NSGA-II			
		MEAN	STD	MIN	ITER	MEAN	STD	MIN	ITER
DTLZ1	3	1.29E-02	5.41E-03	4.69E-03	6.80E+00	2.01E-01	1.34E-01	6.06E-02	5.80E+00
DTLZ1	5	1.73E+00	3.39E-01	1.11E+00	7.90E+00	2.18E+00	2.01E-01	1.95E+00	5.30E+00
DTLZ1	7	1.89E-01	7.71E-02	6.18E-02	7.60E+00	6.70E-02	1.37E-02	4.64E-02	6.10E+00
DTLZ2	3	2.50E-02	5.67E-04	2.44E-02	8.10E+00	5.51E-02	2.62E-02	2.98E-02	6.60E+00
DTLZ2	5	4.65E-01	3.75E-01	1.26E-01	6.00E+00	4.44E-01	1.92E-01	3.62E-01	6.40E+00
DTLZ2	7	1.51E-01	4.08E-02	1.06E-01	7.00E+00	9.11E-02	9.77E-02	2.88E-02	7.00E+00
DTLZ3	3	4.00E+00	1.28E+00	1.70E+00	7.50E+00	1.02E+01	2.35E+00	6.78E+00	6.10E+00
DTLZ3	5	2.59E+01	1.15E+01	1.35E+01	5.90E+00	4.60E+00	2.59E+00	1.88E+00	7.20E+00
DTLZ3	7	1.95E+02	5.60E+01	1.22E+02	7.50E+00	3.30E+02	1.78E+01	2.98E+02	7.80E+00
DTLZ4	3	6.01E-01	0.00E+00	6.01E-01	7.20E+00	6.03E-01	3.07E-03	6.01E-01	4.60E+00
DTLZ4	5	4.39E-01	4.80E-02	3.24E-01	8.30E+00	3.17E-01	9.62E-03	3.03E-01	6.80E+00
DTLZ4	7	6.99E-01	3.37E-02	6.27E-01	7.30E+00	5.87E-01	6.96E-03	5.77E-01	7.30E+00
DTLZ5	3	1.70E-01	2.93E-17	1.70E-01	6.10E+00	1.64E-01	3.84E-03	1.56E-01	7.10E+00
DTLZ5	5	8.96E-03	3.92E-03	2.54E-03	7.00E+00	1.14E-01	2.75E-02	8.81E-02	6.00E+00
DTLZ5	7	1.25E-01	2.48E-02	9.47E-02	5.10E+00	7.54E-02	2.44E-02	4.22E-02	6.20E+00
DTLZ6	3	2.57E-01	6.94E-02	1.54E-01	6.80E+00	1.47E+00	1.16E-01	1.24E+00	7.60E+00
DTLZ6	5	1.07E+00	1.37E+00	1.54E-01	8.30E+00	2.51E+00	1.02E+00	1.29E+00	5.10E+00
DTLZ6	7	4.87E+00	1.23E-01	4.72E+00	7.10E+00	4.56E+00	1.17E-01	4.39E+00	6.50E+00
DTLZ7	3	9.95E-03	2.86E-03	4.28E-03	6.70E+00	3.79E-01	1.05E-01	2.18E-01	6.20E+00
DTLZ7	5	2.34E+00	4.31E-01	1.74E+00	6.10E+00	2.76E+00	6.04E-01	1.99E+00	6.20E+00
DTLZ7	7	2.12E+01	1.81E+00	1.80E+01	8.00E+00	1.01E+01	4.26E+00	2.18E+00	5.40E+00

problems is higher with more objectives, and the number of dimensions tackled by both ADMs is also larger, while the number of evaluations was set similarly for all the problems and numbers of objectives. In this regard, an interesting observation is that R-NSGA-II performed usually better than WASF-GA for 5 and 7 objectives, even when employing a lower number of iterations of ADM/ADM-PSO (ITERS).

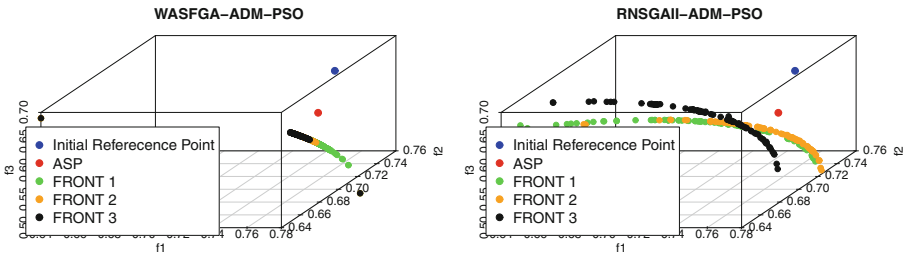
From the point of view of ADM-PSO’s specific performance, it can be seen from Table 3 that it enabled the comparison between WASF-GA and R-NSGA-II and also allowed to capture certain differences in their search strategies. To be more concrete, WASF-GA obtained a better mean (denoted with a grey background in 11 values out of 21 problems) and a minimum distances (11 out of 21 problems) than R-NSGA-II, although the latter needed a slightly lower number of iterations.

To illustrate the behaviour of ADM-PSO with these two iEMOs, Fig. 2 shows representative examples of trajectories walked by the reference points generated, when guiding WASF-GA (left) and R-NSGA-II (right) to solve DTLZ5 with 3 objectives. It can be observed that ADM-PSO with R-NSGA-II resulted with a more spread out trajectory than WASF-GA, whereas the latter algorithm was more concentrated to the area around the ADM-aspiration point (red square symbol). These reference points are in turn generated according to their precedent Pareto front approximations, which are used by ADM-PSO to constitute contextual information. This is indeed illustrated in Fig. 3, where Pareto front approximations are plotted with regards to the 3 closest reference points to the ADM-aspiration point (of ADM-PSO). Accordingly, Pareto front approximations of R-NSGA-II are scattered, whereas WASF-GA showed more concentrated fronts to the ADM-aspiration point area.





**Fig. 2.** Search paths of ADM-PSO when guiding WASF-GA (left) and R-NSGA-II (right) to solve DTLZ5 with 3 objectives. (Color figure online)



**Fig. 3.** Pareto front approximations of WASF-GA (left) and R-NSGA-II (right) according to the reference points of ADM-PSO (the 3 closest reference points to the *asp*), when solving DTLZ5 with 3 objectives.

## 5 Conclusions and Future Work

We have introduced ADM-PSO, a new variant of an ADM for preference articulation in the form of reference points guided by PSO. Our approach enables comparing interactive reference point based EMOs without involving human DMs. ADM-PSO has been implemented following the jMetal architecture and its source codes are freely available.

The proposed approach was demonstrated on the DTLZ benchmark problems with 3, 5 and 7 objectives and using R-NSGA-II and WASF-GA as interactive reference point based methods to be compared. The experimental results show that ADM-PSO is useful and efficient in comparison with the previous ADM. It offers a bio-inspired and flexible mechanism to capture the current context of an ADM in interactive solution processes.

ADM-PSO is conceptually intuitive and straightforward, although it opens a promising line of future research as follows. First, exploring the possibilities of using different metaheuristics like DE, CMA-ES and GA for the generation of reference points instead of PSO. Second, testing parameter tuning in PSO (and other metaheuristics), e.g.,  $\varphi_1$  and  $\varphi_2$ , to control the influence of the current reference point (global best) and/or local history of particles, hence to induce the ADM's behavior in terms of intensification/diversification mechanisms. Third,

carrying out further comparisons of multiple state-of-the-art iEMOs, to test their performances in a controlled and computationally fair execution framework.

**Acknowledgements.** This work was partially funded by Grants TIN2017-86049-R (Spanish MICINN) and P12-TIC-1519 (PAIDI). C. Barba-González was supported by Grant BES-2015-072209 (Spanish MICINN) and University of Jyväskylä. J. García-Nieto is the recipient Post-Doct fellowship of “Plan Propio” at Universidad de Málaga. This work was supported on the part of V. Ojalehto by the Academy of Finland (grant number 287496).

## References

1. Miettinen, K.: *Nonlinear Multiobjective Optimization*. Kluwer, Boston (1999)
2. Miettinen, K., Ruiz, F., Wierzbicki, A.P.: Introduction to multiobjective optimization: interactive approaches. In: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (eds.) *Multiobjective Optimization*. LNCS, vol. 5252, pp. 27–57. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-88908-3\\_2](https://doi.org/10.1007/978-3-540-88908-3_2)
3. Miettinen, K., Hakanen, J., Podkopaev, D.: Interactive nonlinear multiobjective optimization methods. In: Greco, S., Ehrgott, M., Figueira, J. (eds.) *Multiple Criteria Decision Analysis*. ISOR, vol. 233, pp. 931–980. Springer, New York (2016). [https://doi.org/10.1007/978-1-4939-3094-4\\_22](https://doi.org/10.1007/978-1-4939-3094-4_22)
4. López-Ibáñez, M., Knowles, J.: Machine decision makers as a laboratory for interactive EMO. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) *EMO 2015*. LNCS, vol. 9019, pp. 295–309. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-15892-1\\_20](https://doi.org/10.1007/978-3-319-15892-1_20)
5. Purshouse, R.C., Deb, K., Mansor, M.M., Mostaghim, S., Wang, R.: A review of hybrid evolutionary multiple criteria decision making methods. In: *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1147–1154 (2014)
6. Steuer, R.E.: *Multiple Criteria Optimization: Theory, Computation, and Applications*. Wiley, Hoboken (1986)
7. Ojalehto, V., Podkopaev, D., Miettinen, K.: Towards automatic testing of reference point based interactive methods. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) *PPSN 2016*. LNCS, vol. 9921, pp. 483–492. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_45](https://doi.org/10.1007/978-3-319-45823-6_45)
8. Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ACM pp. 635–642 (2006)
9. Ruiz, A.B., Luque, M., Miettinen, K., Saborido, R.: An interactive evolutionary multiobjective optimization method: interactive WASF-GA. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) *EMO 2015*. LNCS, vol. 9019, pp. 249–263. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-15892-1\\_17](https://doi.org/10.1007/978-3-319-15892-1_17)
10. PSO-Central-Group: Standard PSO 2006, 2007, and 2011. Technical report, Particle Swarm Central, January 2011. <http://www.particleswarm.info/>
11. Durillo, J.J., Nebro, A.J.: jMetal: a java framework for multi-objective optimization. *Adv. Eng. Softw.* **42**(10), 760–771 (2011)
12. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: *Proceedings of the 2002 Congress on Evolutionary Computation*. vol. 1, pp. 825–830. IEEE (2002)
13. Sheskin, D.J.: *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, California (2007)



# A Simple Indicator Based Evolutionary Algorithm for Set-Based Minmax Robustness

Yue Zhou-Kangas<sup>(✉)</sup> and Kaisa Miettinen

University of Jyväskylä, Faculty of Information Technology,  
P.O. BOX(35), Agora, 40014 University of Jyväskylä, Finland  
{yue.y.zhou-kangas,kaisa.miettinen}@jyu.fi

**Abstract.** For multiobjective optimization problems with uncertain parameters in the objective functions, different variants of minmax robustness concepts have been defined in the literature. The idea of minmax robustness is to optimize in the worst case such that the solutions have the best objective function values even when the worst case happens. However, the computation of the minmax robust Pareto optimal solutions remains challenging. This paper proposes a simple indicator based evolutionary algorithm for robustness (SIBEA-R) to address this challenge by computing a set of non-dominated set-based minmax robust solutions. In SIBEA-R, we consider the set of objective function values in the worst case of each solution. We propose a set-based non-dominated sorting to compare the objective function values using the definition of lower set less order for set-based dominance. We illustrate the usage of SIBEA-R with two example problems. In addition, utilization of the computed set of solutions with SIBEA-R for decision making is also demonstrated. The SIBEA-R method shows significant promise for finding non-dominated set-based minmax robust solutions.

**Keywords:** Minmax robust Pareto optimal solutions · Hypervolume Set-based dominance · SIBEA · Uncertainty

## 1 Introduction and Background

The need to simultaneously consider multiple objectives and the existence of uncertainty from various sources complicate real-world optimization problems. Uncertainty due to for example imprecise data or uncertain future developments usually reflects as parameters in the objective functions. Traditional multiobjective optimization methods concentrate on optimizing multiple objectives simultaneously and finding a set of Pareto optimal or non-dominated solutions for deterministic formulations of problems. Different approaches can be used to find this set, for example with scalarization techniques (see e.g., [21]) or with evolutionary multiobjective optimization methods (see e.g., [8]). However, the involved uncertainty can affect deterministic Pareto optimal or non-dominated solutions

with undesired degradation in their objective function values. Thus, considering uncertainty in the optimization process is as important as optimizing multiple objectives simultaneously.

The goal of handling uncertainty and multiple objectives simultaneously is finding robust solutions that are sufficiently immune to the uncertainty and with trade-offs among the objectives. Different concepts of robustness and measures of robustness have been proposed in the literature. Typically, robustness measures are incorporated into evolutionary multiobjective optimization methods to quantify the effects of uncertainty on the objective function values (e.g., [4, 9, 12, 17]). Different robustness concepts alter the definition of dominance. Based on the concepts, uncertain multiobjective optimization problems can be transformed to deterministic ones (as summarized in [14, 25]). In addition, different possible values of uncertain parameters can be considered simultaneously during the optimization process (as e.g., in [22, 24]).

Among the robustness concepts, the most widely used ones belong to the family of minmax robustness (e.g., [5, 11, 16]). Due to different possible values of the uncertain parameters, a solution in the decision space can correspond to a set of outcomes (i.e., objective function values). We refer to a set of outcomes corresponding to a solution as the outcome set of the solution. Minmax robustness compares the worst outcomes in the outcome sets and finds the best possible ones. The worst outcomes are referred to as the worst case outcome set.

Set-based minmax robustness [11] finds the solutions with the best worst case outcome sets by utilizing set-based dominance [23]. For feasible solutions considered, we need to identify their worst case outcome sets by maximizing the multiple objectives simultaneously in their outcome sets and compare them with set-based dominance. This series of tasks makes the computation of set-based minmax robust solutions challenging. Methods from robust optimization and mathematical optimization can only address the challenge partially.

Some solution methods via scalarizing and reformulating the scalarized subproblems have been proposed e.g., in [5, 16]. However, typically the reformulations are based on some (strict) assumptions on the characteristics of the problem which cannot be always guaranteed in practical problems. If no assumptions on the characteristics can be made, using samples to replace the uncertainty set has been explored e.g., in [27]. The shortcoming is that the resulting solutions might not be or near to minmax robust. The needs of obtaining a more accurately approximated set of set-based minmax robust solutions have motivated us for further developments.

Different types of evolutionary multiobjective optimization methods have been able to approximate solutions for many challenging problems. For comparing worst case outcome sets, methods which combine non-dominated sorting and crowding distance are not suitable since defining the crowding distance between the worst case outcome sets is not possible. Decomposition based methods cannot be directly applied since we cannot directly associate worst case outcome sets to the weighting vectors. Set-based dominance has been utilized in the evolutionary multiobjective optimization community e.g., in [3, 30]. The population is treated

a whole set and set-based dominance is used to improve the population. Very recently, using set-based dominance to solve problems involving uncertainty has also attracted interest. In [15], a genetic algorithm has been proposed for solving combinatorial bi-objective optimization problems with a set of discrete values of the uncertain parameters. In [13], an evolutionary algorithm has been proposed for solving problems with interval uncertainty (i.e., the uncertain parameters stem from some intervals) with reformulated objective functions. A specific definition of set-based dominance has been used to compare the worst case outcomes in [2]. These earlier research demonstrates potential to address the challenges.

In this paper, we propose utilizing an evolutionary multiobjective optimization approach SIBEA-R to tackle the challenge of approximating set-based minmax robust Pareto optimal solutions. We extend SIBEA [28] for this purpose. We incorporate the definition of set-based minmax robustness into the SIBEA method and develop a non-dominated sorting procedure based on the lower set less order. We also utilize the hypervolume of the worst case outcome sets in the environmental selection process.

The rest of the paper is organized as follows: Sect. 2 presents some concepts we use in this paper. Section 3 presents SIBEA-R followed by some numerical examples of how it can be used in Sect. 4. Finally, Sect. 5 concludes the paper and identifies some future research directions.

## 2 Preliminaries

In this paper, we consider multiobjective optimization problems with uncertainty reflected in the parameters of the objective functions in the following form:

$$\left( \begin{array}{l} \text{minimize } (f(x, \xi) = f_1(x, \xi), \dots, f_k(x, \xi))^T \\ \text{subject to } x \in \mathfrak{X} \end{array} \right)_{\xi \in \mathcal{U}}, \tag{1}$$

where  $x = (x_1, \dots, x_n)^T$  is the decision vector from the feasible set  $\mathfrak{X}$  in the decision space  $\mathbb{R}^n$  whose components are called decision variables and  $\xi$  consists of the uncertain parameters which are assumed to stem from an uncertainty set  $\mathcal{U}$ . With  $\xi$  stemming from  $\mathcal{U}$ , a solution  $x \in \mathfrak{X}$  is mapped in the objective space as a set-valued map [23] under the objective functions  $f_1, \dots, f_k$  to the objective space. We call this set-valued map the outcome set and denote by  $f_{\mathcal{U}}(x) = \{f(x, \xi), \xi \in \mathcal{U}\}$ . In the outcome set, a specific objective vector  $f(x, \xi)$  is called an outcome.

The set-based minmax robust counterpart of (1) is presented in [11] as:

$$\text{minimize}_{x \in \mathfrak{X}} \text{maximize}_{\xi \in \mathcal{U}} f(x, \xi) = (f_1(x, \xi), \dots, f_k(x, \xi))^T. \tag{2}$$

We say that a solution  $x^* \in \mathfrak{X}$  is set-based minmax robust Pareto optimal for problem (1), if there does not exist another solution  $x \in \mathfrak{X}$  such that  $f_{\mathcal{U}}(x) \subseteq f_{\mathcal{U}}(x^*) - \mathbb{R}_{\geq}^k$ , where  $\mathbb{R}_{\geq}^k = \{x \in \mathbb{R}^k : x_i \geq 0, i = 1, \dots, k\}$  [11]. This definition is based on the concept of lower set less order: let  $A$  and  $B$  be arbitrary closed sets,

then  $A \preceq^l B$  implies  $A \subseteq B - \mathbb{R}_{\geq}^k$ . Thus, when we compare two sets of vectors, we say  $A \preceq^l B$  if for all  $a \in A$  there exists  $b \in B$  such that  $a_i \leq b_i, i = 1, \dots, k$ .

Figure 1 illustrates an example of set-based minmax robustness with two objective functions to be minimized. In the example, we have a feasible set  $\mathcal{X} = \{x^1, x^2, x^3, x^4\}$  and an arbitrary uncertainty set  $\mathcal{U}$ . We plot the outcome set of the three solutions in the figure  $f_{\mathcal{U}}(x^1)$  (bold solid curve),  $f_{\mathcal{U}}(x^2)$  (bold dotted line),  $f_{\mathcal{U}}(x^3)$  (bold dashed line), and  $f_{\mathcal{U}}(x^4)$  (bold dash-dotted line). The gray thin lines help us to identify the borders of the outcome sets. Solution  $x^1$  is a set-based minmax robust Pareto optimal solution, since  $f_{\mathcal{U}}(x^1) - \mathbb{R}_{\geq}^2$  does not contain  $f_{\mathcal{U}}(x^2)$  nor  $f_{\mathcal{U}}(x^3)$ . Similarly, we can see that  $x^2$  and  $x^3$  are also set-based minmax robust Pareto optimal solutions. However,  $x^4$  is not set-based minmax robust Pareto optimal since  $f_{\mathcal{U}}(x^4) - \mathbb{R}_{\geq}^2$  contains  $f_{\mathcal{U}}(x^1)$  and  $f_{\mathcal{U}}(x^3)$ . The formulation (2) minimizes the worst case outcomes. As mentioned before, we need to first find the worst case outcomes and compare them as a whole. So, finding set-based minmax robust Pareto optimal solutions requires us to address these two challenges in a systematical way. Finding the worst case outcome set of a fixed solution  $x \in \mathcal{X}$  requires solving a multiobjective optimization problem with the objective functions to be maximized as follows:

$$\underset{\xi \in \mathcal{U}}{\text{maximize}} \quad (f_1(x, \xi), \dots, f_k(x, \xi))^T. \tag{3}$$

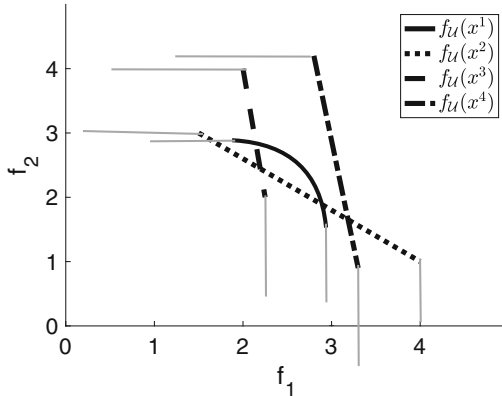


Fig. 1. Example of set-based minmax robustness

### 3 The SIBEA-R Method

In this section, we introduce SIBEA-R for approximating set-based minmax robust Pareto optimal solutions. We first introduce the steps of SIBEA-R. Then, we discuss details of the steps with a concentration on the further developments on SIBEA for set-based minmax robustness.

The SIBEA-R method takes the population size (NP) and the number of generations (NG) as the input and produces a set of non-dominated set-based minmax robust solutions  $A$  as the output. The basic steps are as follows:

**Step 1.** (Initialization) Generate an initial set of decision vectors  $P$  of size  $NP$  and find their worst case outcome sets by solving (3). Set the generation counter  $m = 1$ .

**Step 2.** (Mating) Create an offspring population  $Q$  using crossover and mutation operators and find their worst case outcome sets. Set  $P = P \cup Q$ .

**Step 3.** (Environmental selection) Rank the population  $P$  using lower set less order and sort the individuals into different fronts  $F^i, i = 1, 2, \dots$  and do the following:

- Set a new population  $P^1 = \emptyset$ . Set  $i = 1$  and  $P^1 = P^1 \cup F^1$ . As long as  $|P^1| < NP$ , set  $i = i + 1$ ,  $P^1 = P^1 \cup F^i$ . The notation  $|P^1|$  represents the cardinality of  $P^1$ .
- if  $|P^1| = NP$ , set  $P = P^1$  and go to Step 4. Otherwise, do the following until  $|P^1| = NP$ : identify the solutions with the worst rank  $P' \subset P^1$ .
- For each solution  $x \in P'$ , determine the loss of the value of the hypervolume indicator  $d(x)$  if it is removed from the set  $P'$ . Remove the solution with the smaller loss from  $P'$ , i.e., set  $P' = P' \setminus \{x\}$

**Step 4.** (Termination) If  $m > NG$ , set  $A = P^1$  and stop. Otherwise, set  $m = m + 1$  and go to Step 2.

In Steps 1 and 2, we consider the worst case outcome sets of the individuals and their offspring. We have mentioned earlier that for a fixed solution, finding its worst case outcomes is a multiobjective optimization problem with objectives to be maximized in the uncertainty set. We can solve the maximization problem with an evolutionary multiobjective optimization method to approximate a set of outcomes in the worst case. However, doing so requires a lot of computation resources. Thus, we should find a representative set of solutions of the maximization problem and use it to save the computation resource.

We propose to systematically solve a small number of scalarized subproblems to obtain the representative worst case outcome sets. For example, we can utilize the approach used in [6] to generate a set of evenly distributed points on a unit hyperplane in the objective space. Then, we use them as the reference points to optimize a series of the achievement scalarizing functions (see e.g., [26]). In what follows we denote the number of worst case outcomes in the representative worst case outcome set by  $W$  and the values of the uncertain parameters which the objective functions reach their worst case values by  $\xi^w, w = 1, \dots, W$ . The number of function evaluations depends on the solver used to solve the scalarized subproblems. In case of discrete scenarios in the uncertainty set, the number of function evaluations is  $k \times NP \times NG \times$  number of scenarios.

After we have found the representative worse case outcome sets of the individuals, we need to rank them and sort them into different fronts. We call this step set-based non-dominated sorting, where we define the dominance between two representative worst case outcome sets with lower set less order. The sorting procedure is inspired by that presented in [10]. The steps of the set-based non-dominated sorting are as follows:

**Step 1.** For each solution  $p \in P$ , set the domination count  $n_p = 0$  and the set of solutions dominated by  $p$  as an empty set  $S_p = \emptyset$ . Set  $P = P \setminus \{p\}$  and carry out the following steps:

- (a) For each  $q \in P$ , do the following:  
If for all  $f(q, \xi^w), w = 1, \dots, W$ , there exists  $f(p, \xi^w)$  such that  $f(q, \xi^w) \leq f(p, \xi^w)$ , set  $n_p = n_p + 1$ .  
Otherwise if for all  $f(p, \xi^w), w = 1, \dots, W$ , there exists  $f(q, \xi^w)$  such that  $f(p, \xi^w) \leq f(q, \xi^w)$ , set  $S_p = S_p \cup \{q\}$
- (b) If  $n_q = 0$ , then  $p^{rank} = 1$  and  $F^1 = F^1 \cup \{p\}$ .

**Step 2.** Set front counter  $i = 1$

**Step 3.** Do the following steps until  $F^i = \emptyset$

For each  $p \in F^i$

for each  $q \in S_p$

set  $n_q = n_q - 1$

if  $n_q = 0$ , then  $q^{rank} = i + 1$ , and  $F^{i+1} = F^{i+1} \cup \{q\}$ , set  $i = i + 1$   
and continue with Step 3 to the next front.

In the set-based non-dominated sorting, Step 2(a) is for checking if  $f_U(p) \preceq^l f_U(q)$  or  $f_U(q) \preceq^l f_U(p)$ . We pair-wise compare the solutions and go through the outcomes in the representative worst case outcome sets.

After we have sorted the solutions into different fronts, we start the environmental selection in Step 3. We fill the next generation population incrementally starting from solutions that are in  $F^1$  until the number of solutions exceeds the population size  $NP$ . Then we delete the solutions from the last front based on the loss of the value of the hypervolume indicator (see e.g., [1, 28]). We calculate the loss of the hypervolume when deleting a solution  $x'$  as  $d(x') = H(S) - H(S')$ , where  $S = \{f_U(x) : x \in P'\}$  and  $S' = S \setminus \{f_U(x')\}$ . Here, we use  $\tilde{f}_U$  instead of  $f_U$  because we consider the representative worst case outcome sets.

After step 3, we have a new population. If the number of generations has been exceeded, we terminate the solution process and take the set-based non-dominated solutions of the last generation as the output set  $A$ . If the number of generations has not been exceeded, we continue by going to Step 2.

After obtaining the set  $A$ , a decision maker should choose a final solution. For example, [27] uses an interactive post-processing procedure to find the final solution based on preference information. In the interactive process, we present the outcome of a solution in the nominal case which is the undisturbed or usual case. Then, the decision maker can specify her or his preferences for a more desired solution until (s)he finds a satisfactory solution. The purpose is to help the decision maker to find the final solution based on the nominal value and at the same time the solution is the best possible when the worst case happens.

## 4 Numerical Results

In this section, we demonstrate the usage of the SIBEA-R method with two example problems. The examples help us to test our proposal of using set-based non-dominated sorting in an evolutionary algorithm. The first example problem is a simple linear problem based on one of the examples presented in [25]:



$$\left( \begin{array}{l} \text{minimize} \quad \left( \begin{array}{l} 2\xi_1 x_1 - 3\xi_2 x_2 \\ 5\xi_1 x_1 + \xi_2 x_2 \end{array} \right) \\ \text{subject to} \quad 0 \leq x_1 \leq 1.5 \\ \quad \quad \quad 0 \leq x_2 \leq 3 \end{array} \right)_{\xi \in \mathcal{U}}, \quad (4)$$

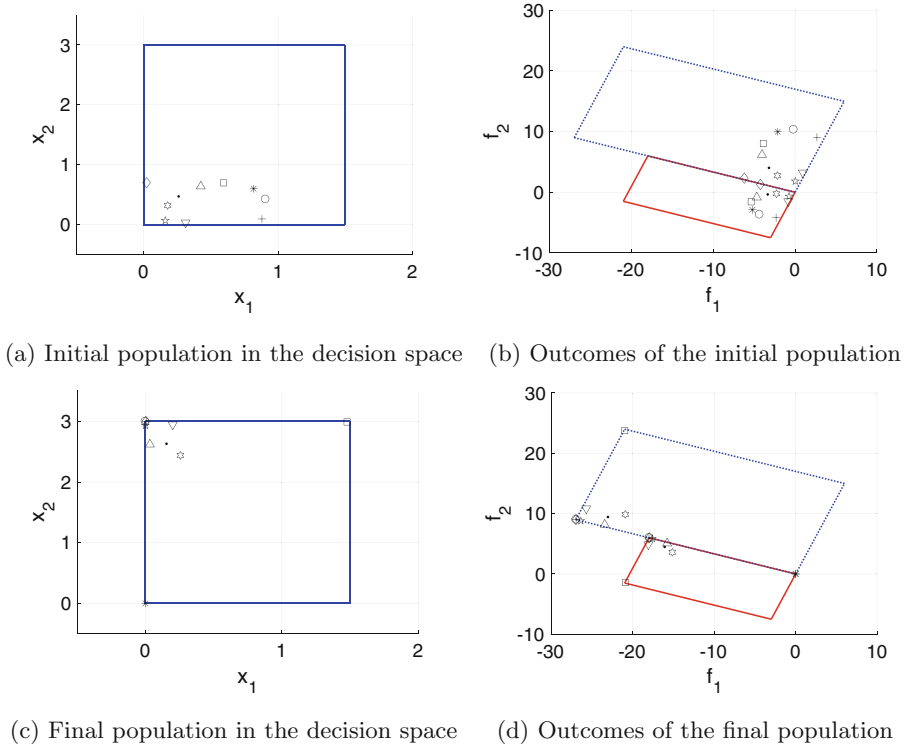
where  $\mathcal{U} = \left\{ \begin{pmatrix} -1 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \end{pmatrix} \right\}$ .

In the experiments, we used the default setting of parameters as in the implementation of SIBEA in [7]. For (4), we can compute the outcomes in both possible sets of values for the uncertain parameters. We first illustrate the evolution of the population, we visualize the initial generation in the decision space in Fig. 2a and in the objective space in Fig. 2b. In the figures, the solid lines are the borders of the feasible set and we visualize 10 individuals because of limited varieties of markers. In Fig. 2b, the same marker appears twice because of the two possible cases in  $\mathcal{U}$ . We use SIBEA-R to evolve the population by considering their outcome sets (each set consists of two outcomes with the same marker in the figure). After 100 generations, the last generation is shown in Fig. 2c in the decision space and in Fig. 2d in the objective space.

We then studied the final populations of 20 independent runs with  $NP = 30$ . It is not even possible to compute a complete set of set-based robust Pareto optimal solutions for linear problems like (4). To the best of our knowledge, methods with similar ideas in the literature (e.g., [2]) had a different definition of robust Pareto optimality. We cannot easily benchmark the example problems. Thus, we first visually compare the solutions computed by SIBEA-R with 30 solutions computed by the weighted-sum approach proposed in [11]. The purpose is to use the solutions computed by the weighted-sum approach as references.

Figures 3a and b illustrate the solutions computed by the weighted-sum approach and SIBEA-R. The solutions computed by the weighted-sum approach are marked as solid red circles in the figures and the solutions computed by SIBEA-R are marked by the gray plus signs. In the figures, the gray cloud consists of the solutions computed with 20 runs of the SIBEA-R method. We can see that SIBEA-R was able to find the solutions found by the weighted-sum approach. In addition, SIBEA-R also found other solutions in the interior of the feasible space. The existence of set-based minmax Pareto optimal solutions in the interior of the feasible space is proven in [20]. For example, the point (0.5, 2.4) is set-based minmax robust Pareto optimal which can be checked by the definition. Based on the visualizations, we can observe that SIBEA-R has considered the outcomes concerning both sets of possible values of the uncertain parameters and found a set of non-dominated set-based minmax robust solutions.

The second example problem is based on a standard benchmark problem, ZDT2 (see, e.g., [8]). In this problem, we introduced two uncertain parameters which stem from a polyhedral uncertainty set. A polyhedral uncertainty set is given as the convex hull of a finite set of points. Even though modifying the problem can cause the loss of the characteristics of the carefully designed test problems, our purpose is to illustrate the solutions found by SIBEA-R and the usage of them for decision making. For the ZDT2-based problem, we set

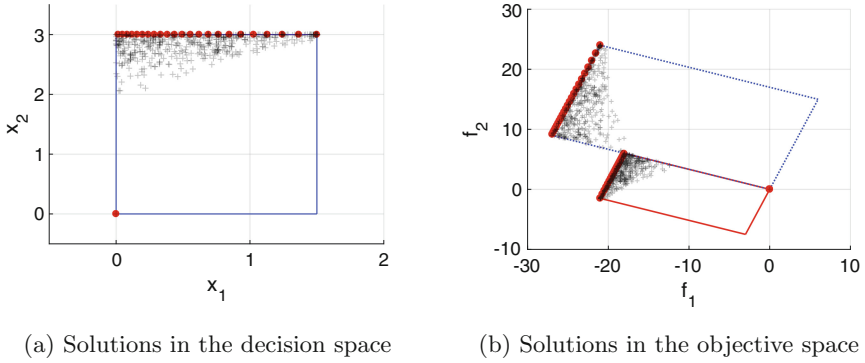


**Fig. 2.** The evolution of the population by SIBEA-R

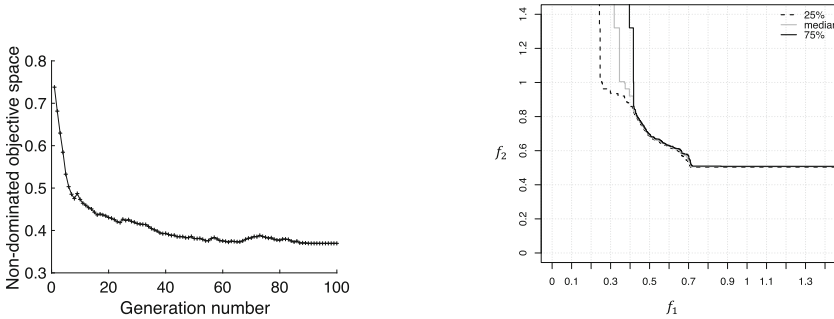
$NG = 100, NP = 30$  and found six worst case outcomes to represent the worst case outcome set. We run SIBEA-R 20 times to solve the problem.

We analyzed the results with the so-called average non-dominated objective space (i.e., the percentage of the volume of objective space between the ideal point and a reference vector which are not covered by the solutions) in each generation in all the runs to observe the convergence (see details in [29]). We also analyzed the attainment surface of the worst case outcome sets from multiple runs with the empirical attainment function graphical tools [18, 19]. We visualized the 25%, 50%, 75% attainment surfaces.

The average non-dominated objective space in each generation for the 20 runs of the ZDT2-based problem is illustrated in Fig. 4. The figure shows that the non-dominated objective space gradually reduced with generations and at the final generations, the average non-dominated space stayed stable. This means that the objective function values of solutions reduced along the generations. The attainment surfaces of the results from the 20 runs are shown in Fig. 5. The figure illustrates that the solutions tend to converge to the area bounded by the intervals  $f_1 = [0.5, 0.8], f_2 = [0.2, 0.7]$ . Based on the experiment results, we can observe that SIBEA-R was able to improve the populations with the generations and the final populations of different runs were similar.



**Fig. 3.** Solutions computed by the weighted-sum approach and SIBEA-R



**Fig. 4.** Average non-dominated objective space, ZDT2-based problem

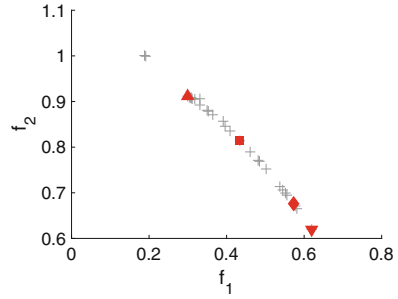
**Fig. 5.** Attainment surface, ZDT2-based problem

After SIBEA-R has found a set of non-dominated set-based minmax robust solutions, the set can be used for decision making. We illustrate the usage with a reference point-based interactive approach (see e.g., [21] for a detailed description). In a reference point-based approach, the decision maker specifies the desired objective function values as a reference point. We find a solution which satisfies the reference point as well as possible and present the solution to the decision maker. This kind of interactive process continues until the decision maker finds a most satisfactory solution. We used the final population of a run of the ZDT2-based problem and helped a decision maker to choose a final solution based on their outcomes in the nominal case. In the nominal case, the uncertain parameters behave normally without disturbance. So, we used the original ZDT2 problem as the nominal case. We carried out four iterations. The reference points and the solutions found are illustrated in Table 1. The solutions are also presented in Fig. 6 with different markers. The decision maker took the third solution as the final solution since it is the nearest to her desired values.

In the examples, we observed that SIBEA-R was able to find set-based minmax robust Pareto optimal solutions found by the weighted-sum approach. It was

**Table 1.** Interactive post-processing

Ref.	Solution	Marker
$(0.3, 0.7)^T$	$(0.43, 0.81)^T$	Square
$(0.3, 0.95)^T$	$(0.3, 0.91)^T$	Up triangle
$(0.5, 0.6)^T$	$(0.57, 0.67)^T$	Diamond
$(0.8, 0.6)^T$	$(0.61, 0.61)^T$	Down triable



**Fig. 6.** Solutions found based on reference points

also able to find some solutions that the weighted-sum approach was not able to find. In the ZDT2-based problem, SIBEA-R was stable regarding finding similar final populations in different runs. These observations suggested that SIBEA-R has an appealing potential for approximating set-based minmax robust Pareto optimal solutions, which can be then used for decision making.

## 5 Conclusions

In this paper, we proposed SIBEA-R to compute an approximated set of set-based minmax robust Pareto optimal solutions. This is an initial study to explore opportunities evolutionary multiobjective optimization methods can provide in tackling challenges with robustness which are otherwise difficult. In SIBEA-R, instead of considering single outcomes, we considered the worst case outcome sets of solutions. We proposed a set-based non-dominated sorting procedure based on the lower set less order to rank the solutions for environmental selection. We illustrated the utilization of SIBEA-R with two example problems. The experiments on the example problems suggest that SIBEA-R can approximate set-based minmax robust Pareto optimal solutions. We also illustrated how the solutions found by SIBEA-R can be used in decision making.

Due to the set-based non-dominated sorting and the calculation of the hypervolume of outcome sets, SIBEA-R is computationally expensive and it tends to work with small population sizes. Thus, an immediate future research direction is to improve the computational efficiency and enable the calculation of a larger number of non-dominated set-based minmax robust solutions. In this paper, we only presented a limited amount of numerical experiments. It is necessary to extend the numerical experiments to a wider range of problems to further identify the strengths and limitations of SIBEA-R.

**Acknowledgments.** We thank Dr. Tinkle Chugh for useful discussions and providing an implementation of SIBEA. This research is related to Decision Analytics (DEMO).

## References

1. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: Optimal  $\mu$ -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, New York (2009)
2. Avigad, G., Branke, J.: Embedded evolutionary multi-objective optimization for worst case robustness. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO 2008, pp. 617–624. ACM (2008)
3. Bader, J., Brockhoff, D., Welten, S., Zitzler, E.: On using populations of sets in multiobjective optimization. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 140–154. Springer, Heidelberg (2009). <https://doi.org/10.1007/978-3-642-01020-0-15>
4. Bader, J., Zitzler, E.: Robustness in hypervolume-based multiobjective search. Technical report, TIK Report 317 (2010)
5. Bokrantz, R., Fredriksson, A.: Necessary and sufficient conditions for Pareto efficiency in robust multiobjective optimization. *Eur. J. Oper. Res.* **262**(2), 682–692 (2017)
6. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **20**(5), 773–791 (2016)
7. Chugh, T., Sindhya, K., Hakanen, J., Miettinen, K.: An interactive simple indicator-based evolutionary algorithm (I-SIBEA) for multiobjective optimization problems. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) EMO 2015. LNCS, vol. 9018, pp. 277–291. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-15934-8\\_19](https://doi.org/10.1007/978-3-319-15934-8_19)
8. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, Heidelberg (2006). <https://doi.org/10.1007/978-0-387-36797-2>
9. Deb, K., Gupta, H.: Introducing robustness in multi-objective optimization. *Evol. Comput.* **14**(4), 463–494 (2006)
10. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
11. Ehrgott, M., Ide, J., Schöbel, A.: Minmax robustness for multi-objective optimization problems. *Eur. J. Oper. Res.* **239**(1), 17–31 (2014)
12. Gaspar-Cunha, A., Covas, J.A.: Robustness in multi-objective optimization using evolutionary algorithms. *Comput. Optim. Appl.* **39**(1), 75–96 (2007)
13. Gong, D., Sun, J., Miao, Z.: A set-based genetic algorithm for interval many-objective optimization problems. *IEEE Trans. Evol. Comput.* **22**(1), 47–60 (2018)
14. Ide, J., Schöbel, A.: Robustness for uncertain multi-objective optimization: a survey and analysis of different concepts. *OR Spectr.* **38**(1), 235–271 (2016)
15. Konur, D., Farhangi, H.: Set-based min-max and min-min robustness for multi-objective robust optimization. In: Coperich, K., Cudney, E., Hembhard, H. (eds.) Proceedings of the 2017 Industrial and Systems Engineering Research Conference, pp. 1–6. Institute of Industrial and Systems Engineers (2017)
16. Kuroiwa, D., Lee, G.M.: On robust multiobjective optimization. *Vietnam J. Math.* **40**(2, 3), 305–317 (2012)
17. Li, M., Azarm, S., Aute, V.: A multi-objective genetic algorithm for robust design optimization. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO 2005, pp. 771–778 (2005)

18. López-Ibáñez, M.: EAF graphical tools. <http://lopez-ibanez.eu/eaftools>
19. López-Ibáñez, M., Paquete, L., Stützle, T.: Exploratory analysis of stochastic local search algorithms in biobjective optimization. In: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) *Experimental Methods for the Analysis of Optimization Algorithms*, pp. 209–222. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-02538-9\\_9](https://doi.org/10.1007/978-3-642-02538-9_9)
20. Majewski, D.: Robust bi-objective linear optimization. Master's thesis, University of Göttingen (2014)
21. Miettinen, K.: *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston (1999)
22. Miettinen, K., Mustajoki, J., Stewart, T.J.: Interactive multiobjective optimization with NIMBUS for decision making under uncertainty. *OR Spectr.* **36**(1), 39–56 (2014)
23. Rodriguez-Marn, L., Sama, M.:  $(\lambda, c)$ -contingent derivatives of set-valued maps. *J. Math. Anal. Appl.* **335**(2), 974–989 (2007)
24. Wiecek, M.M., Blouin, V.Y., Fadel, G.M., Engau, A., Hunt, B.J., Singh, V.: Multi-scenario multi-objective optimization with applications in engineering design. In: Barichard, V., Ehrgott, M., Gandibleux, X., T'Kindt, V. (eds.) *Multiobjective Programming and Goal Programming: Theoretical Results and Practical Applications*. LNE, vol. 618, pp. 283–298. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-540-85646-7\\_26](https://doi.org/10.1007/978-3-540-85646-7_26)
25. Wiecek, M.M., Dranichak, G.M.: Robust multiobjective optimization for decision making under uncertainty and conflict. In: Gupta, A., Capponi, A., Smith, J.C., Greenberg, H.J. (eds.) *Optimization Challenges in Complex, Networked and Risky Systems*, pp. 84–114 (2016)
26. Wierzbicki, A.P.: A mathematical basis for satisficing decision making. *Math. Model.* **3**(5), 391–405 (1982)
27. Zhou-Kangas, Y., Miettinen, K., Sindhya, K.: Interactive multiobjective robust optimization with NIMBUS. In: *Proceedings of Clausthal-Goettingen International Workshop on Simulation Science 2017*. Springer (2018, to appear)
28. 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). [https://doi.org/10.1007/978-3-540-70928-2\\_64](https://doi.org/10.1007/978-3-540-70928-2_64)
29. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. Technical report, TIK Report 103 (2002)
30. Zitzler, E., Thiele, L., Bader, J.: On set-based multiobjective optimization. *IEEE Trans. Evol. Comput.* **14**(1), 58–79 (2010)



# Extending the Speed-Constrained Multi-objective PSO (SMPSO) with Reference Point Based Preference Articulation

Antonio J. Nebro<sup>1</sup>(✉), Juan J. Durillo<sup>2</sup>, José García-Nieto<sup>1</sup>,  
Cristóbal Barba-González<sup>1</sup>, Javier Del Ser<sup>3,4,5</sup>, Carlos A. Coello Coello<sup>6</sup>,  
Antonio Benítez-Hidalgo<sup>1</sup>, and José F. Aldana-Montes<sup>1</sup>

<sup>1</sup> Dept. de Lenguajes y Ciencias de la Computación,  
Universidad de Málaga, Málaga, Spain  
antonio@lcc.uma.es

<sup>2</sup> Leibniz Supercomputing Centre, Munich, Germany

<sup>3</sup> TECNALIA Research and Innovation, Derio, Spain

<sup>4</sup> University of the Basque Country (UPV/EHU), Bilbao, Spain

<sup>5</sup> Basque Center for Applied Mathematics (BCAM), Bilbao, Spain

<sup>6</sup> Computer Science Department, CINVESTAV-IPN  
(Evolutionary Computation Group), Ciudad de México, Mexico

**Abstract.** The Speed-constrained Multi-objective PSO (SMPSO) is an approach featuring an external bounded archive to store non-dominated solutions found during the search and out of which leaders that guide the particles are chosen. Here, we introduce SMPSO/RP, an extension of SMPSO based on the idea of reference point archives. These are external archives with an associated reference point so that only solutions that are dominated by the reference point or that dominate it are considered for their possible addition. SMPSO/RP can manage several reference point archives, so it can effectively be used to focus the search on one or more regions of interest. Furthermore, the algorithm allows interactively changing the reference points during its execution. Additionally, the particles of the swarm can be evaluated in parallel. We compare SMPSO/RP with respect to three other reference point based algorithms. Our results indicate that our proposed approach outperforms the other techniques with respect to which it was compared when solving a variety of problems by selecting both achievable and unachievable reference points. A real-world application related to civil engineering is also included to show up the real applicability of SMPSO/RP.

**Keywords:** Multi-objective optimization · SMPSO  
Decision making · Reference point

## 1 Introduction

Dealing with a multi-objective optimization problem involves finding its Pareto front or a reasonably good approximation to it in case of using non-exact

optimization techniques such as metaheuristics [1]. This accuracy is expressed, in general, in terms of convergence and diversity, with the aim of offering the decision maker (DM) a set of optimal or quasi-optimal solutions evenly spread along the Pareto front. In practice, the DM is usually only interested in a portion of the Pareto front, which can be provided by integrating user's preferences within multi-objective metaheuristics [2]. The preference information can be given to the algorithm *a priori*, before starting the search process, and/or in an interactive way, during the search.

In this paper, we propose an extension of the SMPSO multi-objective particle swarm algorithm [3] to allow DMs to guide the search towards one or more regions of interest by indicating preferences *a priori* and interactively. SMPSO features a bounded-size external archive where a diverse subset of the non-dominated solutions found during the search is kept and from which global leaders are chosen to compute the speed of the particles. When the archive becomes full, a density estimator (e.g., the crowding distance [4]) is applied in order to remove the solution which least contributes in terms of diversity.

Our extension makes use of reference points as a mean for articulating DM's preferences. We associate an external archive to each reference point. Newly solutions (i.e., every time a particle changes its position) are checked to be added within each of these archives as follows: if the newly generated solution and the archive reference point are non-dominated with respect to each other, nothing is done; otherwise, the former is added to the archive using the same strategy as in SMPSO. This way, reference point archives only keep the non-dominated solutions of the selected preference region. Our proposal, called SMPSO/RP, also modifies the leader selection strategy to select an external archive randomly and then take the leader from it; this mechanism promotes diversity of the swarm and avoids concentrating the search in a single region of interest.

As solving real-world problems might be highly time-consuming, adding the capability of changing the reference points interactively is a basic feature that allows the DM to adjust and focus the search towards the regions of interest. On the contrary, approaches based on static reference points would require re-starting the search from scratch every time the reference point is changed. In SMPSO/RP, the strategy followed when a reference point is changed is to remove all the solutions of the corresponding archive that are non-dominated with respect the new reference point.

The main contributions of this paper are summarized as follows:

1. A new algorithm, SMPSO/RP, that extends SMPSO by incorporating interactive reference point preference articulation. SMPSO/RP has the following features:
  - Ability to deal with one or more DM preferences or regions of interest.
  - Ability to interactively change DM preferences by means of changing the desired reference points.
  - Ability of parallel evaluations of particles.
  - GUI for visualizing the computed front evolution for problems with two and three objectives.



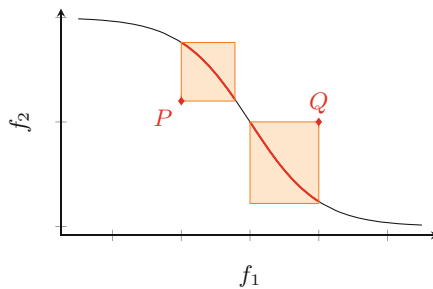
2. Comparison against three reference point based multi-objective evolutionary algorithms.
3. Application of SMPSO/SP to a real-world problem of the domain of civil engineering.
4. Freely available implementation of SMPSO/RP within the jMetal [5] framework<sup>1</sup>.

The rest of the paper is organized as follows. Section 2 contains background concepts and our proposal is described in Sect. 3. We devote Sects. 4 and 5 for assessing the performance of SMPSO/RP. A real-world application of our proposal is included in Sect. 6. The conclusions and some possible paths of future work are indicated in Sect. 7.

## 2 Background

Preference-based multi-objective metaheuristics aim at finding the most interesting parts according the criteria of a DM instead of the full Pareto front. This has been a relatively active research area in the last two decades [6–8].

In this work we are interested in the reference point method [9]. This method constitutes a simple way to delimit an interest region of the objective space by the definition of a user-defined point by the DM, as it requires no parameter defining the width of the region of interest. Given a reference point  $P$ , the region of interest is the subset of the Pareto front dominated by  $P$  if this is not achievable, or the subset of the Pareto front dominating  $P$  if this is achievable. This approach is very similar to the g-dominance concept [10]. Figure 1 illustrates an example of the regions of interest delimited by an achievable and unachievable reference point. Our purpose is to extend SMPSO to allow guiding the search according to this kind of preference articulation mechanism.



**Fig. 1.** Examples of the regions of interest delimited by points  $P$  (unachievable) and  $Q$  (achievable).

<sup>1</sup> <https://github.com/jMetal/jMetal>.

SMPSO [3] is an algorithm following the classic particle swarm algorithm metaheuristic [11], so it manages a set of solutions or *particles* which are referred to as the *swarm*. The position of particle  $\mathbf{x}_i$  at the generation  $t$  is updated with Eq. (1):

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \mathbf{v}_i(t) \quad (1)$$

where the factor  $\mathbf{v}_i(t)$  is known as velocity, and it is defined as:

$$\mathbf{v}_i(t) = w \cdot \mathbf{v}_i(t-1) + C_1 \cdot r_1 \cdot (\mathbf{x}_{p_i} - \mathbf{x}_i) + C_2 \cdot r_2 \cdot (\mathbf{x}_{g_i} - \mathbf{x}_i) \quad (2)$$

In Eq. (1),  $\mathbf{x}_{p_i}$  is the best solution that  $\mathbf{x}_i$  has viewed,  $\mathbf{x}_{g_i}$  is the best particle (known as the *leader*) that the entire swarm has viewed,  $w$  is the inertia weight of the particle and controls the trade-off between global and local influence,  $r_1$  and  $r_2$  are two uniformly distributed random numbers in the range  $[0, 1]$ , and  $C_1$  and  $C_2$  are specific parameters which control the effect of the personal and global best particles.

The motivation to develop SMPSO was originated after stating that the MOPSO algorithm [12], a previously proposed multi-objective PSO based on Eqs. 1 and 2, was able to efficiently solve parameter scalable problems [13] but it had difficulties when dealing with the (multi-frontal) ZDT4 problem. We discovered that by applying the *constriction coefficient* (Eq. (3)) obtained from the constriction factor  $\chi$  originally developed by Clerc and Kennedy (Eq. (2)) in [14], SMPSO could successfully solve that problem with up to 2048 variables. The constriction coefficient is defined as:

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \quad (3)$$

where

$$\varphi = \begin{cases} C_1 + C_2 & \text{if } C_1 + C_2 > 4 \\ 0 & \text{if } C_1 + C_2 \leq 4 \end{cases} \quad (4)$$

Additionally, SMPSO further bounds the accumulated velocity of each variable  $j$  (in each particle) by means of the following *velocity constriction* equation:

$$v_{i,j}(t) = \begin{cases} \delta_j & \text{if } v_{i,j}(t) > \delta_j \\ -\delta_j & \text{if } v_{i,j}(t) \leq -\delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \quad (5)$$

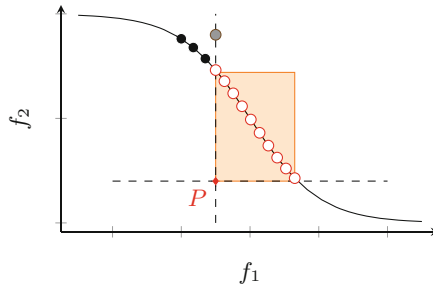
where

$$\delta_j = \frac{(\text{upper\_limit}_j - \text{lower\_limit}_j)}{2} \quad (6)$$

As commented beforehand, SMPSO adopts the use of an external archive to store the non-dominated solutions and out of which leaders are chosen.

### 3 Algorithm Proposal

The basic component of SMPSO/RP is the concept of reference point archive (i.e., an external archive with an associated reference point). The basic idea is to modify the strategy for adding new solutions to the external archive, in such a way that only solutions within the area of interest defined by a reference point  $P$  are kept. The basic approach is as follows: given a solution  $a$  to be inserted, it is first compared with  $P$ . If either  $a$  dominates  $P$  or vice-versa, then  $a$  is checked for insertion in the archive as done in the original SMPSO. However, if none of them dominates the other,  $a$  is discarded.

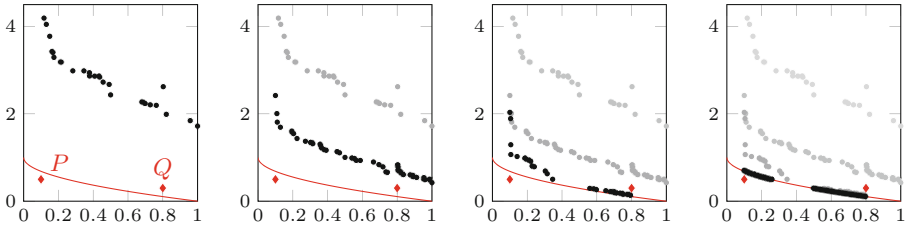


**Fig. 2.** Example illustrating how a point in the boundary of the region of interest can remain in the reference point archive.

This strategy does not work properly in two scenarios. First, when the archive is empty and only non-dominated solutions regarding  $P$  are generated by the search. This scenario results in an empty archive which renders the working behavior of SMPSO impossible, as it may need to select a global leaders from this archive. Our solution is to incorporate the non-dominated solution if the archive is empty. This solution is expected to be removed later by any other solution dominating it.

The second situation has to do with a poor convergence of solutions on any of the ends of the region of interest. The Fig. 2 illustrates this issue. The white points are inside the region of interest defined by  $P$ , and the point with a gray background is exactly in the boundary of this region. The gray point is non-dominated regarding the white points and therefore always kept in the archive as it is assigned an infinite crowding distance by the density estimator. However, it is not close to the Pareto front, so convergence is negatively affected. This would not happen if some of the black points on the left would belong to the region of interest, because they dominate the gray point, which would have been either removed or never inserted. Our approach, then, is to insert non-dominated points which are outside the region of interest with a certain probability for the sake of filtering these poorly converged points in the ends of the region of interest (after some pilot tests, we have set this probability to 0.05). These points outside the area of interest are removed later from the archive.

SMPSO/RP can have more than one reference point archive, so the DM can indicate several regions of interest. The working procedure of SMPSO/RP resembles that of SMPSO, except for subtle yet very relevant differences: the leader selection, which take a leader from a randomly selected reference point archive, and all the archives are updated when any particle moves. SMPSO/RP has been implemented in jMetal 5 [15], which provides parallelism support to evaluate all the solutions in a population or swarm in parallel in multi-core systems. As only the evaluations are computed in parallel, linear speed-ups cannot be expected given that the rest of the algorithm is sequential code. However, these scheme has the advantage that no changes in the algorithm are needed.



**Fig. 3.** Example of the front evolution when solving the ZDT1 problem indicating the unachievable reference point  $P$  (0.1,0.5) and the achievable one  $Q$  (0.8,0.3). The plots depicts the fronts at generations 10, 50, 80, and 120. The population and archive sizes are set to 100.

To illustrate how SMPSO/RP works, Fig. 3 depicts an example of how the computed front evolves over the generations when two reference points, one of each type, have been indicated by the DM.

## 4 Experimental Setup

In this section, we detail the experimentation we have carried out to assess the performance of SMPSO/RP. We describe first the selected algorithms to be compared with our proposal and their parameter settings. Then, we present the chosen benchmark problems and the reference points that have been specified. Finally, we describe the experimentation methodology.

The regions of interest computed by SMPSO/RP are delimited by the dominance relationship in relation to the reference point. Hence, we have considered three algorithms following the same principle. These algorithms are WASF-GA [6], gSMS-EMOA, and gNSGA-II.

WASF-GA or Weighting Achievement Scalarizing Function Genetic Algorithm uses an *scalarization* approach with weight vectors. In each generation WASF-GA classifies individuals into fronts by taking into account the achievement scalarizing function and the reference point. It also requires to know the

**Table 1.** Achievable and unachievable points selected for each of the ZDT, DTLZ, and WFG problems.

Problem	Achievable	Unachievable	Problem	Achievable	Unachievable
ZDT1	(0.80, 0.60)	(0.20, 0.40)	WFG1	(1.31, 1.61)	(0.49, 0.88)
ZDT2	(0.80, 0.80)	(0.50, 0.30)	WFG2	(1.80, 2.91)	(0.23, 0.20)
ZDT3	(0.30, 0.80)	(0.20, 0.00)	WFG3	(1.75, 2.55)	(0.56, 1.61)
ZDT4	(0.99, 0.95)	(0.08, 0.25)	WFG4	(1.88, 3.71)	(0.29, 2.93)
ZDT6	(0.78, 0.61)	(0.39, 0.21)	WFG5	(1.88, 2.46)	(0.47, 1.98)
DTLZ1	(0.41, 0.36)	(0.00, 0.02)	WFG6	(1.46, 3.44)	(0.28, 0.10)
DTLZ2	(0.83, 0.92)	(0.07, 0.51)	WFG7	(1.17, 3.74)	(0.11, 3.03)
DTLZ3	(0.87, 1.00)	(0.15, 0.42)	WFG8	(1.92, 3.60)	(0.29, 3.56)
DTLZ4	(0.97, 0.59)	(0.41, 0.51)	WFG9	(1.83, 3.92)	(0.81, 2.15)
DTLZ5	(0.97, 0.59)	(0.03, 0.27)			
DTLZ6	(0.76, 0.84)	(0.08, 0.48)			
DTLZ7	(0.85, 3.88)	(0.62, 1.27)			

ranges of the objective solutions in the Pareto front from the ideal and nadir points, which need to be estimated.

The other chosen algorithms, gNSGA-II and gSMS-EMOA, are variants of the original NSGA-II and SMS-EMOA algorithms modified to incorporate the concept of g-dominance [10]. NSGA-II [4] is by far the most well-known and used multi-objective evolutionary algorithm, and it is characterized by following a generational scheme which applies a non-dominated sorting algorithm and the crowding distance density estimator to promote, respectively, convergence and diversity. SMS-EMOA [16] is a typical representative of indicator-based multi-objective evolutionary metaheuristics; it is based on a steady-state version of NSGA-II but replacing the crowding distance by the hypervolume contribution. None of the algorithms evaluated in this paper requires additional parameter to determine the extent of the region of interest. Algorithms requiring so, like R-NSGA-II [17] or RPSO-SS [18], are out of the scope of this initial analysis.

All the solvers share common parameter settings. The population/swarm size is set to 100. The stopping condition is to compute 25,000 function evaluations. The mutation operator (turbulence in SMPPO/RP) is the polynomial mutation, applied with probability of  $1/L$  (being  $L$  the number of decision variables of the problem) and a distribution index of 0.20. gNSGA-II, gSMS-EMOA, and WASF-GA apply SBX crossover with a probability of 0.9 and distribution index of 20.0. As these three algorithms only allow indicating a reference point, SMPPO/RP is configured with an external archive with capacity for 100 solutions. WASF-GA generates 100 weight vectors with  $\epsilon = 0.01$ .

As benchmark problems, we have selected the ZDT [19], DTLZ [20], and WFG [21] families and we have solved them by indicating both an achievable and an unachievable reference point. In this study, we have considered the

**Table 2.** Median and interquartile range of the hypervolume quality indicator when solving the problems indicating achievable reference points.

	SMPSO/RP	gSMS-EMOA	gNSGAI	WASF-GA
ZDT1	5.58e - 01 <sub>7.6e-05</sub>	5.58e - 01 <sub>6.6e-05</sub>	5.55e - 01 <sub>8.9e-04</sub>	5.57e - 01 <sub>4.7e-04</sub>
ZDT2	4.48e - 01 <sub>6.9e-05</sub>	4.48e - 01 <sub>9.4e-05</sub>	4.43e - 01 <sub>1.2e-03</sub>	4.46e - 01 <sub>5.7e-04</sub>
ZDT3	3.60e - 01 <sub>3.9e-05</sub>	3.59e - 01 <sub>9.7e-05</sub>	3.57e - 01 <sub>5.4e-04</sub>	3.57e - 01 <sub>3.2e-04</sub>
ZDT4	6.43e - 01 <sub>2.3e-04</sub>	6.40e - 01 <sub>4.3e-03</sub>	6.35e - 01 <sub>4.9e-03</sub>	6.38e - 01 <sub>4.2e-03</sub>
ZDT6	4.16e - 01 <sub>6.3e-05</sub>	4.12e - 01 <sub>1.4e-03</sub>	3.95e - 01 <sub>7.3e-03</sub>	4.03e - 01 <sub>2.4e-03</sub>
DTLZ1	4.94e - 01 <sub>7.7e-05</sub>	0.00e + 00 <sub>0.0e+00</sub>	0.00e + 00 <sub>0.0e+00</sub>	4.88e - 01 <sub>7.3e-03</sub>
DTLZ2	3.96e - 01 <sub>1.2e-04</sub>	3.96e - 01 <sub>1.5e-05</sub>	3.94e - 01 <sub>3.9e-04</sub>	3.96e - 01 <sub>2.2e-05</sub>
DTLZ3	2.85e - 01 <sub>8.7e-05</sub>	0.00e + 00 <sub>0.0e+00</sub>	0.00e + 00 <sub>0.0e+00</sub>	1.41e - 01 <sub>2.0e-01</sub>
DTLZ4	4.11e - 01 <sub>9.1e-05</sub>	4.11e - 01 <sub>2.8e-05</sub>	4.09e - 01 <sub>7.2e-04</sub>	4.10e - 01 <sub>4.1e-01</sub>
DTLZ5	4.12e - 01 <sub>9.0e-05</sub>	4.13e - 01 <sub>1.4e-05</sub>	4.11e - 01 <sub>5.1e-04</sub>	4.12e - 01 <sub>4.5e-05</sub>
DTLZ6	4.48e - 01 <sub>8.3e-05</sub>	0.00e + 00 <sub>0.0e+00</sub>	0.00e + 00 <sub>0.0e+00</sub>	5.25e - 02 <sub>9.0e-02</sub>
DTLZ7	3.05e - 01 <sub>2.7e-05</sub>	3.04e - 01 <sub>1.1e-01</sub>	3.03e - 01 <sub>1.1e-01</sub>	3.03e - 01 <sub>8.1e-05</sub>
WFG1	0.00e + 00 <sub>0.0e+00</sub>	0.00e + 00 <sub>0.0e+00</sub>	0.00e + 00 <sub>1.3e-02</sub>	0.00e + 00 <sub>3.5e-03</sub>
WFG2	4.74e - 01 <sub>3.4e-04</sub>	4.74e - 01 <sub>1.2e-03</sub>	4.73e - 01 <sub>1.2e-03</sub>	4.72e - 01 <sub>1.1e-03</sub>
WFG3	4.94e - 01 <sub>2.3e-04</sub>	4.94e - 01 <sub>1.1e-03</sub>	4.91e - 01 <sub>1.2e-03</sub>	4.93e - 01 <sub>7.4e-04</sub>
WFG4	3.51e - 01 <sub>8.8e-03</sub>	3.53e - 01 <sub>3.1e-05</sub>	3.51e - 01 <sub>7.0e-04</sub>	3.52e - 01 <sub>3.2e-04</sub>
WFG5	2.52e - 01 <sub>2.9e-05</sub>	2.52e - 01 <sub>2.5e-05</sub>	2.51e - 01 <sub>2.0e-04</sub>	2.51e - 01 <sub>2.6e-05</sub>
WFG6	4.48e - 01 <sub>1.1e-04</sub>	3.02e - 01 <sub>2.4e-01</sub>	3.10e - 01 <sub>1.8e-01</sub>	3.68e - 01 <sub>9.6e-02</sub>
WFG7	4.42e - 01 <sub>1.8e-04</sub>	4.43e - 01 <sub>2.6e-04</sub>	4.40e - 01 <sub>7.8e-04</sub>	4.42e - 01 <sub>4.2e-04</sub>
WFG8	2.56e - 01 <sub>7.5e-02</sub>	2.26e - 01 <sub>1.3e-03</sub>	2.26e - 01 <sub>1.2e-03</sub>	2.26e - 01 <sub>5.8e-04</sub>
WFG9	3.27e - 01 <sub>2.3e-04</sub>	3.26e - 01 <sub>3.7e-03</sub>	3.23e - 01 <sub>3.1e-03</sub>	3.25e - 01 <sub>3.4e-03</sub>

two-objective formulations of the DTLZ and WFG problems. As reference points, we have chosen the ones defined in [6], summarized in Table 1.

To compare the four metaheuristics, we have executed 30 independent runs per configuration and computed the hypervolume [22] as a quality indicator to measure both the convergence and diversity of the obtained Pareto front approximations. As this indicator needs a reference point to be calculated and the Pareto fronts of all the problems are known, we have removed from the reference fronts all the solutions that fall out of the region delimited by the reference points.

We report in the tables summarizing the results the median and interquartile range (IQR) as measures of central tendency and dispersion, respectively. With the aim of providing these results with statistical confidence (in this study,  $p$ -value = 0.05), we have applied Friedman’s ranking and Holm’s post-hoc multi-compare tests [23] to know which algorithms are statistically worse than the control one (i.e., the one with the best ranking).

## 5 Results and Discussion

Table 2 summarizes the obtained results when the indicated reference point is achievable. The cells with dark and light gray background indicate the best and second best hypervolume values, respectively. We observe that SMPSO/RP outperformed the other techniques in 14 out of the 21 evaluated problems, followed by gSMS-EMOA which obtained the best indicator values in 6 problems.

The results yielded when indicating unachievable reference points are included in Table 3. SMPSO/RP is again the best performing algorithm since it

**Table 3.** Median and interquartile range of the hypervolume quality indicator when solving the problems indicating unachievable reference points.

	SMPSO/RP	gSMS-EMOA	gNSGAI	WASF-GA
ZDT1	5.19e - 017.5e-05	5.19e - 012.5e-04	5.14e - 011.1e-03	5.17e - 016.8e-04
ZDT2	4.53e - 013.7e-05	4.53e - 011.2e-04	4.48e - 019.5e-04	4.51e - 014.6e-04
ZDT3	4.91e - 012.9e-05	4.90e - 016.2e-04	4.87e - 012.7e-03	4.88e - 014.3e-04
ZDT4	5.69e - 012.6e-04	5.62e - 015.9e-03	5.58e - 017.4e-03	5.62e - 015.5e-03
ZDT6	4.30e - 014.5e-05	4.25e - 017.9e-04	4.10e - 013.5e-03	4.16e - 012.4e-03
DTLZ1	4.95e - 014.9e-05	4.87e - 012.3e-02	4.80e - 017.9e-02	4.89e - 015.8e-03
DTLZ2	3.10e - 016.9e-05	3.10e - 014.0e-05	3.07e - 014.5e-04	3.09e - 012.1e-05
DTLZ3	3.19e - 012.0e-04	0.00e + 000.0e+00	0.00e + 000.0e+00	1.67e - 012.4e-01
DTLZ4	3.91e - 012.4e-04	3.91e - 015.6e-05	3.89e - 014.4e-04	3.91e - 013.3e-05
DTLZ5	2.66e - 011.9e-04	2.66e - 013.6e-05	2.64e - 013.2e-04	2.66e - 011.6e-05
DTLZ6	3.11e - 011.6e-05	0.00e + 000.0e+00	0.00e + 000.0e+00	1.55e - 015.6e-02
DTLZ7	5.85e - 012.1e-05	5.85e - 014.3e-05	5.83e - 015.2e-04	5.59e - 014.8e-05
WFG1	0.00e + 006.8e-04	3.28e - 021.1e-01	1.66e - 011.3e-01	4.77e - 013.2e-01
WFG2	5.56e - 014.5e-04	5.54e - 012.4e-03	5.54e - 012.3e-03	5.53e - 013.2e-03
WFG3	4.95e - 013.4e-04	4.93e - 011.1e-03	4.90e - 012.2e-03	4.91e - 011.9e-03
WFG4	3.59e - 014.7e-03	3.66e - 015.9e-05	3.62e - 016.8e-04	3.65e - 014.6e-04
WFG5	2.20e - 011.1e-05	2.20e - 013.7e-05	2.18e - 013.8e-04	2.18e - 016.8e-06
WFG6	0.00e + 000.0e+00	0.00e + 000.0e+00	0.00e + 000.0e+00	0.00e + 000.0e+00
WFG7	3.70e - 016.0e-04	3.70e - 011.1e-04	3.67e - 019.6e-04	3.68e - 015.5e-04
WFG8	3.04e - 011.4e-02	2.87e - 013.3e-03	2.87e - 014.7e-03	2.87e - 013.2e-03
WFG9	2.85e - 015.1e-04	2.84e - 013.3e-03	2.81e - 013.3e-03	2.81e - 012.3e-03

obtained the best hypervolume values in 12 out of 21 the problems. Meanwhile, the second best, gSMS-EMOA, only achieved the best results in 7 problems.

**Table 4.** Average Friedman’s rankings with Holm’s Adjusted  $p$ -values (0.05) of compared algorithms when solving the problems indicating achievable (left) and unachievable (right) reference points.

Achievable ( $I_{HV}$ )			Unachievable ( $I_{HV}$ )		
Algorithm	$Fri_{Rank}$	$Holm_{Ap}$	Algorithm	$Fri_{Rank}$	$Holm_{Ap}$
<b>*SMPSO/RP</b>	<b>1.52</b>	-	<b>*SMPSO/RP</b>	<b>1.59</b>	-
gSMS-EMOA	2.09	1.51e-01	gSMS-EMOA	2.16	1.51e-01
WASF-GA	2.76	3.77e-03	WASF-GA	2.71	9.94e-03
gNSGAI	3.62	4.34e-07	gNSGAI	3.52	3.88e-06

As shown in Table 4, SMPSO/RP is the best ranked algorithm according to Friedman’s test for achievable, as well as for unachievable reference points. SMPSO/RP is then established as the control algorithm in the post-hoc Holm tests. The adjusted  $p$ -values ( $Holm_{Ap}$  in Table 4) resulting from these comparisons are lower than the confidence level (0.05) for WASF-GA and gNSGAI, which means that differences between SMPSO/RP and these two algorithms are statistically significant.

To have an insight of the time reductions when running SMPSO/RP in a multi-core system, we have executed it on a machine featuring a quad-core Intel i7 at 2.2. GHz and 16 GB of 1600 MHz DDR3 RAM with hyper-threading

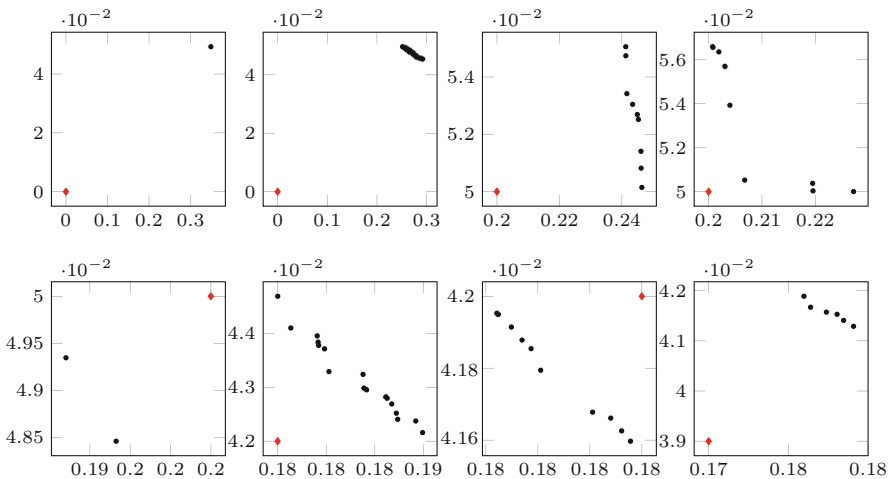
enabled. In particular we have performed these execution using 1, 2, 4 and 8 threads when solving the ZDT4 problem and reference point (0.5,0.5). We have added an idle loop inside the objective functions to increase their computing. The times obtained are 61.5, 45.5, 30.85 and 19.45 s, which mean speed-ups of 1.3, 1.99 and 3.16 with 2, 4, and 8 threads respectively. These speed-ups are expected because, as commented in Sect. 3, only the function evaluations are performed in parallel. Nevertheless, the time reductions are significant and have been achieved with neither major changes in the code nor extra configuration.

## 6 Use Case

This section describes the application of SMPSO/RP to a real-world problem in the field of structural design. The selected problem aims to optimize the design of a cable stayed-bridge having two objectives (total weight and deformation), encompassing 26 decision variables and 68 constraints [24].

We assume here that a civil engineer is interested in finding the region of the front including solutions with the lowest weights. Without any initial knowledge regarding the weight of different solutions, the starting reference point for the civil engineer is set to (0.0,0.0) and he/she interactively changes it during the search as information about different computed structures is obtained. A possible execution is shown in Fig. 4 and is described next:

1. Generation 14: Reference point: (0.0,0.0). The algorithm is looking for a first feasible solution.
2. Generation 64: SMPSO/RP has found a feasible region and a set of non-dominated solutions.



**Fig. 4.** Example of guiding the search in the structural design problem. Each plot depicts the front at generation 14, 64, 104, 208, 278, 465, 534, and 599, respectively. The reference point changes from (0.0, 0.0) to (0.2, 0.05), (0.18, 0.042), and (0.17, 0.039). The x-axis represents the weight and the y-axis the deformation.



3. Generation 104. The reference point is changed to  $(0.2, 0.05)$ , which currently is unfeasible.
4. Generation 208. The front is evolving towards the current reference point.
5. Generation 278. The current reference point is feasible and the computed front of solutions is spreading.
6. Generation 465. The reference point is changed to  $(0.18, 0.042)$ , which is currently unfeasible.
7. Generation 534. The current reference point is feasible and the computed front of solutions is spreading.
8. Generation 599. The reference point is changed to  $(0.17, 0.039)$ , which is currently unfeasible. At this stage, the engineer is satisfied with the solutions obtained and the optimization process is stopped.

## 7 Conclusions and Future Research Lines

We introduced SMPSO/RP, an extension of the SMPSO incorporating a preference articulation mechanism based on indicating reference points. Our approach allows changing the reference points interactively and evaluating particles of the swarm in parallel. SMPSO/RP is implemented within the jMetal framework and its source code is freely available.

We have compared our proposal against three other related algorithms on a benchmark composed of 21 problems. Our results indicate that SMPSO/RP achieved the best overall performance when indicating both achievable and unachievable reference points. We have also measured the time reductions that have been achieved when running the algorithm in a multi-core processor platform.

As a line of future work, we are working on adapting SMPSO/RP to efficiently deal with many-objective problems. This implies to rethink the archiving policy and derive novel Pareto density metrics suitable for such problem formulations.

**Acknowledgement.** This work has been partially funded by Grants TIN2017-86049-R (Spanish Ministry of Education and Science) and P12-TIC-1519 (Plan Andaluz de Investigación, Desarrollo e Innovación). Cristóbal Barba-González is supported by Grant BES-2015-072209 (Spanish Ministry of Economy and Competitiveness). José García-Nieto is the recipient of a Post-Doctoral fellowship of “Captación de Talento para la Investigación” Plan Propio at Universidad de Málaga. Javier Del Ser thanks the Basque Government for its funding support through the EMAITEK program. Carlos A. Coello Coello is supported by CONACyT project no. 221551.

## References

1. Coello Coello, C., Lamont, G., van Veldhuizen, D.: Multi-Objective Optimization Using Evolutionary Algorithms, 2nd edn. Wiley, Hoboken (2007)
2. Coello Coello, C.: Handling preferences in evolutionary multiobjective optimization: a survey. In: Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, vol. 1, pp. 30–37 (2000)

3. Nebro, A., Durillo, J., García-Nieto, J., Coello Coello, C., Luna, F., Alba, E.: SMPSO: a new PSO-based metaheuristic for multi-objective optimization. In: IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, MCDM 2009, pp. 66–73. IEEE Press (2009)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
5. Durillo, J.J., Nebro, A.J.: jMetal: a Java framework for multi-objective optimization. *Adv. Eng. Softw.* **42**(10), 760–771 (2011)
6. Ruiz, A., Saborido, R., Luque, M.: A preference-based evolutionary algorithm for multiobjective optimization: the weighting achievement scalarizing function genetic algorithm. *J. Glob. Optim.* **62**(1), 101–129 (2015)
7. Branke, J.: MCDA and multiobjective evolutionary algorithms. In: Greco, S., Ehrgott, M., Figueira, J. (eds.) *Multiple Criteria Decision Analysis*. ISOR, vol. 233, pp. 977–1008. Springer, New York (2016). [https://doi.org/10.1007/978-1-4939-3094-4\\_23](https://doi.org/10.1007/978-1-4939-3094-4_23)
8. Li, L., Wang, Y., Trautmann, H., Jing, N., Emmerich, M.: Multiobjective evolutionary algorithms based on target region preferences. *Swarm Evol. Comput.* **40**, 196–215 (2018)
9. Wierzbicki, A.P.: Reference point approaches. In: Gal, T., Stewart, T.J., Hanne, T. (eds.) *Multicriteria Decision Making*. ISOR, vol. 21, pp. 237–275. Springer, Boston (1999). [https://doi.org/10.1007/978-1-4615-5025-9\\_9](https://doi.org/10.1007/978-1-4615-5025-9_9)
10. Molina, J., Santana, L., Hernández-Díaz, A., Coello Coello, C., Caballero, R.: g-dominance: Reference point based dominance for multiobjective metaheuristics. *Eur. J. Oper. Res.* **197**(2), 685–692 (2009)
11. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948 (1995)
12. Sierra, M.R., Coello Coello, C.A.: Improving PSO-based multi-objective optimization using crowding, mutation and  $\epsilon$ -dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) *EMO 2005*. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-31880-4\\_35](https://doi.org/10.1007/978-3-540-31880-4_35)
13. Durillo, J., Nebro, A., Coello Coello, C., García-Nieto, J., Luna, F., Alba, E.: A study of multiobjective metaheuristics when solving parameter scalable problems. *IEEE Trans. Evol. Comput.* **14**(4), 618–635 (2010)
14. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **6**(1), 58–73 (2002)
15. Nebro, A.J., Durillo, J.J., Vergne, M.: Redesigning the jMetal multi-objective optimization framework. In: *Proceedings of the Companion of the Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 1093–1100 (2015)
16. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
17. Deb, K., Sundar, J., Ubay, B., Chaudhuri, S.: Reference point based multi-objective optimization using evolutionary algorithm. *Int. J. Comput. Intell. Res.* **2**(6), 273–286 (2006)
18. Allmendinger, R., Li, X., Branke, J.: Reference point-based particle swarm optimization using a steady-state approach. In: Li, X., et al. (eds.) *SEAL 2008*. LNCS, vol. 5361, pp. 200–209. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-89694-4\\_21](https://doi.org/10.1007/978-3-540-89694-4_21)
19. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. *Evol. Comput.* **8**(2), 173–195 (2000)

20. 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*. AI&KP, pp. 105–145. Springer, London (2005). [https://doi.org/10.1007/1-84628-137-7\\_6](https://doi.org/10.1007/1-84628-137-7_6)
21. Huband, S., Barone, L., While, L., Hingston, P.: A scalable multi-objective test problem toolkit. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) *EMO 2005*. LNCS, vol. 3410, pp. 280–295. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-31880-4\\_20](https://doi.org/10.1007/978-3-540-31880-4_20)
22. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Trans. Evol. Comput.* **3**(4), 257–271 (1999)
23. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **1**(1), 3–18 (2011)
24. Zavala, G., Nebro, A.J., Luna, F., Coello Coello, C.: Structural design using multi-objective metaheuristics. Comparative study and application to a real-world problem. *Struct. Multidiscip. Optim.* **53**(3), 545–566 (2016)



# Improving 1by1EA to Handle Various Shapes of Pareto Fronts

Yiping Liu<sup>1</sup>, Hisao Ishibuchi<sup>2</sup>, Yusuke Nojima<sup>1</sup>(✉), Naoki Masuyama<sup>1</sup>,  
and Ke Shang<sup>2</sup>

<sup>1</sup> Department of Computer Science and Intelligent Systems,  
Graduate School of Engineering, Osaka Prefecture University,  
Sakai, Osaka 599-8531, Japan

yiping01iu@gmail.com, {nojima,masuyama}@cs.osakafu-u.ac.jp

<sup>2</sup> Department of Computer Science and Engineering,  
Southern University of Science and Technology,  
Shenzhen 518055, Guangdong, China  
hisao@sustc.edu.cn, kshang@foxmail.com

**Abstract.** 1by1EA is a competitive method among existing many-objective evolutionary algorithms. However, we find that it may fail to find boundary solutions depending on the Pareto front shape. In this study, we present an improved version of 1by1EA, named 1by1EA-II, to enhance the flexibility in handling various shapes of Pareto fronts. In 1by1EA-II, the Chebyshev distances from a solution to the nadir and ideal points are alternately employed as two convergence indicators. Using the first convergence indicator, boundary solutions are preferred for a wide spread in the objective space. With the other convergence indicator, non-boundary solutions are preferred to promote diversity. We empirically compare the proposed 1by1EA-II with its original version as well as four other state-of-the-art algorithms on DTLZ and Minus-DTLZ test problems. The results show that 1by1EA-II is the most flexible algorithm.

**Keywords:** Many-objective evolutionary computation  
Pareto front shape · Convergence · Diversity

## 1 Introduction

There exist a large number of multi-objective optimization problems (MOPs) in real-world applications. The conflict of objectives implies that there is no single optimal solution to an MOP, rather a set of trade-off solutions, called the Pareto optimal solution set (PS). The image of PS in the objective space is referred to as the Pareto front (PF). Without loss of generality, an MOP can be mathematically expressed as follows:

$$\begin{aligned} \min \mathbf{f}(\mathbf{x}) &= \min(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x})) \\ \text{s.t. } \mathbf{x} &\in S \subset \mathbf{R}^n \end{aligned} \quad (1)$$

where  $\mathbf{x} = (x_1, \dots, x_n)$  represents an  $n$ -dimensional decision vector in space  $S$ ,  $f_m(\mathbf{x})$ ,  $m = 1, \dots, M$ , is the  $m$ -th objective to be minimized, and  $M$  is the number of objectives. When  $M > 3$ , the problem in Eq. (1) is referred to as a many-objective optimization problem (MaOP).

Multi-Objective Evolutionary Algorithms (MOEAs) are widely applied to solve MOPs, where the Pareto dominance-based ones are most popular, such as Non-dominated Sorting Genetic Algorithm II (NSGA-II) [4] and Strength Pareto Evolutionary Algorithm 2 (SPEA2) [17]. However, their performance generally deteriorates appreciably for MaOPs. One main reason is the low efficiency of the Pareto dominance-based selection strategy in the high-dimensional objective space.

To address this issue, various MOEAs aiming at solving MaOPs have been developed in recent years. Generally, they can be categorized into the following three categories: (1) improved Pareto dominance-based methods, e.g., SPEA2 with shift-based density estimation (SDE) [8] and NSGA-III [3]; (2) decomposition-based methods, e.g., MOEA/D [15] and Reference-Vector-guided Evolutionary Algorithm (RVEA) [2]; (3) indicator-based methods, e.g., Hypervolume Estimation algorithm (HypE). Besides, there are a number of novel methods that have not been categorized, e.g., Grid-based Evolutionary Algorithm (GrEA) [14], Knee point driven Evolutionary Algorithm (KnEA) [16], Bi-Goal Evolutionary approach (BiGE) [9], Reference Points-based Evolutionary Algorithm (RPEA) [11], and One-by-One selection-based Evolutionary Algorithm (1by1EA) [10].

These MOEAs often show encouraging performance on widely used benchmarks such as DTLZ [5] and WFG [6]. However, their performance may strongly depend on the PF shapes. For example, by simply inverting the PF shapes of DTLZ, the performance of a decomposition-based method noticeably degrades [7]. Similarly, NSGA-III and MOMBI-II which share the concept of decomposition would also have the issue. For another instance, the performance of some methods like GrEA is very sensitive to the parameter settings, and it is difficult to tune the parameters according to the PF shapes. The real-world optimization problems usually have various shapes of PFs. Therefore, developing more flexible MOEAs is a must, where improving the flexibility of existing state-of-the-art MOEAs is very promising.

In this study, we improve 1by1EA's ability in solving MaOPs with various shapes of PFs. 1by1EA is very competitive among existing many-objective optimizers. As shown by our computational experiments later in this paper, 1by1EA has high search ability on DTLZ (which is higher than other many-objective optimizers such as NSGA-III, MOEA/D, BiGE and KnEA). 1by1EA adopts not only a one-by-one selection strategy to well balance the convergence and the diversity of solutions, but also a corner solution preserving strategy for a wide spread. However, we find that 1by1EA may not perform well when the corner solutions are difficult to be located on a PF. In this study, we present an improved version of 1by1EA, named 1by1EA-II. It alternately employs two convergence indicators to search the boundary and non-boundary solutions and is more flexible than its original version.

The remainder of this paper is organized as follows. In Sect. 2, 1by1EA is first briefly introduced and the motivation of this work is elaborated. The proposed 1by1EA-II is then described in detail in Sect. 3. Section 4 presents experimental results and discussions. Section 5 concludes the paper and provides future research directions.

## 2 Preliminaries

In this section, we first briefly introduce 1by1EA [10] and then elaborate the motivation of this work.

### 2.1 A Brief Introduction to 1by1EA

The general framework of 1by1EA is similar to standard generational evolutionary algorithms, whereas its environmental selection makes it special.

Assume to solve the problem in Eq. (1) using 1by1EA. Before the environmental selection, the convergence and distribution indicators of each candidate solution are calculated. Please refer to the original study for the examples of these indicators [10]. The convergence indicator is usually a scalarizing function aggregating all objective functions, such as the sum of all objective functions or the Euclidean distance between the solution and the ideal (nadir) point. Note that we estimate the ideal (nadir) point in terms of the minimum (maximum) objective values among obtained non-dominated solutions in this study. The convergence indicator can provide an extremely large selection pressure towards the PF. The general formulation of the convergence indicator can be summarized as follows:

$$c(\mathbf{x}) = \text{agg}(f_1(\mathbf{x}), \dots, f_M(\mathbf{x})). \quad (2)$$

The distribution indicator is the cosine similarity between the solution and each of the others. It can efficiently reduce the number of dominance resistant solutions.

Next,  $M$  corner solutions are selected to estimate the boundary of the PF. The  $m$ th corner solution  $\mathbf{x}_m^{\text{corner}}$  is obtained by the following method:

$$\mathbf{x}_m^{\text{corner}} = \arg \min_{\mathbf{x}_i \in Q} c_m(\mathbf{x}_i), m = 1, \dots, M, \quad (3)$$

where  $c_m(\mathbf{x}_i) = \text{agg}(f_1(\mathbf{x}), \dots, f_{m-1}(\mathbf{x}), f_{m+1}(\mathbf{x}), \dots, f_M(\mathbf{x}))$ , and  $Q$  is the current population.

Finally, the one-by-one selection strategy is applied. It consists of the two important steps. In the first step, only one solution with the best value of the convergence indicator is selected, focusing on the convergence. In the second step, solutions close to the one selected in the first step are de-emphasized according to the distribution indicator, thus maintaining the diversity of the population. By repeating the above two steps, a solution set with good convergence and diversity can be obtained.

1by1EA has been demonstrated to be a competitive many-objective optimizer. However, we find that 1by1EA is not flexible enough due to the corner solution preserving strategy. The motivation of improving 1by1EA is elaborated in the next subsection.

### 2.2 Motivation

As reported in [7] recently, a number of newly proposed decomposition-based algorithms seem to be overspecialized for the popular benchmarks like DTLZ. In [7], the minus version of DTLZ (denoted as Minus-DTLZ) is presented, where the PF shapes are inverted from those of DTLZ. Figure 1 shows the true PFs of DTLZ2 and Minus-DTLZ2 with three objectives for intuitive understanding.

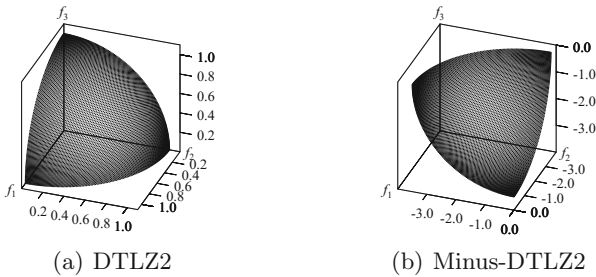
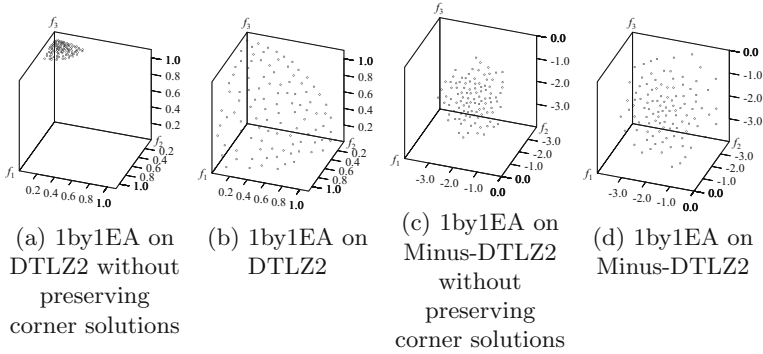


Fig. 1. The true PFs of DTLZ2 and Minus-DTLZ2 with three objectives.

The performance of decomposition-based algorithms appreciably deteriorates on Minus-DTLZ merely because the PF shapes of Minus-DTLZ are different from those of DTLZ. Some recent researches on using two reference vector sets have addressed this issue [1, 13]. This inspires us to investigate the behaviors of some non-decomposition-based algorithms like 1by1EA when handling various shapes of PFs.

The corner solution preserving strategy plays an important role in 1by1EA. Figure 2 presents the solution sets obtained by 1by1EA with and without preserving corner solutions on DTLZ2 and Minus-DTLZ2 with three objectives in a typical run, where the Euclidean distance between a solution and the ideal point is chosen as the convergence indicator. Note that in this study, the inverted generational distance (IGD) [18] of the solution set obtained in the typical run is the nearest to the average IGD over 40 runs.

We can see from Fig. 2 that preserving corner solutions can lead to the solution sets widely spread in the objective space both on DTLZ2 and Minus-DTLZ2. The solution set in Fig. 2(b) approximates well to the true PF in Fig. 1(a). However, the solution set in Fig. 2(d) fails to cover some boundary regions, comparing to the true PF in Fig. 1(b). The reason is that the corner solutions to DTLZ2 (i.e., (1, 0, 0), (0, 1, 0), (0, 0, 1)) can be easily located by Eq. (3) (i.e., minimizing



**Fig. 2.** The solution sets obtained on DTLZ2 and Minus-DTLZ2 with three objectives.

$f_2 + f_3$ ,  $f_3 + f_1$ , and  $f_1 + f_2$ , respectively), whereas those to Minus-DTLZ2 (i.e.,  $(-3.5, 0, 0)$ ,  $(0, -3.5, 0)$ ,  $(0, 0, -3.5)$ ) cannot. Of course, we can use another method (i.e., by minimizing  $f_1$ ,  $f_2$ , and  $f_3$ , respectively) to obtain the corner solutions of Minus-DTLZ2. However, for real-world optimization problems, we usually cannot obtain the *a priori* knowledge of corner points and apply a proper method to locate all of them. Furthermore, even if we use both of the above-mentioned methods, the shape of a PF could be too complex to locate the corner solutions.

From these observations, we notice that if the corner solution can be precisely located, 1by1EA performs perfectly; otherwise, it may miss some boundary regions on a PF. This indicates that the performance of 1by1EA also depends on the shapes of PFs.

In view of this, we present an improved version of 1by1EA, named 1by1EA-II, to enhance its flexibility in handling various shapes of PFs. In 1by1EA-II, the corner solutions are no longer needed to be preserved. The diversity of solutions are promoted by alternately employing two different convergence indicators. The details of 1by1EA-II are described in the next section.

### 3 1by1EA-II

The difference between 1by1EA and 1by1EA-II is that in the environmental selection of 1by1EA-II, the corner solutions are no longer preserved by Eq. (3), and two convergence indicators are alternately employed to select the solution with best convergence performance. That is, after selecting a solution according to a convergence indicator, the next solution to be selected is based on the other convergence indicator. Please refer to the original study of 1by1EA for the environmental selection procedure [10]. We describe the two convergence indicators used in 1by1EA-II in the following parts.



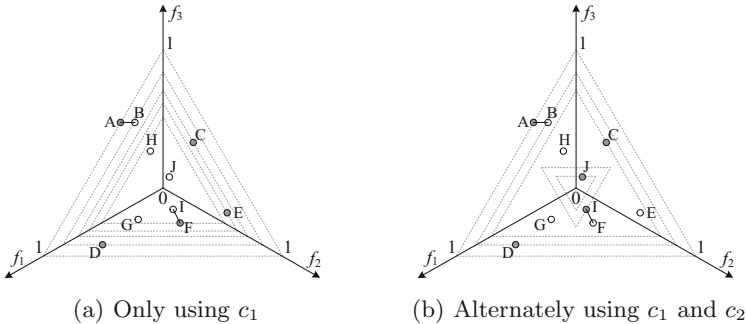
The first convergence indicator adopted in 1by1EA-II is the Chebyshev distance between a solution and the nadir point. It is formulated as follows:

$$c^{\text{CdN}}(\mathbf{x}) = \max_{1 \leq m \leq M} |f_m(\mathbf{x}) - z_m^{\text{nad}}|, \tag{4}$$

where  $\mathbf{z}^{\text{nad}} = (z_1^{\text{nad}}, \dots, z_M^{\text{nad}})$  is the nadir point. Note that there is no weight in the convergence indicators, since all the objectives are equally considered in this study. The solution farthest from the nadir point is supposed to have the best convergence performance. However, since the nadir point is estimated based on the obtained non-dominated solutions, a solution dominated by the estimated nadir point may be better than others according to Eq. (4). This situation should be avoided. In addition, we want to minimize the convergence indicator. Therefore, we modify Eq. (4) into the following formulation:

$$c_1(\mathbf{x}) = \min_{1 \leq m \leq M} (f_m(\mathbf{x}) - z_m^{\text{nad}}). \tag{5}$$

By minimizing  $c_1$  in 1by1EA-II, the boundary solutions (including the corner solutions) are preferred (no matter the Pareto front is convex or concave). Figure 3(a) presents an illustration of the solution selection procedure only using  $c_1$ .

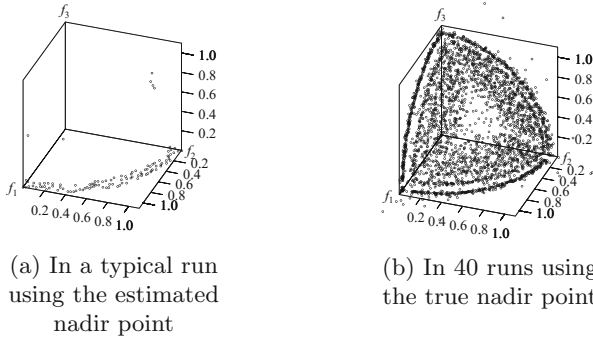


**Fig. 3.** Solution selection procedure in 1by1EA-II.

In Fig. 3, assume (0, 0, 0) and (1, 1, 1) are the ideal and nadir points, respectively. Dots A-J are candidate solutions, and we want to select five of them into the next generation. The dashed triangles are the intersections of the contour lines of  $c_1$  and the hyperplan defined by  $f_1 + f_2 + f_3 = 1$  (note that the solutions are not necessarily on the hyperplan). Solutions connected with each other are in the other’s niche according to the diversity indicator. As can be seen from Fig. 3(a), the boundary solution A has the minimum value of  $c_1$  and all the other solutions are within the corresponding dashed triangle of A. Therefore, A is selected first. Then B is de-emphasized since it is too close to A. Next, C, D, E, and F are selected one by one due to their minimum  $c_1$  values among the

rest. From this illustration, we can observe that boundary solutions are always preferred by minimizing  $c_1$  no matter what is the shape of a PF. Consequently, preserving corner solutions is unnecessary in 1by1EA-II.

However, employing  $c_1$  as the only convergence indicator may result in two issues. The first issue is that it may lead the population into a partial region of the PF, since the estimated nadir point is usually quite different from the true one at the early stage of evolution. Minimizing  $c_1$  with an incorrect nadir point will result in solutions located in partial regions. Conversely, the nadir point in the next generation could be estimated more incorrectly by these solutions. Figure 4(a) shows the obtained solution set on DTLZ2 with three objectives in a typical run when  $c_1$  is employed as the only convergence indicator. We can see that most solutions locate in a small region. The other issue is that even if we use the true nadir point, the solutions are more likely to locate in the boundary region. Figure 4(b) shows all the solution sets obtained in 40 runs where the true nadir point is used in  $c_1$ . We can observe that the solutions in the boundary region are denser than those in the central region.



**Fig. 4.** The solution set obtained on DTLZ2 with three objectives when  $c_1$  is employed as the only convergence indicator.

In view of this, we employ the Chebyshev distance between a solution and the ideal point as the other convergence indicator, which is formulated as follows:

$$c_2(\mathbf{x}) = \max_{1 \leq m \leq M} |f_m(\mathbf{x}) - z_m^*|, \quad (6)$$

where  $\mathbf{z}^* = (z_1^*, \dots, z_M^*)$  is the ideal point. In contrast to minimizing  $c_1$ , non-boundary solutions are preferred when minimizing  $c_2$ . Alternately employing  $c_1$  and  $c_2$  is helpful to promote diversity, since both non-boundary and boundary solutions have a chance to be selected. Let us see Fig. 3(b) as an example, where the dashed inverted triangles are the intersections of the contour lines of  $c_2$  and the hyperplan defined by  $f_1 + f_2 + f_3 = 1$ . The solution J has the minimum value of  $c_2$  and all the other solutions are outside the corresponding dashed inverted triangle of J. In this case, A, J, C, I, and D are selected one by one, and B and F

are de-emphasized after selecting A and I, respectively. There are more solutions in the central region in Fig. 3(b) than that in Fig. 3(a). In addition, solutions selected by alternately using  $c_1$  and  $c_2$  have a much lower chance to converge into a partial region, and the nadir point could be estimated more precisely.

By the cooperation among the above-mentioned two convergence indicators and the distribution indicator, the one-by-one selection strategy in 1by1EA-II is expected to locate the boundary solutions on a PF and maintain a good diversity within the boundary.

## 4 Experiments and Discussions

In this section, we empirically evaluate and discuss the performance of 1by1EA-II by comparing it with 1by1EA, NSGA-III, MOEA/D, BiGE and KnEA. DTLZ1 to 4 and Minus-DTLZ1 to 4 are chosen as test problems. We consider these test problems with 3, 4, 6, and 8 objectives. The number of variables  $n$  is set to  $M + 4$  for DTLZ1 and Minus-DTLZ1, and  $M + 9$  for the other test problems ( $M$  is the number of objectives). NSGA-III and MOEA/D are supposed to be overspecialized for DTLZ test problems according to [7]. No study has shown that BiGE and KnEA are overspecialized so far.

For all compared algorithms, simulated binary crossover and polynomial mutation are used as the crossover and mutation operators, with both distribution indexes being set to 20. The crossover and mutation probabilities are 1.0 and  $1/n$ , respectively. The population size  $N$  is set to 105, 120, 132 and 156 when  $M$  is 3, 4, 6, and 8, respectively. In 1by1EA, the Euclidean distance between a solution and the ideal point is chosen as the convergence indicator. In MOEA/D, the PBI method with  $\theta = 5$  is adopted. In KnEA,  $T$  is set to 0.5. Each algorithm is run for 40 times on each test problem, where the termination condition is set to 600 generations for DTLZ3 and Minus-DTLZ3, and 300 generations for the other test problems. The source codes of 1by1EA and 1by1EA-II can be downloaded from <https://github.com/yiping0liu>. All the other compared algorithms are implemented by PlatEMO [12].

Table 1 lists the average values of IGD over 40 runs in gray scale, where a darker tone corresponds to a larger average value of IGD. Note that in this study the reference points for IGD calculation are uniformly sampled on a true PF, and the number of reference points is around  $10^4$ . In Table 1, “Rank<sup>1</sup>”, “Rank<sup>-1</sup>”, and “Rank<sup>all</sup>” denote the average ranks of each algorithm according to the average IGD values on DTLZ, Minus-DTLZ, and all the test problems, respectively. DTLZ $n$ - $m$  (DTLZ $n$ <sup>-1- $m$</sup> ) denotes DTLZ $n$  (Minus-DTLZ $n$ ) with  $m$  objectives. “+” indicates that the result is significantly different from that of 1by1EA-II by Wilcoxon’s rank sum test where the null hypothesis is rejected at a significant level of 0.05. “+”, “-”, and “=” indicate the number of test problems where 1by1EA-II shows significantly better, worse, and similar performance, respectively.

From Table 1, we can see that 1by1EA-II, 1by1EA, NSGA-III, and MOEA/D generally achieve satisfactory results on DTLZ, where 1by1EA obtains the best

**Table 1.** Average IGD obtained by different algorithms.

IGD	1by1EA-II	1by1EA	NSGA-III	MOEA/D	BiGE	KnEA
DTLZ1-3	5.449E-2	4.834E-2 †	1.940E-2 †	1.943E-2 †	3.507E-2 †	5.564E-2 †
DTLZ1-4	4.599E-2	4.639E-2	4.186E-2 †	4.168E-2 †	8.023E-2 †	1.339E-1 †
DTLZ1-6	8.035E-2	8.406E-2 †	1.070E-1 †	8.048E-2	2.005E-1 †	2.432E-1 †
DTLZ1-8	1.056E-1	1.116E-1 †	1.610E-1 †	8.979E-2 †	3.508E-1 †	9.064E-1 †
DTLZ2-3	7.544E-2	5.143E-2 †	5.032E-2 †	5.031E-2 †	7.764E-2	6.645E-2
DTLZ2-4	1.316E-1	1.184E-1 †	1.212E-1 †	1.212E-1 †	1.640E-1 †	1.420E-1 †
DTLZ2-6	2.623E-1	2.499E-1 †	2.577E-1 †	2.558E-1 †	3.140E-1 †	2.809E-1 †
DTLZ2-8	3.692E-1	3.492E-1 †	3.619E-1	3.156E-1 †	4.006E-1 †	3.837E-1 †
DTLZ3-3	8.314E-2	5.053E-2 †	5.151E-2 †	5.163E-2 †	9.779E-2 †	1.135E-1 †
DTLZ3-4	1.345E-1	1.246E-1	1.232E-1	1.560E-1 †	2.709E-1 †	2.792E-1 †
DTLZ3-6	2.702E-1	2.591E-1 †	5.501E-1 †	4.295E-1 †	1.734E+0 †	1.651E+0 †
DTLZ3-8	3.839E-1	3.604E-1 †	1.267E+0 †	5.105E-1 †	1.465E+1 †	6.748E+1 †
DTLZ4-3	9.048E-2	6.865E-2 †	9.944E-2 †	3.826E-1 †	1.084E-1 †	9.513E-2 †
DTLZ4-4	1.656E-1	1.205E-1 †	1.764E-1 †	4.217E-1 †	1.659E-1	1.705E-1 †
DTLZ4-6	2.717E-1	2.835E-1	2.854E-1 †	5.511E-1 †	3.112E-1 †	3.105E-1 †
DTLZ4-8	3.763E-1	3.522E-1 †	3.452E-1 †	6.229E-1 †	3.980E-1 †	3.753E-1
Rank <sup>1</sup>	3.0	2.0	2.9	3.1	5.1	5.0
+/-/=	\	2/11/3	7/7/2	7/8/1	13/1/2	14/0/2
DTLZ1 <sup>-1</sup> -3	2.376E+1	4.686E+1 †	3.056E+1 †	3.850E+1 †	2.980E+1 †	4.453E+1 †
DTLZ1 <sup>-1</sup> -4	4.651E+1	7.525E+1 †	7.991E+1 †	6.203E+1 †	5.555E+1 †	6.800E+1 †
DTLZ1 <sup>-1</sup> -6	8.708E+1	1.540E+2 †	1.256E+2 †	1.661E+2 †	9.844E+1 †	9.990E+1 †
DTLZ1 <sup>-1</sup> -8	1.666E+2	2.526E+2 †	1.544E+2 †	2.645E+2 †	1.244E+2 †	1.199E+2 †
DTLZ2 <sup>-1</sup> -3	2.345E-1	3.358E-1 †	2.390E-1	2.432E-1 †	3.262E-1 †	2.368E-1
DTLZ2 <sup>-1</sup> -4	4.589E-1	7.004E-1 †	5.178E-1 †	5.622E-1 †	6.616E-1 †	5.704E-1 †
DTLZ2 <sup>-1</sup> -6	9.169E-1	1.365E+0 †	1.121E+0 †	1.241E+0 †	1.332E+0 †	1.225E+0 †
DTLZ2 <sup>-1</sup> -8	1.367E+0	1.886E+0 †	1.684E+0 †	2.125E+0 †	1.879E+0 †	1.570E+0
DTLZ3 <sup>-1</sup> -3	1.466E+2	2.615E+2 †	1.517E+2	1.532E+2 †	1.993E+2 †	1.648E+2 †
DTLZ3 <sup>-1</sup> -4	2.884E+2	4.496E+2 †	3.333E+2 †	3.534E+2 †	4.167E+2 †	3.976E+2 †
DTLZ3 <sup>-1</sup> -6	5.717E+2	8.651E+2 †	7.039E+2 †	7.769E+2 †	8.412E+2 †	7.371E+2 †
DTLZ3 <sup>-1</sup> -8	7.744E+2	1.173E+3 †	1.044E+3 †	1.333E+3 †	1.185E+3 †	9.401E+2 †
DTLZ4 <sup>-1</sup> -3	2.214E-1	5.638E-1 †	2.348E-1 †	8.504E-1 †	2.676E-1 †	2.351E-1 †
DTLZ4 <sup>-1</sup> -4	5.669E-1	5.781E-1	5.160E-1 †	9.226E-1 †	5.544E-1	5.477E-1 †
DTLZ4 <sup>-1</sup> -6	1.109E+0	1.146E+0	1.049E+0	1.848E+0 †	1.129E+0	1.013E+0 †
DTLZ4 <sup>-1</sup> -8	1.454E+0	1.559E+0 †	1.403E+0 †	2.423E+0 †	1.581E+0 †	1.328E+0 †
Rank <sup>-1</sup>	1.8	5.3	2.6	4.8	3.9	2.7
+/-/=	\	14/0/2	10/3/3	16/0/0	13/1/2	10/4/2
Rank <sup>all</sup>	2.4	3.7	2.8	3.9	4.5	3.8
+/-/=	\	16/11/5	17/10/5	23/8/1	26/2/4	24/4/4

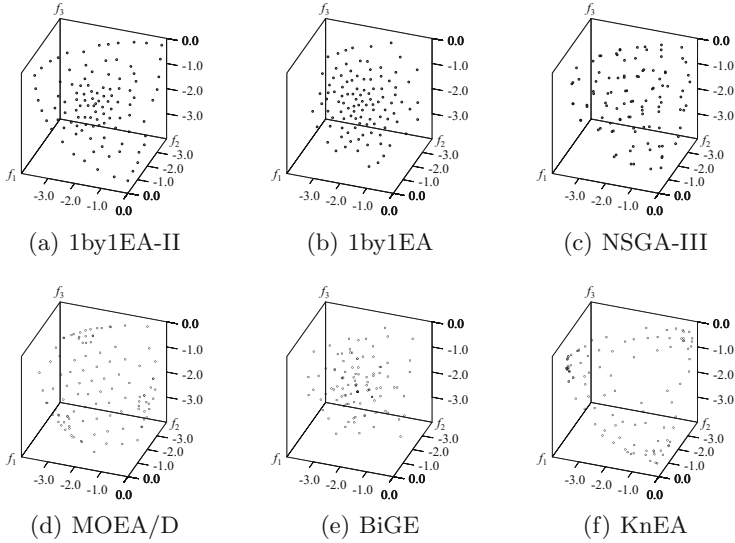
“Rank<sup>1</sup>”. The “Rank<sup>1</sup>” of 1by1EA-II is very close to those of NSGA-III and MOEA/D, which are verified to have strength in solving DTLZ. This indicates that 1by1EA-II is very effective on solving these problems. However, 1by1EA-II does not perform as well as 1by1EA, NSGA-III, and MOEA/D on DTLZ1,

DTLZ2 and DTLZ3 with three objectives. MOEA/D performs poorly on DTLZ4, since DTLZ4 has a bias PF, which results in the failure of the PBI method in maintaining diversity in the objective space. BiGE and KnEA obtain larger values of “Rank<sup>1</sup>” than the others, which suggests that they do not achieve appealing results, comparing to the algorithms that are supposed to overspecialized for DTLZ. However, they show relatively good performance on DTLZ4.

For Minus-DTLZ, 1by1EA-II outperforms the others on most test problems and obtains the best “Rank<sup>-1</sup>”. On the contrary, 1by1EA achieves poor IGD values on these problems and obtains the worst “Rank<sup>-1</sup>”. Comparing the results obtained by 1by1EA on DTLZ and Minus-DTLZ, we can notice that the performance of 1by1EA strongly depends on the PF shapes. The distribution of reference vectors (points) used in both MOEA/D and NSGA-III is inconsistent with the PF shapes of Minus-DTLZ. The performance of MOEA/D generally degrades appreciably on Minus-DTLZ, whereas the results of NSGA-III on Minus-DTLZ are still acceptable. The reason is that every reference vector (point) has to be assigned a solution in MOEA/D while it does not in NSGA-III. Moreover, in NSGA-III, multiple solutions can be clustered to one reference point, and then the reference points within the region of PF are assigned more solutions than those outside the region of PF. Consequently, the diversity of the solution set can be well maintained in NSGA-III. This observation indicates that the performance of NSGA-III is less sensitive to the PF shape than MOEA/D. The average ranks obtained by BiGE and KnEA on Minus-DTLZ are better than those on DTLZ. They achieve encouraging results on some Minus-DTLZ test problems. Theoretically, both of them are not designed to solve particular problems, however, they seem to perform better on Minus-DTLZ when comparing to the other algorithms.

As a whole, 1by1EA-II achieves the best overall performance among the compared algorithms, since it obtains the best value of “Rank<sup>all</sup>”. Besides, both “Rank<sup>1</sup>” and “Rank<sup>-1</sup>” obtained by 1by1EA-II are satisfying. Therefore, we can conclude that 1by1EA-II is most flexible among the compared algorithm on DTLZ and Minus-DTLZ test problems.

To visually demonstrate the superiority of 1by1EA-II over the other algorithms, we show the solution sets obtained by the compared algorithms in a typical run in Fig. 5. Due to space limits, only results on Minus-DTLZ2 with three objectives are presented. As can be seen from Fig. 5, 1by1EA-II can locate the boundary solutions according to the PF shape and maintain good diversity within the boundary. 1by1EA behaves as we have explained in Subsect. 2.2. For NSGA-III, there are many solutions very close to another one, since multiple solutions are clustered to one reference point. In MOEA/D, the reference vectors outside the true PF are assigned to solutions that are close to those inside the true PF, and thus the solutions obtained by MOEA/D are denser in the certain regions. The solutions obtained by BiGE fail to trace the true PF shape, and some overlap with each other. KnEA can find the boundary solutions, however, most solutions concentrate on the corner regions.



**Fig. 5.** The solution sets obtained by different algorithms on Minus-DTLZ2 with three objectives in a typical run.

## 5 Conclusions

In this paper, we presented an improved version of 1by1EA, named 1by1EA-II. 1by1EA-II has two distinct features. The first is that it does not preserve corner solution in the environmental selection. The other is that it alternately employs two different convergence indicators, which are the Chebyshev distances from a solution to the nadir and ideal points, respectively. By using these two convergence indicators, 1by1EA-II has an ability in achieving a well-distributed solution set according to the PF shape.

To demonstrate the effectiveness of 1by1EA-II, we tested it on DTLZ and Minus-DTLZ test problems in comparison with five state-of-the-art algorithms, namely, 1by1EA, NSGA-III, MOEA/D, BiGE, and KnEA. The experimental results demonstrated that 1by1EA-II is a competitive and flexible method among the chosen algorithms.

To further investigate and improve the flexibility of 1by1EA-II, we will apply it to optimization problems with other shapes of PFs in the future work. In addition, based on the boundary locating technique in 1by1EA-II, developing a reference vector generation method for decomposition-based algorithms is of great interest.

**Acknowledgments.** This work was supported by the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284).

## References

1. Bhattacharjee, K.S., Singh, H.K., Ray, T., Zhang, Q.: Decomposition based evolutionary algorithm with a dual set of reference vectors. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 105–112. IEEE (2017)
2. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **20**(5), 773–791 (2016)
3. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part I: solving problems with box constraints. *IEEE Trans. Evol. Comput.* **18**(4), 577–601 (2013)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
5. 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*, pp. 105–145. Springer, London (2005). [https://doi.org/10.1007/1-84628-137-7\\_6](https://doi.org/10.1007/1-84628-137-7_6)
6. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **10**(5), 477–506 (2006)
7. Ishibuchi, H., Setoguchi, Y., Masuda, H., Nojima, Y.: Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Trans. Evol. Comput.* **21**(2), 169–190 (2017)
8. Li, M., Yang, S., Liu, X.: Shift-based density estimation for Pareto-based algorithms in many-objective optimization. *IEEE Trans. Evol. Comput.* **18**(3), 348–365 (2014)
9. Li, M., Yang, S., Liu, X.: Bi-goal evolution for many-objective optimization problems. *Artif. Intell.* **228**, 45–65 (2015)
10. Liu, Y., Gong, D., Sun, J., Jin, Y.: A many-objective evolutionary algorithm using a one-by-one selection strategy. *IEEE Trans. Cybern.* **47**(9), 2689–2702 (2017)
11. Liu, Y., Gong, D., Sun, X., Zhang, Y.: Many-objective evolutionary optimization based on reference points. *Appl. Soft Comput.* **50**(1), 344–355 (2017)
12. Tian, Y., Cheng, R., Zhang, X., Jin, Y.: PlatEMO: a MATLAB platform for evolutionary multi-objective optimization [educational forum]. *IEEE Comput. Intell. Mag.* **12**(4), 73–87 (2017)
13. Wang, Z., Zhang, Q., Li, H., Ishibuchi, H., Jiao, L.: On the use of two reference points in decomposition based multiobjective evolutionary algorithms. *Swarm Evol. Comput.* **34**, 89–102 (2017)
14. Yang, S., Li, M., Liu, X., Zheng, J.: A grid-based evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **17**(5), 721–736 (2013)
15. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
16. Zhang, X., Tian, Y., Jin, Y.: A knee point driven evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **19**(6), 761–776 (2015)
17. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength Pareto evolutionary algorithm. Technical report, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK) (2001)
18. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. Evol. Comput.* **7**(2), 117–132 (2003)



# New Initialisation Techniques for Multi-objective Local Search

## Application to the Bi-objective Permutation Flowshop

Aymeric Blot<sup>1</sup>(✉), Manuel López-Ibáñez<sup>2</sup>, Marie-Éléonore Kessaci<sup>1</sup>,  
and Laetitia Jourdan<sup>1</sup>

<sup>1</sup> Université de Lille, CNRS, UMR 9189 – CRISTAL, Lille, France  
{aymeric.blot,mkessaci,laetitia.jourdan}@univ-lille.fr

<sup>2</sup> Alliance Manchester Business School, University of Manchester, Manchester, UK  
manuel.lopez-ibanez@manchester.ac.uk

**Abstract.** Given the availability of high-performing local search (LS) for single-objective (SO) optimisation problems, a successful approach to tackle their multi-objective (MO) counterparts is scalarisation-based local search (SBLS). SBLS strategies solve multiple scalarisations, aggregations of the multiple objectives into a single scalar value, with varying weights. They have been shown to work specially well as the initialisation phase of other types of MO local search, e.g., Pareto local search (PLS). A drawback of existing SBLS strategies is that the underlying SO-LS method is unaware of the MO nature of the problem and returns only a single solution, discarding any intermediate solutions that may be of interest. We propose here two new SBLS initialisation strategies (ChangeRestart and ChangeDirection) that overcome this drawback by augmenting the underlying SO-LS method with an archive of nondominated solutions used to dynamically update the scalarisations. The new strategies produce better results on the bi-objective permutation flowshop problem than other five SBLS strategies from the literature, not only on their own but also when used as the initialisation phase of PLS.

**Keywords:** Flowshop scheduling · Local search · Heuristics  
Multi-objective optimisation · Combinatorial optimisation

## 1 Introduction

Multi-objective (MO) local search methods [5, 7, 11] are usually classified into two types. Scalarisation-based local search (SBLS) strategies aggregate the multiple objectives into a single (scalar) one by means of weights, and use single-objective (SO) local search to tackle each scalarised problem. Dominance-based local search (DBLS) strategies search the neighbourhood of candidate solutions for (Pareto) dominating or nondominated solutions. Successful algorithms for MO combinatorial optimisation problems often hybridise both strategies by generating a set of high-quality solutions by means of SBLS, and further improving this set by applying a DBLS method [3, 5, 7, 8].



Various SBLS strategies have been proposed in the literature that mainly differ in the sequence of weights explored during the search and the starting solution for solving each scalarisation. The simplest method, henceforth called **Restart** [10], uses a uniform set of weights and starts each scalarisation from a randomly (or heuristically) generated solution. More advanced strategies, such as **AdaptiveAnytime** [4], dynamically compute the next weight and choose a starting solution among the best ones found so far with the goal of closing the largest “gap” in the current Pareto front approximation.

We propose to augment the SO local search that solves the scalarisations with an archive of nondominated solutions, such that they are able to return a set of solutions that are available to the overall SBLS strategy and to the other local search runs solving other scalarisations. We also propose two new SBLS strategies able to generate high-quality solutions to initialize DBLS. With **ChangeRestart**, we subdivide the time granted to solve each scalarisation in multiple steps, and use intermediary solutions to restart each local search run when it falls behind. With **ChangeDirection**, we further improve **ChangeRestart** by changing not only the starting solution of a local search run, but also the weight that defines the scalarisation being solved. As a case study, we focus on a bi-objective variant of the permutation flowshop scheduling problem (PFSP), which has been used previously as a benchmark for MO local search [3].

This paper is organised as follows. Section 2 describes classical SBLS strategies and the bi-objective PFSP considered here. Section 3 proposes to augment SO local search used within SBLS strategies with a nondominated archive, and Sect. 4 proposes two new SBLS strategies. The experimental setup and results are discussed in Sects. 5 and 6, respectively. Section 7 summarises the conclusions.

## 2 Background

### 2.1 Scalarisation-Based Local Search (SBLS)

In MO combinatorial optimisation problems, we have a set of feasible solutions  $S$ , where each solution  $s \in S$  may be evaluated according to a vector of  $M$  objectives  $\mathbf{f}(s) = (f_1(s), \dots, f_M(s))$ . Without a priori information, candidate solutions are usually compared in terms of Pareto dominance:  $s_1$  dominates  $s_2$  iff  $\forall i = 1, \dots, M, f_i(s_1) \leq f_i(s_2)$  and  $\exists j, f_j(s_1) < f_j(s_2)$  (assuming minimisation without loss of generality). The goal becomes to find, or approximate as well as possible, the Pareto-optimal set, i.e., the set of solutions  $S^* \subset S$  that are not dominated by any other solution in  $S$ . The image of the Pareto-optimal set in the objective space is called the Pareto front.

An MO problem can be transformed into a SO one by *scalarising* it, for example, by means of weighted sum. For simplicity, we will focus on the bi-objective ( $M = 2$ ) case in the following. Given a problem with two objectives  $\mathbf{f}(s) = (f_1(s), f_2(s))$  and a normalised weight vector  $\boldsymbol{\lambda} = (\lambda, 1 - \lambda)$ , where  $\lambda \in [0, 1] \subset \mathbb{R}$ , the corresponding scalarised problem (*scalarisation*) is computed as  $f_\lambda(s) = \lambda \cdot f_1(s) + (1 - \lambda) \cdot f_2(s)$ . An optimal solution of this SO scalarisation is a Pareto-optimal solution of the MO problem, thus multiple Pareto-optimal

solutions (although maybe not all) may be obtained by solving multiple scalarisations with different weights. The main advantage of solving scalarisations instead of the original MO problem is that, very often, highly effective and efficient local search algorithms exist for the single-objective case. SBLS approaches are conceptually related to decomposition-based algorithms (e.g., MOEA/D [14]).

Classical SBLS strategies differ in how weights are generated and which solution is used as the starting point of each local search run  $LS_\lambda$  solving  $f_\lambda$ :

**Restart.** Perhaps the simplest strategy consists in generating a set of uniformly distributed weights and start each  $LS_\lambda$  run from a randomly or heuristically generated solution [10].

**TPLS.** In its simplest version [10], one high-quality solution is generated by optimising just the first objective. In a second phase, a sequence of scalarisations of the problem, with weights that increasingly favours the second objective, are tackled by running  $LS_\lambda$ , thus generating solutions *along* the Pareto frontier from the first to the second objective. Moreover, each run of  $LS_\lambda$  starts from the best solution found for the previous scalarisation. This strategy is called **1to2** or **2to1** depending on the objective optimised in the first phase, and produce better solutions towards that objective. An alternative strategy (**Double**) avoids this bias by using half of the weights for 1to2 and the other half for 2to1 [4, 10].

**AdaptiveAnytime.** Unless the problem is fairly regular in terms of difficulty and the Pareto front is roughly symmetric for all scalarising directions, the above TPLS strategies can lead to uneven exploration of the objective space and poorly distributed approximation of the Pareto front. Similar poor results are also obtained when the algorithm is terminated before finishing the predefined number of scalarisations. The **AdaptiveAnytime** strategy was proposed to address these issues [4]. Similar to **Double TPLS**, a first phase generates one high-quality solution for each individual objective and a second phase solves a sequence of scalarisations. **AdaptiveAnytime** maintains a set  $G$  of “gaps” in the current Pareto front approximation, where each gap is a pair of solutions that are neighbours in the objective space, i.e., no other solution exists within the hyper-cube defined by them, and the size of the gap is the volume of this hyper-cube. The most successful variant of **AdaptiveAnytime** solves two scalarisations at each step, by first finding the largest gap in  $G$ , e.g.,  $(s_1, s_2)$  with  $f_1(s_1) < f_1(s_2)$ , then computing:

$$\begin{cases} \lambda_1 = \lambda - \theta \cdot \lambda \\ \lambda_2 = \lambda + \theta \cdot (1 - \lambda) \end{cases} \text{ where } \lambda = \frac{f_2(s_1) - f_2(s_2)}{f_2(s_1) - f_2(s_2) + f_1(s_2) - f_1(s_1)} \quad (1)$$

and  $\theta \in [0, 1]$  is a parameter that biases  $\lambda_1$  towards the first objective and  $\lambda_2$  towards the second objective; and solving  $f_{\lambda_1}$  starting from  $s_1$  and  $f_{\lambda_2}$  starting from  $s_2$ . The solution returned by solving each scalarisation is used to update  $G$ , by removing any dominated solutions and updating the corresponding gaps. Thus, each step of the **AdaptiveAnytime** strategy tries to reduce the size of the largest gap and adapt the weights to the shape of the current front.

## 2.2 Bi-objective Permutation Flowshop Scheduling

The above SBLs strategies have been tested on various bi-objective PFSPs [4] and `AdaptiveAnytime` was later used as the initialisation phase of the state-of-the-art MO local search [3]. The PFSP is among the best-known problems in the scheduling literature, since it models several typical problems in manufacturing. Given a set of  $n$  jobs to be processed sequentially on  $m$  machines, where each job requires a different processing time on each machine, the goal is to find a permutation of the jobs that optimises particular objectives, such that all the jobs are processed in the same order on all machines, and the order of the machines is the same for all jobs. In this paper, we focus on the bi-objective variant (bPFSP) that minimises the completion time of the last job (makespan) and the sum of completion times of all jobs (total flowtime).

## 3 Archive-Aware SBLs Strategies

Classical SBLs strategies (`Restart`, `1to2`, `2to1`, `Double` and `AdaptiveAnytime`) use a SO local search to find a new solution optimised for a given scalarisation. Each local search run ( $LS_\lambda$ ) starts from a given solution and returns the single best solution found for that particular scalarisation. Any other solution found during the run is discarded, even when not dominated by the solution returned.

We propose to augment the SO local search with an *archive* that keeps track of nondominated solutions found while solving a scalarisation, in order to preserve solutions that may be optimal for the MO problem, even if they are not for the particular scalarisation. Since such intermediary solutions are fully evaluated to compute their scalarised value, keeping an archive of these solutions only adds the computational overhead of updating the archive. In practice, adding every solution evaluated to the archive would require too much time. Instead, we only update the archive when a new solution replaces the current one.

As an example of SO local search, let us consider iterated greedy (IG) [12]. At each iteration of IG, the current solution  $\pi$  is randomly destructed (by removing some jobs from it), heuristically reconstructed (by re-inserting the jobs in new positions), and the resulting solution may be further improved by another local search. An acceptance criterion replaces the current solution ( $\pi$ ) with the new one if the latter is better or some other condition is met. In any case, if the new solution improves the best-so-far one ( $\pi^*$ ), the latter is replaced. The algorithm returns  $\pi^*$  once it terminates.

Our proposed archive-aware IG adds an archive of nondominated solutions ( $A$ ) that is updated every time a better current solution is found, and returns the archive in addition to the best solution found. Any other SO local search used within SBLs strategies can be made archive-aware in a similar manner.

We propose variants of the classical SBLs strategies that make use of such archive-aware SO local search and we denote such variants with the suffix “<sub>arch</sub>”. In `Restartarch`, `1to2arch`, `2to1arch` and `Doublearch`, each local search run produces an archive instead of a single solution. The resulting  $N^{\text{scalar}}$  archives are independent of each other until merged into a final archive. Thus, the search

trajectory of these archive-aware SBLS variants is the same as their original counterparts, except for the overhead incurred by updating the archives. In the case of `AdaptiveAnytimearch`, the archive returned by each local search run is immediately merged with the overall archive so that all solutions returned by the local search are used for computing the next largest gap.

## 4 New SBLS Strategies: `ChangeRestart`, `ChangeDirection`

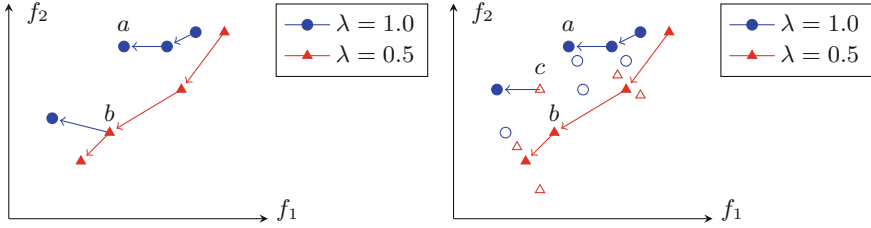
### 4.1 `ChangeRestart`

We observed that the sub-spaces searched by running the SO local search for different values of  $\lambda$  often overlap, thus the best-so-far solution found for one scalarisation may be worse than the best-so-far solution found for another when the latter solution is evaluated on the former scalarisation. Thus, the main idea behind `ChangeRestart` is to divide each local search run ( $LS_\lambda$ ) into smaller steps and, at each step, decide to either continue the run until the next step or restart it from a new solution. In particular, the time limit assigned to each  $LS_\lambda$  run is divided by  $N^{\text{steps}}$  (when  $N^{\text{steps}} = 1$ , `ChangeRestart` is identical to `Restart`). When interrupted,  $LS_\lambda$  returns its best-so-far solution ( $\pi_\lambda^*$ ). Then, for all weights  $\lambda$ , we calculate the scalarised value  $f_\lambda$  of all solutions in the current nondominated archive  $A$ , and we can limit the computational overhead of this recalculation by reducing the number of steps ( $N^{\text{steps}}$ ). After interrupting  $LS_{\lambda'}$  and  $LS_{\lambda''}$ , if  $f_{\lambda'}(\pi_{\lambda''}^*) < f_{\lambda'}(\pi_{\lambda'}^*)$ , then  $LS_{\lambda'}$  restarts its search from  $\pi_{\lambda''}^*$ . In the archive-aware variant `ChangeRestartarch`, each run of  $LS_\lambda$  returns a nondominated archive that is merged with the overall archive  $A$ .

Figure 1 shows possible executions of `ChangeRestart` and `ChangeRestartarch` for two scalarisations and three steps ( $N^{\text{steps}} = 3$ ). Blue points ( $\bullet$ ) and red triangles ( $\blacktriangle$ ) show the initial solutions and the best solutions found after each step. These solutions are connected with arrows to show the trajectory followed by each run of  $LS_\lambda$ . Unfilled points ( $\circ$ ) and triangles ( $\triangle$ ) show intermediary solutions in the archive after each step. For `ChangeRestart` (left), after the second step, the solution ( $a$ ) found for  $\lambda = 1$  has a worse value in the first objective than the solution ( $b$ ) found for  $\lambda = 0.5$ . Thus, the local search for  $\lambda = 1$  re-starts from solution  $b$  instead of  $a$ . For `ChangeRestartarch` (right), the local search re-starts instead from solution ( $c$ ), as it has an even better value regarding objective  $f_1$ .

### 4.2 `ChangeDirection`

While `ChangeRestart` is an extension of `Restart`, the second SBLS strategy proposed here, `ChangeDirection`, is inspired by the more advanced `AdaptiveAnytime`, which dynamically adapts scalarisation weights according to the gaps in the current overall archive ( $A$ ), in order to focus the search in the direction that will most improve the current approximation to the Pareto front. In `ChangeDirection`, as in `ChangeRestart`, the runs of  $LS_\lambda$  are also divided in a number of steps and, after each step, solutions from different scalarisations are merged into  $A$ . However, instead of only updating the starting solution of each  $LS_\lambda$  run, the weight



**Fig. 1.** Example runs of `ChangeRestart` (left) and `ChangeRestartarch` (right) ( $N^{\text{steps}} = 3$ ). (Color figure online)

$\lambda$  is also updated. That is, in addition to speeding up an  $LS_\lambda$  run by re-starting from a better initial solution, the scalarisation direction of  $LS_\lambda$  may be changed to focus on the largest gap in the current approximation front. In particular, a weight  $\lambda$  is replaced by another weight whenever the best-so-far solution of  $LS_\lambda$  is worse, according to  $f_\lambda$ , than a solution returned by another local search run. In that case, the computational resources allocated to searching in the direction given by  $\lambda$  could be better used in searching on a different direction.

`ChangeDirection` only differs from `ChangeRestart` in the deletion and replacement of scalarisation directions. Thus, we will only explain those novel parts. First, we delete those scalarisation weights for which the best solution found in the last run of  $LS_\lambda$  is worse, according to the same scalarisation  $f_\lambda$ , than a solution in  $A$ . Then, following the strategy of `AdaptiveAnytime` explained earlier, the gaps in the current approximation front are computed and new weights are generated from the largest gap to replace the deleted ones. In particular, two weights are generated from each gap (Eq. 1) until all deleted weights are replaced. When only one additional weight is needed, it is chosen randomly between the two weights produced by the gap. The new scalarisations then start from the solutions constituting the sides of the gap. If all gaps are used and additional weights are needed, they are drawn uniformly at random within  $[0, 1]$  and initial solutions are taken uniformly at random from  $A$ . Finally, as in `ChangeRestart`, each  $LS_\lambda$  either re-starts from a new initial solution if its scalarisation was introduced in this step, or continues from its current solution, otherwise. As previously, in the archive-aware variant `ChangeDirectionarch`, each run of  $LS_\lambda$  returns a nondominated archive that is merged with the overall archive  $A$ .

## 5 Experimental Setup

We wish to investigate not only whether the new proposed SBLS strategies work well on their own, but also if they provide a good initial set for a dominance-based local search (DBLS) algorithm. Thus, we use the various SBLS strategies as the initialisation phase of a hybrid of SBLS + DBLS algorithm, where a SBLS strategy generates an initial approximation front that is further improved by a DBLS strategy, in our case, an iterated variant of Pareto local search (IPLS).

We use IG as the single-objective local search ( $LS_\lambda$ ) and the algorithms are evaluated on the bi-objective PFSP (bPFSP). In this section, we explain the details of the our experimental setup.

**bPFSP Instances.** As a benchmark, we consider the well-known Tailard instances [13], in particular, 80 instances divided into 8 classes with  $\{20, 50, 100, 200\}$  jobs and  $\{10, 20\}$  machines, i.e., 10 instances for each combination jobs  $\times$  machines.

**Iterated Greedy (IG).** The single-objective local search used by the SBLS strategies is Iterative greedy (IG) [12], which is a state-of-the-art algorithm for the single-objective PFSP. The particular IG variant and parameter settings are directly taken from the bPFSP literature [3]. For the archive-aware SBLS strategies, we augment this IG variant with an archive as explained in Sect. 3.

**Iterated Pareto Local Search (IPLS).** As the DBLS component of our hybrid SBLS + DBLS algorithm, we consider an iterated variant of Pareto local search (PLS) [9], as it was shown that even simple perturbations could benefit PLS algorithms [2]. Our iterated PLS (IPLS) extends the PLS used in [3] by perturbing the archive when the latter converges to a Pareto local optimal set, using the generalised framework of [1]. The perturbation used creates a new archive by taking every current solution and replacing it with one of its neighbours, taken uniformly at random, three times in a row; dominated solutions from this new set are then filtered. As the neighbourhood of PLS, we use the union of the exchange and insertion neighbourhoods [6], in which two positions of two jobs are swapped and one job is reinserted at another position, respectively.

**Termination Criteria.** The termination criterion of algorithms applied to the bPFSP is usually set as maximum running time linearly proportional to both the number of jobs  $n$  and machines  $m$  (e.g.,  $0.1 \cdot n \cdot m$  CPU seconds [3]). Instead, we use a maximum running time for the hybrid SBLS + IPLS of  $0.002 \cdot n^2 \cdot m$  CPU seconds. Indeed, the total number of solutions grows exponentially and the typical size of permutation neighbourhoods grows quadratically, making a linear running time less relevant. The coefficient 0.002 was chosen to match the linear time for  $n = 50$  and  $m = 20$ . The SBLS strategies are limited to 25% of this maximum running time, and the remaining 75% is allocated to IPLS. In IPLS, a perturbation occurs after  $n$  successive iterations without improvement.

The main parameter of the SBLS strategies is the number of scalarisations ( $N^{\text{scalar}}$ ), that is, the number of runs of IG executed in addition to two individual runs for each of the two single objectives. Following [3], we perform longer runs of IG for the two single objectives ( $IG_{\{1,2\}}$ ) than for the other scalarisations ( $IG_\lambda$ ), with the time assigned to  $IG_{\{1,2\}}$  being 1.5 times the time assigned to  $IG_\lambda$ . As more time is allocated to  $IG_{\{1,2\}}$  than to  $IG_A$ , their respective running time budgets are  $1.5/(N^{\text{scalar}} + 3)$  and  $1/(N^{\text{scalar}} + 3)$  of the total time assigned to the SBLS strategy. In the case of *ChangeRestart* and *ChangeDirection*, the maximum runtime of each IG is further divided by  $N^{\text{steps}}$ .

The following experiments are separated in three successive phases. First, we analyse the effect of using an archive-aware IG on the five SBLS strategies

from the literature (Restart, 1to2, 2to1, Double, and AdaptiveAnytime). Second, we compare all these SBLs variants with the new SBLs strategies proposed here (ChangeRestart and ChangeDirection), including their archive-aware counterparts. Finally, we analyse other possible setting for the parameters  $N^{\text{scalar}}$  and  $N^{\text{steps}}$ . Unless stated otherwise, ChangeRestart and ChangeDirection use  $N^{\text{steps}} = 20$ ; all SBLs strategies use a fixed value of  $N^{\text{scalar}} = 12$ ; and both AdaptiveAnytime and ChangeDirection use  $\theta = 0.25$  for Eq. 1 [3].

In all cases, we run the hybrid SBLs+IPLS and we save the archive returned by the SBLs strategies (before IPLS) and the final archive (after IPLS). Each experiment is repeated 5 times, using independent random seeds, on each of the 80 Taillard instances, that is, we perform for each strategy 50 runs per instance class and 400 runs in total. All replications use the same seeds on the same instances. All the experiments have been conducted on Intel Xeon E5-2687W V4 CPUs (3.0 GHz, 30 MB cache, 64 GB RAM).

Results are evaluated according to both the hypervolume and the additive- $\varepsilon$  indicators [15]. Indicator values have been computed independently on every run by aggregating all results generated for an instance and scaling both objectives to a 0–1 scale in which 0 (1) corresponds to the minimum (maximum) objective value reached by any solution. The hypervolume variant  $1 - HV$  is used, with 0 corresponding to the maximum hypervolume, so that both indicators are to be minimised. The reference point used for computing the hypervolume indicator is (1.0001, 1.0001). The reference set for computing the additive- $\varepsilon$  indicator is the set of nondominated solutions from all aggregated results for each instance.

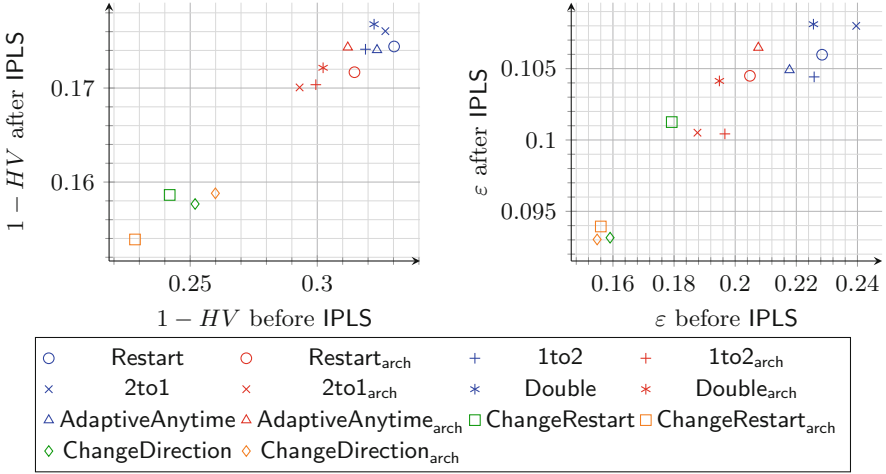
## 6 Experimental Results

### 6.1 Known SBLs Strategies vs. Their Archive-Aware Variants

First, we compare the five classical SBLs strategies with their archive-aware variants. Figure 2 shows the mean hypervolume and additive- $\varepsilon$  values, over all 80 bPFSP instances, obtained by each strategy before and after running IPLS. For both indicators, all the archive-aware variants (in red) lead to improved quality before IPLS. After results are improved by IPLS, all archive-aware variants produce again better results than their original counter-parts, with the exception of AdaptiveAnytime. This is somewhat surprising and further analysis is needed to understand this behaviour. Interestingly, some of the archive-aware variants are able to outperform AdaptiveAnytime when their original variants are not.

### 6.2 Performance of the Two New SBLs Strategies

We now compare the newly proposed SBLs strategies (ChangeRestart and ChangeDirection) to the ones from the literature as well as their archive-aware variants. In terms of hypervolume (Fig. 2, left), all four new strategies achieve in average much better results than the strategies from the literature, with ChangeRestart<sub>arch</sub> achieving the best results both on its own and when further



**Fig. 2.** Comparison of all SBLS strategies according to (left) mean hypervolume and (right) mean additive- $\epsilon$ . (Color figure online)

improved by IPLS. However, in terms of additive- $\epsilon$  (Fig. 2, right), the non-archive-aware ChangeRestart strategy performs much worse than the three other new strategies, but still better than most strategies from the literature. Overall, the best strategy according to both indicators appears to be the archive-aware ChangeRestart strategy.

To validate these observations, Table 1 shows the results of a statistical analysis comparing all approaches, without averaging over instance classes, for both hypervolume (top) and  $\epsilon$  (bottom). For each instance class, and for all possible pairs of strategies, we conducted a statistical Wilcoxon test comparing their final quality (after IPLS) paired on the 50 values per class. The symbol “✓” in the table indicates strategies for which there was no other strategy performing statistically better (95% confidence). In other words, within each row, all strategies with “✓” are not statistically better to each other, while for those strategies without “✓”, there was at least one other strategy statistically better. As shown in Table 1, the SBLS strategies from the literature are often outperformed by some other strategy, whereas their archive-aware variants are less often so, in particular on the smaller instances with 20 and 50 jobs. Finally, ChangeRestart<sub>arch</sub> and both variants of ChangeDirection are almost never outperformed, even on the largest instances, validating our previous observations.

### 6.3 Analysis of Parameters $N^{\text{scalar}}$ and $N^{\text{steps}}$

SBLS strategies strongly depend on the number of scalarisations. Our choice of  $N^{\text{scalar}} = 12$  was motivated by previous studies claiming that few scalarisations should be preferred [3]. Figure 3 (left) shows for the 14 previous strategies the final performance regarding both hypervolume and additive  $\epsilon$  indicators, for both



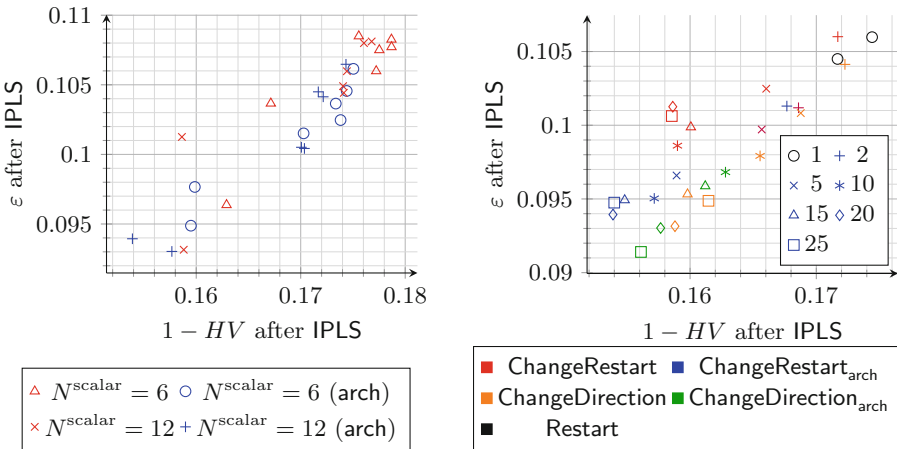
**Table 1.** SBLs strategies not statistically outperformed by another strategy after IPLS step, using paired Wilcoxon tests (left: hypervolume; right: additive- $\epsilon$ )

	Restart	1to2	2to1	Double	AdaptiveAnytime	Restart <sub>arch</sub>	1to2 <sub>arch</sub>	2to1 <sub>arch</sub>	Double <sub>arch</sub>	AdaptiveAnytime <sub>arch</sub>	ChangeRestart	ChangeRestart <sub>arch</sub>	ChangeDirection	ChangeDirection <sub>arch</sub>	Restart	1to2	2to1	Double	AdaptiveAnytime	Restart <sub>arch</sub>	1to2 <sub>arch</sub>	2to1 <sub>arch</sub>	Double <sub>arch</sub>	AdaptiveAnytime <sub>arch</sub>	ChangeRestart	ChangeRestart <sub>arch</sub>	ChangeDirection	ChangeDirection <sub>arch</sub>
20 × 10						✓	✓			✓	✓	✓	✓							✓	✓	✓	✓	✓	✓	✓	✓	
20 × 20						✓	✓			✓	✓	✓	✓							✓	✓	✓	✓	✓	✓	✓	✓	
50 × 10		✓				✓	✓			✓	✓	✓	✓					✓		✓	✓	✓	✓	✓	✓	✓	✓	
50 × 20	✓	✓	✓			✓	✓			✓	✓	✓	✓		✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	
100 × 10											✓	✓	✓								✓	✓	✓	✓	✓	✓	✓	
100 × 20		✓				✓					✓	✓	✓		✓			✓		✓	✓	✓	✓	✓	✓	✓	✓	
200 × 10											✓	✓	✓											✓	✓	✓	✓	
200 × 20											✓	✓	✓											✓	✓	✓	✓	

parameter values of  $N^{\text{scalar}} \in \{6, 12\}$  scalarisations, in order to see the impact of archives-aware mechanisms when using very few scalarisations.

We can see that for all strategies, both with and without archiving, using 12 scalarisations improves significantly the mean performance regarding hypervolume and slightly the one regarding the  $\epsilon$  indicator, hinting that even with archiving a sufficient number of scalarisations are still required.

The number of steps, i.e., how many times we can restart the scalarisations, is at the core of the two new SBLs strategies we propose. Figure 3 (right) shows



**Fig. 3.** Impact of the number of scalarisations (left) and the number of steps (right) (Color figure online)

for all variants of the **ChangeRestart** and **ChangeDirection** strategies the impact of the parameter  $N^{\text{steps}}$ , for values of  $N^{\text{steps}} = 1$  (equivalent to **Restart**) and  $N^{\text{steps}} \in \{2, 5, 10, 15, 20, 25\}$ , using the final performance regarding both the hypervolume and  $\varepsilon$  indicators. Marks indicate the value of  $N^{\text{steps}}$ , while colours indicate the strategy.

As the figure shows, at first increasing the number of steps from one largely improves the quality of the results. Increasing the number of steps further especially benefits the archive-aware variants and in particular, **ChangeDirection<sub>arch</sub>**. However, for large values of  $N^{\text{steps}}$ , the quality improvements stop or, in several cases, worsen. Thus, it appears that even larger values would not improve the results reported here.

## 7 Conclusion

This paper proposes and evaluates two complementary ways of augmenting scalarisation-based local search (SBLS) strategies by making the underlying single-objective local search aware of the multi-objective nature of the problem. Our first proposal adds an archive of nondominated solutions to the single-objective local search. Our results showed that these archive-aware SBLS variants always improve over their original counterparts when ran on their own. Moreover, this improvement also shows for nearly all SBLS strategies when acting as the initialisation phase of an iterated Pareto local search (IPLS).

Our second proposal was to divide each run of the single-objective local search into a number of smaller steps and, at each step, restart scalarisations that produce poor results. We proposed two SBLS strategies that differ on what is changed by the restart. In **ChangeRestart**, the local search for solving a scalarisation is restarted from the best-known solution for that scalarisation problem. This solution was possibly generated when solving a different scalarisation. In **ChangeDirection**, not only the starting solution, but also the weight that defines the scalarisation problem itself being solved are both updated in order to re-focus this particular run on the largest gap of the current approximation front.

Our experimental results show that these two new SBLS strategies outperform five classical SBLS strategies from the literature, even when the latter are using an archive-aware local search. In particular, **ChangeDirection** produces consistently the best results, either on its own or when used as the initialisation phase of a hybrid SBLS + IPLS algorithm, which suggests that the new strategies may lead to new state-of-the-art results for the bi-objective permutation flowshop [3], and other problems. An additional benefit of **ChangeDirection** is that it maintains the adaptive behaviour of **AdaptiveAnytime**, while it also may perform  $N^{\text{scalar}}$  local search runs in parallel between steps.

Future work will analyse in more detail the interaction between the new SBLS strategies and the archive-aware SO local search. A more comprehensive analysis of the effect of the  $N^{\text{scalar}}$  and  $N^{\text{steps}}$  parameters would be needed to understand their interactions with problem features. We would also hope to evaluate the new proposals in terms of their anytime behaviour [4]. Finally,

we focused here on archive-aware mechanisms and we did not consider various common speedups that would be required for a fair comparison with other state-of-the-art algorithms.

## References

1. Blot, A., Jourdan, L., Kessaci-Marmion, M.E.: Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem. In: GECCO 2017, pp. 227–234. ACM Press (2017)
2. 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 2010. LNCS, vol. 6238, pp. 485–495. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_49](https://doi.org/10.1007/978-3-642-15844-5_49)
3. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *COR* **38**(8), 1219–1236 (2011)
4. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Improving the anytime behavior of two-phase local search. *AMAI* **61**(2), 125–154 (2011)
5. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Combining two search paradigms for multi-objective optimization: two-phase and Pareto local search. In: Talbi, E.G. (ed.) *Hybrid Metaheuristics*, vol. 434, pp. 97–117. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-30671-6\\_3](https://doi.org/10.1007/978-3-642-30671-6_3)
6. Dubois-Lacoste, J., López-Ibáñez, M., Stützle, T.: Anytime Pareto local search. *EJOR* **243**(2), 369–385 (2015)
7. Liefoghe, 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. *JOH* **18**(2), 317–352 (2011)
8. Lust, T., Teghem, J.: The multiobjective multidimensional knapsack problem: a survey and a new approach. *ITOR* **19**(4), 495–520 (2012)
9. 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*. LNMES, vol. 535, pp. 177–200. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-642-17144-4\\_7](https://doi.org/10.1007/978-3-642-17144-4_7)
10. Paquete, L., Stützle, T.: A two-phase local search for the biobjective traveling salesman problem. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L., Deb, K. (eds.) *EMO 2003*. LNCS, vol. 2632, pp. 479–493. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36970-8\\_34](https://doi.org/10.1007/3-540-36970-8_34)
11. Paquete, L., Stützle, T.: Stochastic local search algorithms for multiobjective combinatorial optimization: a review. In: *Handbook of Approximation Algorithms and Metaheuristics*, pp. 29–1–29–15. Chapman & Hall/CRC (2007)
12. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *EJOR* **177**(3), 2033–2049 (2007)
13. Taillard, É.D.: Benchmarks for basic scheduling problems. *EJOR* **64**(2), 278–285 (1993)
14. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE TEC* **11**(6), 712–731 (2007)
15. 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)



# Towards a More General Many-objective Evolutionary Optimizer

Jesús Guillermo Falcón-Cardona<sup>(✉)</sup> and Carlos A. Coello Coello

Computer Science Department, CINVESTAV-IPN,  
Av. IPN No. 2508, Col. San Pedro Zacatenco, 07300 México D.F., Mexico  
jfalcon@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

**Abstract.** Recently, it has been shown that the current Many-Objective Evolutionary Algorithms (MaOEAs) are overspecialized in solving certain benchmark problems. This overspecialization is due to a high correlation between the Pareto fronts of the test problems with the convex weight vectors commonly used by MaOEAs. The main consequence of such overspecialization is the inability of these MaOEAs to solve the minus versions of well-known benchmarks (e.g., the DTLZ<sup>-1</sup> test suite). In furtherance of avoiding this issue, we propose a novel steady-state MaOEA that does not require weight vectors and uses a density estimator based on the IGD<sup>+</sup> indicator. Moreover, a fast method to calculate the IGD<sup>+</sup> contributions is integrated in order to reduce the computational cost of the proposed approach, which is called IGD<sup>+</sup>-MaOEA. Our proposed approach is compared with NSGA-III, MOEA/D, IGD<sup>+</sup>-EMOA (the previous ones employ convex weight vectors) and SMS-EMOA on the test suites DTLZ and DTLZ<sup>-1</sup>, using the hypervolume indicator. Our experimental results show that IGD<sup>+</sup>-MaOEA is a more general optimizer than MaOEAs that need a set of convex weight vectors and it is competitive and less computational expensive than SMS-EMOA.

**Keywords:** Multi-objective optimization · Quality indicators  
Density estimation

## 1 Introduction

In the scientific and industrial fields, there is a wide variety of problems that involve the simultaneous optimization of several, often conflicting, objective functions. These are the so-called multi-objective optimization problems (MOPs) which are mathematically defined as follows:

$$\min_{\mathbf{x} \in \Omega} \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}))^T \quad (1)$$

---

The first author acknowledges support from CONACyT and CINVESTAV-IPN to pursue graduate studies in Computer Science. The second author gratefully acknowledges support from CONACyT project no. 221551.

where  $\mathbf{x}$  is the vector of decision variables,  $\Omega \subseteq \mathbb{R}^n$  is the decision variable space and  $\mathbf{F}(\mathbf{x})$  is the vector of  $m (\geq 2)$  objective functions. The solution of an MOP is a set of solutions that represent the best possible trade-offs among the objectives, i.e., finding solutions in which an objective function cannot be improved without worsening another. The particular set that yields the optimal values is known as the Pareto Optimal Set ( $\mathcal{P}^*$ ) and its image in the objective space is known as the Pareto Optimal Front ( $\mathcal{PF}^*$ ).

Multi-Objective Evolutionary Algorithms (MOEAs) are population-based and gradient-free methods that have been successfully applied to solve MOPs [1]. For several years, MOEAs have adopted the Pareto dominance relation.<sup>1</sup> However, Pareto-based MOEAs does not perform properly when tackling MOPs having four or more objective functions, i.e., the so-called many-objective optimization problems (MaOPs) [2]. This behavior is due to the rapid increase of solutions preferred by the use of Pareto dominance which directly produces a dilution of the selection pressure. With the aim of properly regulating the selection pressure of a MOEA three main approaches have been considered for MaOPs: (1) to define new dominance relations (mainly based on relaxed forms of Pareto dominance), (2) decomposition of the MOP, and (3) indicator-based selection.

Many-Objective Evolutionary Algorithms (MaOEAs) based on decomposition and performance indicators<sup>2</sup> are the most popular alternatives in the current literature [2]. Most of the state-of-the-art MaOEAs employ a set of convex weight vectors. A vector  $\mathbf{w} \in \mathbb{R}^m$  is a convex weight vector if  $\sum_{i=1}^m w_i = 1$  and  $w_i \geq 0$  for all  $i = 1, \dots, m$ . These weight vectors lie on an  $(m - 1)$ -simplex and are used by MaOEAs as search directions [3], reference points [4, 5] or as part of an indicator's definition [6]. However, in 2017, Ishibuchi *et al.* [7] empirically showed that the use of convex weight vectors overspecializes MaOEAs on MOPs whose Pareto fronts are strongly correlated to the simplex formed by the weight vectors.

In this paper, we propose a steady-state MaOEA that uses Pareto dominance as its main selection criterion and a density estimator based on the Inverted Generational Distance plus (IGD<sup>+</sup>) indicator. The proposed approach, called IGD<sup>+</sup>-MaOEA, does not require a set of convex weight vectors in any of its mechanisms in furtherance of avoiding the previously indicated overspecialization. Furthermore, a fast IGD<sup>+</sup> contribution computation method is integrated into the proposed approach to reduce its computational cost.

The remainder of this paper is organized as follows. Section 2 presents an overview of some state-of-the-art MaOEAs. The detailed description of our proposal is outlined in Sect. 3. Our experimental results are provided in Sect. 4. Finally, Sect. 5 presents our conclusions and some possible paths for future work.

<sup>1</sup> Given two solutions  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ ,  $\mathbf{u}$  dominates  $\mathbf{v}$  (denoted as  $\mathbf{u} \prec \mathbf{v}$ ), if and only if  $u_i \leq v_i$  for all  $i = 1, \dots, m$  and there exists at least an index  $j \in \{1, \dots, m\}$  such that  $u_j < v_j$ . In case  $u_i \leq v_i$  for all  $i = 1, \dots, m$ ,  $\mathbf{u}$  is said to weakly dominate  $\mathbf{v}$  (denoted as  $\mathbf{u} \preceq \mathbf{v}$ ).

<sup>2</sup> A unary performance indicator  $I$  is a function that assigns a real value to a set of  $m$ -dimensional vectors.

## 2 Previous Related Work

The MOEA based on Decomposition (MOEA/D) [3] transforms an MOP into as many single-objective optimization problems as weight vectors there are, through a scalarizing function. For each weight vector  $w^i$ , MOEA/D defines a neighborhood of size  $T$ , i.e., it finds the  $T$  nearest solutions to  $w^i$ , using Euclidean distances. Using this neighborhood structure, MOEA/D tries to optimize the scalarizing functions at each generation simultaneously. Hence, the aim is to find the intersections between the Pareto front and the weight vectors according to the value of the scalarizing function.

Deb *et al.* [4] proposed the Nondominated Sorting Genetic Algorithm III (NSGA-III). NSGA-III uses a  $(\mu + \lambda)$  selection scheme, i.e., using a population of  $\mu$  potential parents produces, at each generation,  $\lambda$  offspring. Then, the union set of parents and offspring is classified using the nondominated sorting method [8] that creates a set of disjoint ranks  $R_1, R_2, \dots, R_k$ , using Pareto dominance. Ranks are added into the next population until one of them (e.g.,  $R_j$ ) makes the population size to be larger than  $\mu$ . Hence, some solutions have to be deleted from  $R_j$  using a density estimator that employs a set of convex weight vectors to define a niche count per weight vector. Solutions from the most crowded regions are deleted until the desired population size is achieved.

In 2016, Manoatl and Coello [5] introduced the IGD<sup>+</sup>-Evolutionary Multi-Objective Algorithm (IGD<sup>+</sup>-EMOA) that is an indicator-based MaOEA. They defined an environmental selection mechanism on the transformation of an MOP into a Linear Assignment Problem, using the IGD<sup>+</sup> indicator. As IGD<sup>+</sup> needs a reference set, the authors proposed to use a set of weight vectors that try to approximate the Pareto front geometry employing Lamé Superspheres. However, by doing this, only smooth concave or convex geometries can be appropriately approximated. Consequently, IGD<sup>+</sup>-EMOA has difficulties to solve MOPs having highly irregular Pareto fronts, namely disconnected and degenerated.

The  $\mathcal{S}$ -Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) [9] is a steady-state version of the NSGA-II [8] but it implements a density estimator based on the hypervolume (HV) indicator. Due to this HV-based density estimator, SMS-EMOA increases selection pressure and drives the population to the maximization of the HV, which is directly related to finding Pareto optimal solutions [10]. Moreover, SMS-EMOA does not rely on convex weight vectors. However, its main drawback is the high computational cost associated to the computation of the individual HV contributions when the number of objective functions is greater than three, which makes its use prohibitive in MaOPs.

## 3 Our Proposed Approach

Ishibuchi *et al.* [11] proposed the IGD<sup>+</sup> indicator as an improved version of the Inverted Generational Distance (IGD) indicator [1]. The main difference between

IGD<sup>+</sup> and IGD is that the former is weakly Pareto-compliant<sup>3</sup> while the latter is Pareto non-compliant. Mathematically, given an approximation to the Pareto front  $\mathcal{A}$  and a reference set denoted as  $\mathcal{Z}$ , IGD<sup>+</sup> is defined as follows (we assume minimization):

$$\text{IGD}^+(\mathcal{A}, \mathcal{Z}) = \frac{1}{|\mathcal{Z}|} \sum_{\mathbf{z} \in \mathcal{Z}} \min_{\mathbf{a} \in \mathcal{A}} d^+(\mathbf{a}, \mathbf{z}), \quad (2)$$

where  $d^+(\mathbf{a}, \mathbf{z}) = \sqrt{\sum_{k=1}^m [\max(a_k - z_k, 0)]^2}$ . IGD<sup>+</sup> measures the average distance from each reference vector to the nearest dominated region related to an element in  $\mathcal{A}$ . The aim is to minimize the value of IGD<sup>+</sup>. If  $\text{IGD}^+(\mathcal{A}, \mathcal{Z}) = 0$ , it implies that  $\mathcal{A} = \mathcal{Z}$ ; else if the value is greater than zero, IGD<sup>+</sup> intends to determine how different are both sets.

The contribution  $C$  of a solution  $\mathbf{a} \in \mathcal{A}$  to IGD<sup>+</sup>, is defined as follows:

$$C(\mathbf{a}, \mathcal{A}, \mathcal{Z}) = |\text{IGD}^+(\mathcal{A}, \mathcal{Z}) - \text{IGD}^+(\mathcal{A} \setminus \{\mathbf{a}\}, \mathcal{Z})|. \quad (3)$$

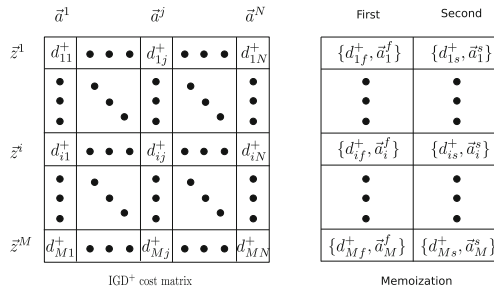
Clearly, the computational cost of calculating the contribution of a single solution is  $\Theta(mNM)$ , where  $|\mathcal{A}| = N$  and  $|\mathcal{Z}| = M$ . Based on Eq. (3), our proposed IGD<sup>+</sup>-based density estimator (IGD<sup>+</sup>-DE) aims to delete from  $\mathcal{A}$  the solution having the minimum contribution. The total runtime of IGD<sup>+</sup>-DE is  $\Theta(mN^2M)$  which is too expensive. In furtherance of reducing this computational cost, in the next section we propose a method based on memoization to achieve  $\Theta(mNM)$  time for the full IGD<sup>+</sup>-DE procedure.

### 3.1 Fast IGD<sup>+</sup> Contribution

IGD<sup>+</sup> in Eq. (2) is basically composed by  $|\mathcal{Z}|$  minimum  $d^+$  values, where each one is related to a solution, not necessarily different, in  $\mathcal{A}$ . If  $\mathbf{a} \in \mathcal{A}$  is related to one or more elements in  $\mathcal{Z}$ , it is called *contributing solution*; otherwise, it is called *noncontributing solution*. It is worth noting that the IGD<sup>+</sup> contribution of the latter is zero, and, thus, IGD<sup>+</sup>-DE deletes it first. Algorithm 1, proposed by Falcón-Cardona and Coello [12], stores in a memoization structure, for each  $\mathbf{z} \in \mathcal{Z}$ , the two smallest  $d^+$  values and the corresponding pointers to the solutions in  $\mathcal{A}$  (see Fig. 1) when  $\text{IGD}^+(\mathcal{A}, \mathcal{Z})$  is computed in line 2. For each  $\mathbf{a} \in \mathcal{A}$ , the nested for-loops of lines 4–15 compute  $\psi = \text{IGD}^+(\mathcal{A} \setminus \{\mathbf{a}\}, \mathcal{Z})$ . For this purpose, the algorithm takes advantage of the memoization structure. If  $\mathbf{a}$  is related to one or more minimum  $d^+$  values, then the second best value is added to  $\psi$ ; otherwise, the minimum  $d^+$  is added. At the end,  $C(\mathbf{a}, \mathcal{A}, \mathcal{Z}) = |\text{IGD}^+(\mathcal{A}, \mathcal{Z}) - \psi|$  is assigned to  $C_i$ . Consequently, the total runtime of Algorithm 1 is  $\Theta(mNM) + \Theta(mNM) = \Theta(mNM)$ .

When this method is integrated into IGD<sup>+</sup>-DE, its overall cost goes from  $\Theta(mN^2M)$  to  $\Theta(mNM)$ . The cost of calculating all the IGD<sup>+</sup> contributions is  $\Theta(mNM)$  and it takes  $\Theta(M)$  finding the minimum contribution, thus,  $\Theta(mNM) + \Theta(M) = \Theta(mNM)$  is the runtime of IGD<sup>+</sup>-DE.

<sup>3</sup> Let  $A$  and  $B$  be two non-empty sets of  $m$ -dimensional vectors and let  $I$  be a unary indicator.  $I$  is weakly Pareto-compliant if and only if  $A$  weakly dominates  $B$  implies  $I(A) \leq I(B)$  (assuming minimization of  $I$ ).



**Fig. 1.** IGD<sup>+</sup> cost matrix and the memoization structure. Each row of the memoization structure stores the two smallest  $d^+$  values and the corresponding pointers to the related solutions.

---

### Algorithm 1. Fast IGD<sup>+</sup> Contribution

---

**Require:** Approximation set  $\mathcal{A}$  of size  $N$ ; Reference set  $\mathcal{Z}$  of size  $M$

**Ensure:** Vector  $C = (C_i)_{i=1, \dots, N}$  of IGD<sup>+</sup> contributions

```

1: Memoization  $\leftarrow \emptyset$ 
2: total  $\leftarrow \text{IGD}^+(\mathcal{A}, \mathcal{Z}, \text{Memoization})$ 
3:  $\forall i \in \{1, \dots, |\mathcal{A}|\}, C_i \leftarrow 0$ 
4: for  $i = 1$  to  $N$  do
5:    $\psi \leftarrow 0$ 
6:   for  $j = 1$  to  $M$  do
7:     if Memoization[ $j$ ]. $a_j^f = a^i$  then
8:        $\psi \leftarrow \psi + \text{Memoization}[j].d_{j_s}^+$ 
9:     else
10:       $\psi \leftarrow \psi + \text{Memoization}[j].d_{j_f}^+$ 
11:    end if
12:  end for
13:   $\psi \leftarrow \psi/N$ 
14:   $C_i \leftarrow |\text{total} - \psi|$ 
15: end for
16: return  $C$ 

```

---

## 3.2 IGD<sup>+</sup>-MaOEA

IGD<sup>+</sup>-MaOEA is a steady-state MOEA similar to SMS-EMOA [9]. However, instead of using HV contributions, this approach uses IGD<sup>+</sup>-DE. Algorithm 2 describes the general framework of IGD<sup>+</sup>-MaOEA, where the main loop is presented in lines 2 to 13. First, a new solution  $q$  is generated by variation operators.<sup>4</sup>  $q$  is added to  $P$  to create the temporary population  $Q$  which is ranked by the nondominated sorting method in line 5. If the layer  $R_k$  has more than one solution, then IGD<sup>+</sup>-DE is executed in line 7, using Algorithm 1 where the set of nondominated solutions  $R_1$  performs as the reference set  $\mathcal{Z}$ . In case  $|R_k| = 1$ , the sole solution of  $R_k$  is deleted. For both cases,  $\mathbf{u}_{\text{worst}}$  denotes the solution to be deleted. In line 12, the population for the next generation is set. At the end of the evolutionary process, the current population  $P$  is returned.

<sup>4</sup> Simulated binary crossover (SBX) and polynomial-based mutation operators are employed [8].



**Algorithm 2.** IGD<sup>+</sup>-MaOEA general framework

---

**Require:** No special parameters needed  
**Ensure:** Approximation to the Pareto front

- 1: Randomly initialize population  $P$
- 2: **while** stopping criterion is not fulfilled **do**
- 3:    $q \leftarrow \text{Variation}(P)$
- 4:    $Q \leftarrow P \cup \{q\}$
- 5:    $\{R_1, \dots, R_k\} \leftarrow \text{NondominatedSorting}(Q)$
- 6:   **if**  $|R_k| > 1$  **then**
- 7:      $C \leftarrow \text{IGD}^+\text{DE}(\mathcal{A} = R_k, \mathcal{Z} = R_1)$
- 8:     Let  $u_{\text{worst}}$  be the solution with the minimum IGD<sup>+</sup> contribution in  $C$
- 9:   **else**
- 10:     Let  $u_{\text{worst}}$  be the sole solution in  $R_k$
- 11:   **end if**
- 12:    $P \leftarrow Q \setminus \{u_{\text{worst}}\}$
- 13: **end while**
- 14: **return**  $P$

---

## 4 Experimental Results

In order to assess the performance of IGD<sup>+</sup>-MaOEA<sup>5</sup>, we used the Deb-Thiele-Laumanns-Zitzler (DTLZ) test suite and its minus version, DTLZ<sup>-1</sup> proposed by Ishibuchi *et al.* [7] adopting  $m = 3, 4, 5, 6, 7$  objective functions. For all DTLZ and DTLZ<sup>-1</sup> instances,  $n = m + K - 1$ , where  $K$  is set to 5 for DTLZ1, 10 for DTLZ2-6 and 20 for DTLZ7 [1]. The values of  $K$  apply to the corresponding minus problems. The purpose of using DTLZ<sup>-1</sup> is to show that IGD<sup>+</sup>-MaOEA is more general than traditional MaOEAs based on the use of convex weight vectors. We compared IGD<sup>+</sup>-MaOEA with respect to NSGA-III<sup>6</sup>, MOEA/D<sup>7</sup>, IGD<sup>+</sup>-EMOA<sup>8</sup> and SMS-EMOA<sup>9</sup> (the latter for only MOPs having 3 and 4 objective functions due to its high computational cost). Results were compared using the hypervolume indicator, using the following reference points:  $(1, 1, \dots, 1)$  for DTLZ1/DTLZ1<sup>-1</sup>,  $(1, 1, \dots, 1, 21)$  for DTLZ7/DTLZ7<sup>-1</sup> and  $(2, 2, \dots, 2)$  for the remaining MOPs.

### 4.1 Parameters Settings

Since our approach and all the considered MaOEAs are genetic algorithms that use SBX and PBX, we set the crossover probability ( $P_c$ ), crossover distribution index ( $N_c$ ), mutation probability ( $P_m$ ) and the mutation distribution index ( $N_m$ ) as follows. For MOPs having 3 objective functions  $P_c = 0.9$  and  $N_c = 20$ , while for MaOPs,  $P_c = 1.0$  and  $N_c = 30$ . In all cases,  $P_m = 1/n$ , where  $n$  is the number

<sup>5</sup> The source code of IGD<sup>+</sup>-MaOEA is available at <http://computacion.cs.cinvestav.mx/~jfalcon/IGD+-MOEA.html>.

<sup>6</sup> We used the implementation available at: <http://web.ntnu.edu.tw/~tcchiang/publications/nsga3cpp/nsga3cpp.htm>.

<sup>7</sup> We used the implementation available at: <http://dces.essex.ac.uk/staff/zhang/wbofmoad.htm>.

<sup>8</sup> The source code was provided by its author, Edgar Manóatl López.

<sup>9</sup> We employed the implementation available at jMetal 4.5.

of decision variables and  $N_m = 20$ . Table 1 shows the population size, objective function evaluations (employed as our stopping criterion) and the parameter  $H$  for the generation of the set of convex weight vectors described in [3]. The population size  $N$  is equal to the number of weight vectors, i.e.,  $N = C_{m-1}^{H+m-1}$ . In all cases, the neighborhood size  $T$  of MOEA/D is set to 20.

**Table 1.** Common parameters settings

Objectives	3	4	5	6	7
Population size ( $N$ )	120	120	126	126	210
Objective function evaluations ( $\times 10^3$ )	50	60	70	80	90
Weight-vector partitions ( $H$ )	14	7	5	4	4

## 4.2 Comparison with MaOEAs Based on Convex Weight Vectors

Tables 3 and 4 show the average HV and the standard deviation (in parentheses) obtained by all the algorithms compared. The two best values among the MaOEAs are emphasized in grayscale, where the darker tone corresponds to the best value. Aiming to have statistical confidence of the results, we performed a one-tailed Wilcoxon test using a significance level of 0.05. Based on the Wilcoxon test, the symbol # is placed when IGD<sup>+</sup>-MaOEA performs better than other MaOEA in a statistically significant way.

Regarding the original DTLZ problems, in Table 3 it is shown that IGD<sup>+</sup>-MaOEA achieves the best performance in 9 out of 35 problems. Our proposed approach obtained the best HV values in DTLZ3, DTLZ5 and DTLZ6. For DTLZ7, IGD<sup>+</sup>-MaOEA obtained the second best value when using from 5 to 7 objective functions. Regarding DTLZ1, DTLZ2 and DTLZ4, our proposed approach never obtained the first or the second best HV values among the compared MaOEAs in a statistically significant manner. Nevertheless, it is worth noting that numerically, the differences in all cases are minimal. On the other hand, NSGA-III obtained the best HV values in 7 of the 35 instances, being the best in DTLZ1 and DTLZ7. Overall, IGD<sup>+</sup>-EMOA obtained the worst place in the performance rank because it only produced the best HV values only in 2 instances. Hence, we conclude that IGD<sup>+</sup>-MaOEA outperforms MOEA/D and IGD<sup>+</sup>-EMOA and is competitive with respect to NSGA-III.

**Table 2.** Average runtimes (in seconds) of IGD<sup>+</sup>-MaOEA and SMS-EMOA on the DTLZ and DTLZ<sup>-1</sup> test suites using 3 objective functions.

MaOEa	Type	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
IGD <sup>+</sup> -MaOEA	Original	55.87 s	81.66 s	42.44 s	72.80 s	54.92 s	65.31 s	76.26 s
	Minus	78.45 s	91.74 s	68.86 s	92.13 s	93.18 s	102.94 s	81.92 s
SMS-EMOA	Original	963.43 s	2144.43 s	359.28 s	1648.35 s	995.15 s	1944.93 s	1785.38 s
	Minus	1453.27 s	1868.63 s	1125.25 s	1906.52 s	1947.56 s	1950.85 s	1364.88 s

**Table 3.** Hypervolume results for the compared MOEAs on the compared DTLZ problems. We show the mean and standard deviations (in parentheses). The two best values are shown in gray scale, where the darker tone corresponds to the best value. The symbol # is placed when IGD<sup>+</sup>-MaOEA performs better in a statistically significant way.

MOP	Dim.	IGD <sup>+</sup> -MaOEA	IGD <sup>+</sup> -EMOA	NSGA-III	MOEA/D	SMS-EMOA
DTLZ1	3	9.664790e-01 (2.049666e-03)	9.740508e-01 (4.467021e-04)	9.741141e-01 (3.120293e-04)	9.740945e-01 (2.619649e-04)	9.745172e-01 (5.241259e-05)
	4	9.846496e-01 (2.656403e-03)	9.943998e-01 (9.261547e-05)	9.942231e-01 (8.570576e-04)	9.944018e-01 (6.220464e-05)	9.946409e-01 (2.134463e-05)
	5	9.881899e-01 (3.232379e-03)	9.943585e-01 (2.338311e-02)	9.986867e-01 (3.379577e-05)	9.986355e-01 (3.735697e-05)	
	6	9.906617e-01 (2.651917e-03)	9.035094e-01# (7.491169e-02)	9.996492e-01 (2.587221e-05)	9.996231e-01 (1.535746e-05)	
	7	9.948828e-01 (1.318848e-03)	9.264419e-01# (6.287378e-02)	9.999224e-01 (7.339504e-06)	9.998569e-01 (2.567104e-05)	
DTLZ2	3	7.420261e+00 (1.353052e-03)	7.421843e+00 (1.327349e-04)	7.421572e+00 (6.064709e-04)	7.421715e+00 (1.372809e-04)	7.431551e+00 (5.463841e-05)
	4	1.556161e+01 (2.748489e-03)	1.556734e+01 (4.007277e-04)	1.556646e+01 (6.681701e-04)	1.556718e+01 (2.213968e-04)	1.558874e+01 (6.349012e-05)
	5	3.166574e+01 (5.201361e-03)	3.166818e+01 (3.831826e-04)	3.166721e+01 (6.548007e-04)	3.166781e+01 (5.129480e-04)	
	6	6.373545e+01 (5.321646e-03)	6.182623e+01 (4.486397e+00)	6.373806e+01 (1.136133e-03)	6.373808e+01 (6.532194e-04)	
	7	1.278044e+02 (5.835291e-03)	1.117158e+02+ (1.213189e+01)	1.278161e+02 (1.524540e-03)	1.278230e+02 (4.937498e-04)	
DTLZ3	3	7.304310e+00 (5.416726e-01)	5.978405e+00# (2.296587e+00)	6.762070e+00# (1.512456e+00)	7.191410e+00# (9.234976e-01)	7.116381e+00# (1.038033e+00)
	4	1.554332e+01 (1.357241e-02)	1.553667e+01 (2.805291e-02)	1.426614e+01# (3.337968e+00)	1.525936e+01# (9.041126e-01)	1.557833e+01 (4.705930e-03)
	5	3.165020e+01 (9.384670e-03)	3.165404e+01 (6.820552e-03)	2.926244e+01# (5.291705e+00)	2.921654e+01# (6.617692e+00)	
	6	6.371498e+01 (1.113938e-02)	5.883028e+01# (5.646345e+00)	5.837271e+01# (1.552667e+01)	5.395689e+01# (1.319237e+01)	
	7	1.277759e+02 (1.177247e-02)	1.178341e+02# (3.658990e+00)	1.164877e+02# (2.147719e+01)	1.086977e+02# (2.778728e+01)	
DTLZ4	3	6.874113e+00 (7.238869e-01)	7.037545e+00 (7.189670e-01)	7.218780e+00 (4.062937e-01)	7.421636e+00 (1.147608e-04)	6.960992e+00 (5.030399e-01)
	4	1.495718e+01 (1.406114e+00)	1.491851e+01 (1.029726e+00)	1.540943e+01 (3.164949e-01)	1.556707e+01 (2.297960e-04)	1.506728e+01 (6.892799e-01)
	5	3.141161e+01 (5.091958e-01)	3.011363e+01# (1.320577e+00)	3.163040e+01 (1.455720e-01)	3.166733e+01 (4.792449e-04)	
	6	6.342094e+01 (8.053848e-01)	6.220439e+01# (4.109418e-01)	6.374155e+01 (5.870500e-04)	6.373585e+01 (1.078543e-03)	
	7	1.276686e+02 (5.342428e-01)	1.268979e+02# (4.641205e-01)	1.278235e+02 (5.765414e-04)	1.278246e+02 (3.325992e-04)	
DTLZ5	3	6.103250e+00 (3.206747e-04)	4.126358e+00# (1.356638e-01)	6.086240e+00# (3.462620e-03)	6.046024e+00# (2.227008e-04)	6.105419e+00 (1.265596e-05)
	4	1.195066e+01 (1.060364e-02)	8.053758e+00# (6.181680e-02)	1.176583e+01# (3.990838e-02)	1.187250e+01# (4.856384e-03)	1.200938e+01 (7.506854e-04)
	5	2.352758e+01 (5.631168e-02)	1.617222e+01# (1.916164e-01)	2.162912e+01# (9.476133e-01)	2.328373e+01# (1.640165e-02)	
	6	4.655654e+01 (1.477530e-01)	3.216498e+01# (2.350120e-01)	4.222308e+01# (1.270959e+00)	4.584961e+01# (4.179642e-02)	
	7	9.259723e+01 (2.885851e-01)	6.433872e+01# (6.900391e-01)	8.421920e+01# (2.089834e+00)	9.094108e+01# (1.339743e-01)	
DTLZ6	3	5.822452e+00 (9.468474e-02)	5.524093e+00# (8.062048e-01)	5.755154e+00# (7.832234e-02)	5.774939e+00# (8.361881e-02)	5.838678e+00 (7.196085e-02)
	4	1.141949e+01 (1.435037e-01)	9.520791e+00# (5.465663e-01)	5.969793e+00# (6.529944e-01)	1.136532e+01 (1.519071e-01)	1.112687e+01# (1.725538e-01)
	5	2.243194e+01 (2.205059e-01)	1.230783e-02# (1.960431e-02)	6.433325e-02# (1.002102e-01)	2.217372e+01# (3.778954e-01)	
	6	4.395244e+01 (4.872918e-01)	6.039732e+00# (1.208422e+01)	3.872393e+00# (7.548978e-01)	4.349163e+01# (5.731473e-01)	
	7	8.562322e+01 (8.346399e-01)	3.737526e+01# (3.051737e-01)	7.781012e+01# (2.442098e-00)	8.668146e+01 (1.610733e+00)	
DTLZ7	3	1.613138e+01 (1.102308e-01)	1.571995e+01# (7.026627e-02)	1.631926e+01 (1.253568e-02)	1.620770e+01 (1.240925e-01)	1.637100e+01 (7.629934e-02)
	4	1.435812e+01 (1.541455e-01)	1.364183e+01# (1.305431e-01)	1.462787e+01 (3.713300e-02)	1.406944e+01# (5.544544e-02)	1.483349e+01 (1.533320e-01)
	5	1.221977e+01 (5.193563e-01)	1.133320e+01# (1.223979e-01)	1.284401e+01 (3.182259e-02)	6.515913e+00# (1.170945e+00)	
	6	1.035596e+01 (4.758743e-01)	9.287520e+00# (9.704494e-02)	1.082465e+01 (7.434508e-02)	1.366732e+00# (1.894512e+00)	
	7	8.804845e+00 (3.468746e-01)	7.339032e+00# (9.787487e-02)	8.942419e+00 (5.155349e-02)	1.089167e+01 (1.867035e-01)	

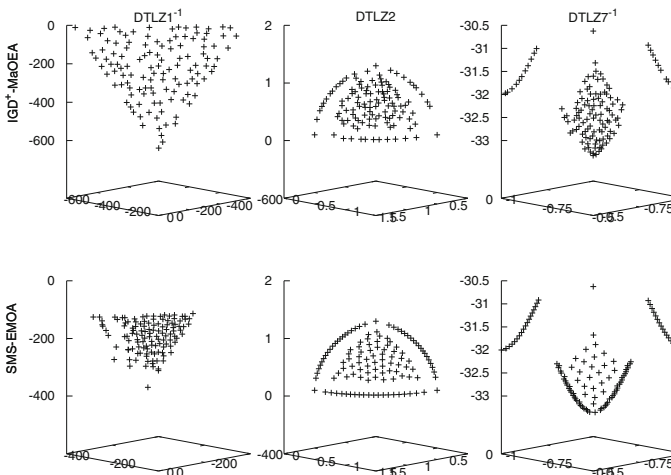
**Table 4.** Hypervolume results for the compared MOEAs on the DTLZ<sup>-1</sup> problems. We show the mean and standard deviations (in parentheses). The two best values are shown in gray scale, where the darker tone corresponds to the best value. The symbol # is placed when IGD<sup>+</sup>-MaOEA performs better in a statistically significant way.

MOP	Dim.	IGD <sup>+</sup> -MaOEA	IGD <sup>+</sup> -EMOA	NSGA-III	MOEA/D	SMS-EMOA
DTLZ1 <sup>-1</sup>	3	2.264909e+07 (8.207717e+04)	1.140466e+07# (1.217933e+06)	2.044422e+07# (2.230718e+05)	1.708422e+07# (2.776295e+05)	1.640482e+07# (1.253694e+06)
	4	1.663320e+09 (4.001511e+07)	3.783933e+07# (1.747066e+07)	6.137596e+08# (8.114743e+07)	3.671230e+08# (8.437648e+07)	1.176107e+09# (1.162071e+08)
	5	6.119188e+10 (4.760735e+09)	3.145584e+06# (6.453973e+06)	1.653440e+10# (7.395153e+09)	1.275157e+10# (5.929635e+09)	6.835890e+10# (4.577981e+10)
	6	1.040799e+12 (2.723386e+11)	5.143618e+05# (1.818714e+06)	3.525438e+11# (1.554685e+11)	6.835890e+10# (4.577981e+10)	5.582247e+11# (9.246709e+11)
	7	1.879388e+13 (7.487935e+12)	3.352615e+05# (1.083160e+06)	5.717044e+12# (2.906156e+12)	5.582247e+11# (9.246709e+11)	
DTLZ2 <sup>-1</sup>	3	1.210884e+02 (9.009171e-01)	9.369690e+01# (5.010715e+00)	1.226427e+02 (4.332124e-01)	1.241646e+02 (1.767939e-01)	1.261046e+02 (1.456397e-02)
	4	4.674859e+02 (6.158074e+00)	6.908303e+01# (2.593222e-01)	4.670265e+02# (5.036135e+00)	4.782322e+02 (3.762262e-01)	5.109249e+02 (4.731194e-01)
	5	1.655899e+03 (3.942682e+01)	1.817170e+02# (2.352582e+00)	1.529187e+03# (3.829295e+01)	1.570781e+03# (5.466206e+00)	
	6	5.470358e+03 (1.134490e+02)	4.572952e+02# (8.088396e+00)	4.188435e+03# (3.496415e+02)	3.701069e+03# (1.866271e+01)	
	7	1.926684e+04 (4.521928e+02)	1.187017e+03# (1.260695e+01)	1.321225e+04# (1.030901e+03)	1.320162e+04# (6.203137e+01)	
DTLZ3 <sup>-1</sup>	3	5.017451e+09 (1.676399e+07)	3.163373e+09# (3.448716e+08)	4.769399e+09# (4.395958e+07)	4.788299e+09# (5.251105e+07)	3.617983e+09# (1.229064e+08)
	4	5.016984e+12 (2.782494e+10)	1.858417e+11# (1.368270e+11)	3.421113e+12# (1.621812e+11)	3.382020e+12# (8.271736e+10)	2.942443e+12# (1.497601e+11)
	5	4.010397e+15 (5.4911013e+13)	2.308672e+10# (5.932196e+10)	1.418461e+15# (2.265638e+14)	2.169617e+15# (3.559794e+13)	
	6	2.671524e+18 (7.441405e+16)	6.882907e+09# (2.710629e+10)	4.952138e+17# (1.783349e+17)	7.151922e+17# (2.068326e+16)	
	7	1.792722e+21 (4.730737e+19)	3.686677e+10# (1.841504e+11)	1.374261e+20# (6.319205e+19)	8.941855e+20# (5.275602e+19)	
DTLZ4 <sup>-1</sup>	3	1.232680e+02 (5.341538e-01)	8.745995e+01# (7.308267e+00)	1.231716e+02# (3.158586e-01)	1.241412e+02 (2.261829e-01)	1.261219e+02 (1.400665e-02)
	4	4.872739e+02 (2.714648e+00)	6.889884e+01# (2.509073e-01)	4.703987e+02# (3.758543e+00)	4.774396e+02# (2.932713e-01)	5.114649e+02 (3.829142e-01)
	5	1.751991e+03 (1.604473e+01)	1.667599e+02# (4.139344e+01)	1.532427e+03# (3.367009e+01)	1.577174e+03# (3.235047e+00)	
	6	5.844499e+03 (5.546305e+01)	4.266016e+02# (3.968221e+02)	4.188345e+03# (2.845836e+02)	3.654612e+03# (4.982487e+00)	
	7	2.024392e+04 (1.637229e+02)	2.470440e+02# (8.390175e+01)	1.311381e+04# (6.546800e+02)	1.295551e+04# (5.175739e+01)	
DTLZ5 <sup>-1</sup>	3	1.189566e+02 (1.131492e+00)	1.045511e+02# (2.925727e+00)	1.212729e+02 (4.506920e-01)	1.230132e+02 (1.173182e-01)	1.248782e+02 (1.400672e-02)
	4	4.524837e+02 (6.184888e+00)	1.458893e+02# (1.837707e+01)	4.617533e+02 (3.033948e+00)	4.737665e+02 (5.201724e-01)	5.067611e+02 (3.943537e-01)
	5	1.590424e+03 (3.260130e+01)	1.247849e+03# (7.727654e+01)	1.526551e+03# (4.186892e+01)	1.532378e+03# (6.612506e+00)	
	6	5.201281e+03 (1.024733e+02)	4.775094e+03# (8.471898e+02)	3.648377e+03# (3.589604e+02)	3.670455e+03# (1.117756e+01)	
	7	1.798605e+04 (3.438881e+02)	3.663675e+03# (2.075826e+03)	1.169538e+04# (9.150133e+02)	1.287945e+04# (5.086978e+01)	
DTLZ6 <sup>-1</sup>	3	1.277596e+03 (8.980299e+00)	5.926270e+02# (4.387564e+01)	1.281204e+03 (4.388455e+00)	1.290813e+03 (6.053013e-01)	1.307600e+03 (1.645502e+00)
	4	9.344785e+03 (1.172155e+02)	7.139870e+02# (1.364398e+02)	8.894185e+03# (9.665925e+01)	8.908490e+03# (7.411574e+00)	9.489564e+03 (6.321189e+01)
	5	5.967189e+04 (9.243485e+02)	4.054599e+03# (5.178149e+02)	4.774990e+04# (2.111510e+03)	5.337501e+04# (1.101944e+02)	
	6	3.401029e+05 (5.077651e+03)	2.826444e+04# (4.152159e+03)	1.871320e+05# (4.124992e+04)	1.611984e+05# (2.134698e+02)	
	7	2.037103e+06 (1.966308e+04)	6.996351e+04# (2.447352e+02)	6.943417e+05# (1.558202e+05)	1.227654e+06# (7.772613e+03)	
DTLZ7 <sup>-1</sup>	3	2.145249e+02 (5.714409e-01)	2.121154e+02# (5.197201e+00)	2.144482e+02 (1.844494e-02)	2.144785e+02 (3.401603e-03)	2.143458e+02# (2.207311e-04)
	4	5.142917e+02 (2.116147e+00)	4.945863e+02# (1.805875e+01)	5.130456e+02# (1.613943e+00)	5.083181e+02# (1.486713e-01)	5.142100e+02# (1.152841e+00)
	5	1.199552e+03 (1.152678e+00)	4.348046e+02# (6.886537e+01)	1.190442e+03# (4.159670e+00)	6.388549e+02# (5.254422e-01)	
	6	2.741875e+03 (1.509787e+01)	7.362027e+02# (1.136842e+02)	2.691994e+03# (7.841504e+00)	9.262902e+02# (3.468054e+00)	
	7	6.176946e+03 (1.308306e+01)	1.355104e+03# (3.306126e+02)	6.016129e+03# (2.260447e+01)	1.621765e+03# (1.220737e+02)	

Table 4 shows the statistical results for the DTLZ<sup>-1</sup> test suite. IGD<sup>+</sup>-MaOEA is the best MaOEA in these problems because it presented the best HV values in 27 out of 35 instances. Its performance is more evident when tackling the instances having many objectives. In case of three-dimensional problems, it obtained the second best overall HV values, being SMS-EMOA the best optimizer. It is worth noticing that none of the MaOEAs that use convex weight vectors obtained the best HV value in any of the problems. This strongly evidences their overspecialization in MOPs whose Pareto fronts are closely related to the shape of an  $(m - 1)$ -simplex. MOEA/D obtained the second place in 16 problems and NSGA-III in 15. IGD<sup>+</sup>-EMOA is the worst MaOEA in these problems as it never obtained the best HV values nor the second best ones. Hence, it is evident that the strategy based on weight vectors for the construction of the IGD<sup>+</sup>-EMOA's reference set has a negative impact on its performance. Moreover, based on the direct comparison between IGD<sup>+</sup>-MaOEA and IGD<sup>+</sup>-EMOA, the former can be considered as a better optimizer.

### 4.3 Comparison with SMS-EMOA

From Tables 3 and 4, it is clear that SMS-EMOA outperforms IGD<sup>+</sup>-MaOEA in the DTLZ test suite and that both are competitive in the DTLZ<sup>-1</sup> instances. However, the aim of SMS-EMOA is to maximize HV and this indicator is being employed for comparison purposes which clearly favor this algorithm. Nevertheless, it is worth noting that the overall HV differences between both algorithms is not very significant. It is also worth highlighting that IGD<sup>+</sup>-MaOEA generates similar distributions to those of SMS-EMOA. This is shown in Fig. 2 where



**Fig. 2.** Pareto fronts produced by IGD<sup>+</sup>-MaOEA and SMS-EMOA for DTLZ1<sup>-1</sup>, DTLZ2 and DTLZ7<sup>-1</sup> for 3 objective functions. Each front corresponds to the median HV values.

the Pareto fronts for DTLZ2 are similar. This distribution is due to the use of the set of nondominated solutions as the reference set in the IGD<sup>+</sup>-DE algorithm. Hence, this kind of reference set is highly recommended to approximate the performance of HV-based MaOEAs using the IGD<sup>+</sup> indicator. Moreover, the average computational cost of IGD<sup>+</sup>-MaOEA is significantly lower than that of SMS-EMOA. This claim is supported by the average runtimes shown in Table 2.

## 5 Conclusions and Future Work

In this paper, we have proposed a steady-state MaOEA, called IGD<sup>+</sup>-MaOEA, that adopts an IGD<sup>+</sup>-based density estimator and Pareto dominance as its main selection criterion. Moreover, a fast method to compute the IGD<sup>+</sup> contributions is employed in order to reduce the computational cost from  $\Theta(mN^2M)$  to  $\Theta(mNM)$ , where  $m$  is the number of objective functions,  $N$  is the cardinality of the approximation set and  $M$  the size of the reference set. IGD<sup>+</sup>-MaOEA does not adopt convex weight vectors in any of its mechanisms. In consequence, the performance of IGD<sup>+</sup>-MaOEA does not strongly depend on the Pareto front shape. Our experimental results show that IGD<sup>+</sup>-MaOEA is a more general multi-objective optimizer because its performance does not degrade when solving the DTLZ<sup>-1</sup> test suite. In fact, IGD<sup>+</sup>-MaOEA is competitive with NSGA-III and outperforms MOEA/D and IGD<sup>+</sup>-EMOA in the original DTLZ test suite and it outperforms these MaOEAs in all the DTLZ<sup>-1</sup> problems. Moreover, we compared our approach with SMS-EMOA and our experimental results indicate that IGD<sup>+</sup>-MaOEA performs similarly to the former, which makes it a remarkable approach to approximate the performance of HV-based MaOEAs. As part of our future work, we are interested in producing uniformly distributed solutions for both the DTLZ and DTLZ<sup>-1</sup> test suites. Furthermore, we aim to improve the convergence results of IGD<sup>+</sup>-MaOEA in the DTLZ test problems without worsening its performance on the DTLZ<sup>-1</sup> instances.

## References

1. Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-objective Problems, 2nd edn. Springer, New York (2007). <https://doi.org/10.1007/978-0-387-36797-2>. ISBN 978-0-387-33254-3
2. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: a short review. In: 2008 Congress on Evolutionary Computation (CEC 2008), Hong Kong, pp. 2424–2431. IEEE Service Center, June 2008
3. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
4. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE Trans. Evol. Comput.* **18**(4), 577–601 (2014)
5. Lopez, E.M., Coello, C.A.C.: IGD<sup>+</sup>-EMOA: a multi-objective evolutionary algorithm based on IGD<sup>+</sup>. In: 2016 IEEE Congress on Evolutionary Computation (CEC 2016), Vancouver, Canada, 24–29 July 2016, pp. 999–1006. IEEE Press (2016). ISBN 978-1-5090-0623-9

6. Gómez, R.H., Coello, C.A.C.: Improved metaheuristic based on the  $R2$  indicator for many-objective optimization. In: 2015 Genetic and Evolutionary Computation Conference (GECCO 2015), Madrid, Spain, July 11–15 2015, pp. 679–686. ACM Press (2015). ISBN 978-1-4503-3472-3
7. Ishibuchi, H., Setoguchi, Y., Masuda, H., Nojima, Y.: Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE Trans. Evol. Comput.* **21**(2), 169–190 (2017)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
9. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
10. Fleischer, M.: The measure of pareto optima applications to multi-objective metaheuristics. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L., Deb, K. (eds.) EMO 2003. LNCS, vol. 2632, pp. 519–533. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36970-8\\_37](https://doi.org/10.1007/3-540-36970-8_37)
11. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified distance calculation in generational distance and inverted generational distance. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) EMO 2015. LNCS, vol. 9019, pp. 110–125. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-15892-1\\_8](https://doi.org/10.1007/978-3-319-15892-1_8)
12. Falcón-Cardona, J.G., Coello, C.A.C.: Multi-objective evolutionary hyper-heuristic based on multiple indicator-based density estimators. In: 2018 Genetic and Evolutionary Computation Conference (GECCO 2018), Kyoto, Japan, 15–19 July 2018. ACM Press (To be published)



# Towards Large-Scale Multiobjective Optimisation with a Hybrid Algorithm for Non-dominated Sorting

Margarita Markina and Maxim Buzdalov<sup>(✉)</sup>

ITMO University, 49 Kronverkskiy prosp., Saint-Petersburg 197101, Russia  
margaritam2706@gmail.com, mbuzdalov@gmail.com

**Abstract.** We present an algorithm for non-dominated sorting that is suitable for large-scale multiobjective optimisation. This algorithm is a hybrid of two previously known algorithms: the divide-and-conquer algorithm initially proposed by Jensen, and the non-dominated tree algorithm proposed by Gustavsson and Syberfeldt.

While possessing the good worst-case asymptotic behaviour of the divide-and-conquer algorithm, the proposed algorithm is also very efficient in practice. In our experimental study it is shown to outperform both of its parents on the majority of problem instances, both sampled uniformly from a hypercube and having a single front, with as large as  $10^6$  points and up to 15 objectives.

**Keywords:** Multiobjective optimisation · Non-dominated sorting  
Large-scale optimisation

## 1 Introduction

Many real-world optimisation problems are inherently multiobjective, that is, they require maximizing or minimizing not a single objective, but several ones, which often conflict with each other. For this reason, there are typically many optimal solutions which are incomparable and trade one objective for another. Even in the conditions that only one of these solutions must be chosen, this choice is often advised to be done lately, as the acquired knowledge of the problem can influence the preferences of the decision maker [1].

According to the tutorial [1], most general-purpose evolutionary multiobjective algorithms that do not try to incorporate the prior knowledge or user preferences belong to three categories: Pareto-based [5–7, 26], indicator-based [25], and decomposition-based [23] algorithms.

In turn, Pareto-based algorithms can be classified by how they rank or select solutions. Some of them maintain an archive of non-dominated solutions [3, 5, 13], others perform non-dominated sorting [6, 7], use domination count [9] or domination strength [26] to assign fitness values. In this paper, we consider non-dominated sorting, as many popular algorithms rely on this procedure [6, 7].



### 1.1 Non-dominated Sorting: Definition and Algorithms

From now on we assume, without loss of generality, that we need to minimize all objectives. We also explicitly state that in this paper we consider only the objective space and ignore the existence of decision variables and the questions of genotype-to-phenotype mapping. Throughout the paper, we denote as  $M$  the number of objectives.

To define non-dominated sorting, we first need to introduce the Pareto dominance relation. A point  $p$  is said to *dominate* a point  $q$ , denoted as  $p \prec q$ , if for every objective index  $i$ ,  $1 \leq i \leq M$ , it holds that  $p_i \leq q_i$ , and there exists an index  $j$  such that  $p_j < q_j$ .

Non-dominated sorting assigns *ranks* to solutions from the solution set  $P$  in the following way: every solution from  $P$  that is not dominated by any other solution from  $P$  gets rank 0, and every solution which is dominated by at least one solution of rank  $i$  gets rank  $i + 1$ . A set of all points having the same rank is often called a *front*, a *level* or a *layer*. In the work where this procedure was originally proposed [20], it was performed in  $O(N^3M)$ , where  $N$  is the population size. This was later improved to be  $O(N^2M)$  in a subsequent work that introduced the famous NSGA-II algorithm [7].

In NSGA-II, non-dominated sorting determines the computational complexity of a single iteration, as all other parts of an iteration scale better as  $N$  grows. This poses a problem either when fitness evaluation and variation operators are cheap, or when the population size  $N$  is large. As a result, there is quite a number of works dedicated to reduction of either theoretical complexity or practical running times of non-dominated sorting. Due to space limitations, we cannot consider each work in detail, nor can we cite all of them, so we just briefly describe the two prevailing directions.

The first direction aims at developing algorithms that work efficiently on inputs common to evolutionary multiobjective optimisation, but their worst-case time is still  $\Omega(N^2M)$ . A remarkable number of papers belongs to this direction [8, 11, 16, 18, 22, 24], where most of the algorithms have  $\Theta(N^2M)$  worst-case complexity, while Deductive Sort [16] can be forced to run in  $\Theta(N^3M)$  time. Among these, the best performing algorithms to date are Best Order Sort [18] and the ENS-NDT algorithm [11].

The second direction tries to reduce not only the running times, but also the computational complexity. Jensen [12] was the first to adapt the earlier result of Kung et al. [14], who solved the problem of finding non-dominated solutions in  $O(N(\log N)^{\max(1, M-2)})$ , to non-dominated sorting. This algorithm has the worst-case complexity of  $O(N(\log N)^{M-1})$ . However, this algorithm could not handle coinciding objective values, which was later corrected in subsequent works [2, 10].

We shall also note that in a different community, where this problem is called *layers of maxima*, an algorithm for  $M = 3$  was found [17], whose complexity is  $O(N(\log \log N)^2)$  with the use of randomized data structures, or  $O(N(\log \log N)^3)$  for deterministic ones. Whether this algorithm is useful in practice is still an open question.

Finally, we should mention our own recent work [15], where we tried to unify the benefits of the two directions above under the cover of a single algorithm. Our current paper builds on some of the insights of that paper and pushes these ideas towards a new level of quality.

## 1.2 Our Motivation and Contribution

Apart from a purely fundamental desire to develop efficient algorithms for hard problems, our research is motivated by a very important practical problem: the *multiobjective in-core fuel management optimisation* problem, instances of which needs to be solved during the functioning of a nuclear reactor. This problem is a hard combinatorial optimisation problem, the solutions of which need to optimise a number of contradicting objectives, such as the power received from the reactor, the amount of neutrons flying out from the reactor and more.

In a multiobjective setting, this problem attracted significant attention in the recent years. Several approaches used in practice use algorithms that employ non-dominated sorting. The reader is directed to the dissertation of Evert Schlünz for further reading [19]. An application of simulated annealing to this problem recommended numbers of samples up to  $10^5$  already in 1995 [21], which become population sizes in multiobjective settings and can nowadays rise up to  $10^6$ .

In this paper, we consider hybridising the divide-and-conquer approach, initially proposed by Jensen [12] and subsequently refined by Fortin et al. [10] and Buzdalov and Shalyto [2], and the recently proposed ENS-NDT approach by Gustavsson and Syberfeldt [11]. The latter algorithm is used to solve subproblems, which are created by the divide-and-conquer algorithm and have small enough sizes. This particular scheme resembles the production-ready implementations of the mergesort algorithm, which delegate small sub-arrays to the insertion sort.

In the case of non-dominated sorting, however, the subproblems are not completely equivalent to the initial non-dominated sorting problem. The straightforward adaptation of the ENS-NDT algorithm to solving these subproblems has rendered invalid a number of its invariants, which appear to be necessary for fast operation of the algorithm. This forced us to develop a slightly different version of ENS-NDT, which also appeared to be interesting on its own: in particular, it appeared to be more efficient than the original version for smaller values of  $M$ .

Our experiments show that our hybrid algorithm tends to outperform both its origins, namely, the ENS-NDT algorithm (including its variation developed by us) and the divide-and-conquer algorithm, especially for large problem sizes ( $N > 10^5$ ). This claim is supported by experimental results on two types of data (the “uniform hypercube”, also known as the “cloud dataset”, and the “uniform hyperplane” that consists of a single front) with  $M$  up to 15 and  $N$  up to  $10^6$ .

The rest of the paper is structured as follows. Section 2 describes the necessary details of the divide-and-conquer algorithm, as well as of ENS-NDT. Section 3 presents the modified version of the ENS-NDT algorithm, that is used in the hybrid, as well as the hybrid itself. Experiments are presented and discussed in Sect. 4. Finally, Sect. 5 concludes.

## 2 Preliminaries: The Algorithms to Hybridise

In this section, we describe the two algorithms, that we are going to use, in more detail. We start with the divide-and-conquer approach by Jensen [12], however, we use the version taken from [2] which is provably correct on every input unlike the algorithm from [12] and unlike the algorithm from [10] has a provably fast asymptotic behaviour. The second algorithm will be the non-dominated tree approach from [11], which is also known as ENS-NDT.

We assume that we perform non-dominated sorting on a set of points  $P$  from the  $M$ -dimensional objective space. Since non-dominated sorting is based entirely on the Pareto dominance relation, we can safely assume that this objective space is  $\mathbb{R}^M$ , as otherwise we can sort all points in every objective and transform objectives into integers while preserving Pareto dominance.

### 2.1 The Divide-and-Conquer Algorithm

The divide-and-conquer algorithm is based on the following observation. Assume we took some value  $q$  of the  $j$ -th objective and we split the set of points  $P$  into two sets, the set  $P_L = \{p \in P \mid p_j \leq q\}$  and  $P_R = \{p \in P \mid p_j > q\}$ . Then no point from  $P_R$  can dominate any point from  $P_L$ , because every point from  $P_L$  is less than any point from  $P_R$  in the  $j$ -th objective. So we can find the ranks for points  $P_L$  on their own, then perform the necessary comparisons between points from  $P_L$  and from  $P_R$ , always having points from  $P_L$  on the left side of the dominance relation to be checked, and, finally, refine the ranks for points from  $P_R$  by comparing them one to another.

The operations on  $P_L$  and  $P_R$  alone can be implemented in mostly the same way (again choosing an objective, splitting into halves and performing the same actions on the halves), thus allowing a recursive implementation. The operation on two arguments,  $P_L$  and  $P_R$ , is different, but it can also benefit from divide-and-conquer: if we split both sets of points, using the same value  $q$  of the same objective, into sets  $L_L, L_R, R_L$  and  $R_R$ , we can use the same procedure on pairs  $L_L$  and  $R_L, L_L$  and  $R_R, L_R$  and  $R_R$ , but we can avoid calling it on  $L_R$  and  $R_L$ .

For performance reasons, the value  $q$  is always chosen to be a median of the set of  $j$ -th objectives, and all sets are split into three parts (less than  $q$ , equal to  $q$  and greater than  $q$ ). What is more, the objective  $j$  is always chosen to be the maximum objective in which the comparison still makes sense: in HELPERA all points have the same value for every objective greater than  $j$ , and in HELPERB every  $l \in L$  dominates every  $r \in R$  in all objectives greater than  $j$ .

To complete the algorithm, one needs to provide recursion terminators. There are two types of them: the first ones trigger when one of the sets becomes too small, the second ones are called when only two meaningful objectives remain. The former case is solved straightforwardly. For the latter case, a special sweep-line algorithm is used, which is described in detail in [12].

The outline of the algorithm is given in Algorithm 1. The runtime of the sweep line subroutines is known to be  $O(n \log n)$  where  $n$  is the number of points supplied. Using this fact, and by noticing that  $\max(|P_L|, |P_R|) \leq 1/2 \cdot |P|$

---

**Algorithm 1.** The outline of the divide-and-conquer algorithm
 

---

```

function DIVIDECONQUERSORTING( $P, M$ )
  HELPERA( $P, M$ )
end function
function HELPERA( $P, m$ )
  if  $|P| \leq 1$  then
    return
  else if  $|P| = 2$  then
    Compare points in first  $m$  objectives
  else if  $m = 2$  then
    Run the sweep line subroutine
  else
     $q \leftarrow \text{MEDIAN}(\{p_m \mid p \in P\})$ 
     $\langle P_L, P_M, P_R \rangle \leftarrow \text{SPLIT}(P, m, q)$ 
    HELPERA( $P_L, m$ )
    HELPERB( $P_L, P_M, m - 1$ )
    HELPERA( $P_M, m - 1$ )
    HELPERB( $P_L \cup P_M, P_R, m - 1$ )
    HELPERA( $P_R, m$ )
  end if
end function
function HELPERB( $L, R, m$ )
  if  $|L| \leq 1$  or  $|R| \leq 1$  then
    Compare all pairs of points in first  $m$  objectives
  else if  $m = 2$  then
    Run the sweep line subroutine
  else
     $q \leftarrow \text{MEDIAN}(\{p_m \mid p \in L \cup R\})$ 
     $\langle L_L, L_M, L_R \rangle \leftarrow \text{SPLIT}(L, m, q)$ 
     $\langle R_L, R_M, R_R \rangle \leftarrow \text{SPLIT}(R, m, q)$ 
    HELPERB( $L_L, R_L, m$ )
    HELPERB( $L_R, R_R, m$ )
    HELPERB( $L_L \cup L_M, R_M \cup R_R, m - 1$ )
  end if
end function

```

---

and  $\max(|L_L| + |R_L|, |L_R| + |R_R|) \leq 1/2 \cdot (|L| + |R|)$ , one can use the Master theorem for solving recurrence relations [4] and prove the  $O(|P| \cdot (\log |P|)^{M-1})$  worst-case running time bound.

Note that even the HELPERA function solves a more general problem than non-dominated sorting: this function must cope with the existing lower bounds on ranks, arising from comparisons of points from the set  $P$  with points outside this set. From this point of view, HELPERB can be seen as the function that upgrades ranks of points from the set  $R$  by comparing them with points from the set  $L$ , whose ranks are known and will not subsequently change. It is possible to switch to other algorithms instead of HELPERA and HELPERB, for instance on smaller sizes to improve performance, if they produce the expected result.

## 2.2 The ENS-NDT Algorithm

This algorithm belongs to another family of algorithms for non-dominated sorting, termed Efficient Non-dominated Sorting, or ENS [24]. The main idea is to first sort all points lexicographically (by comparing the first objectives, move on to the second objectives if the first are equal, and continuing this way). A point cannot dominate any other point which comes before in the lexicographical order. The algorithm then traverses the points in the sorted order, while maintaining some data structure that makes comparisons with the previous points faster. For each point, first a rank query is performed against the data structure, then the point with the determined rank is added to that data structure.

Two algorithms from this family, ENS-SS and ENS-BS [24], maintain a list of already ranked points for each rank value, and for each such list the dominance check is performed, starting with the most recently added point. They are different in that ENS-SS performs the sequential search for a rank, starting with the first one, and ENS-BS performs binary search for a rank.

The ENS-NDT algorithm proposed by Gustavsson and Syberfeldt [11], instead of a list, uses a *k-d tree* (this name comes from a “*k*-dimensional tree”) to store points of each rank. To do this efficiently, the objective space is partitioned in advance: first all points are split by the  $M$ -th objective into two approximately equal parts (using the median similarly to the divide-and-conquer algorithm), then every such part is further partitioned into halves using the  $(M-1)$ -th objective and so on. After the second objective, the  $M$ -th objective comes again, as splitting in the first objective never makes sense. Every tree that stores the points will subsequently use this space partitioning scheme.

Ranking a newly inserted point is performed by running binary search for the rank, and for each rank a query to the *k-d tree* is made. The tree is traversed from the root towards the leaves. When the branching node is visited, its child corresponding to smaller objective values is always visited, while its other child is visited only if the splitting value stored in the node is not greater than the corresponding objective of the query point. Dominance comparisons in leaves are made straightforwardly, and if one of them succeeds, the procedure terminates.

The possibility of skipping entire subtrees determines the impressive performance of this algorithm. In particular, for many distributions of input points one can show a constant upper bound  $\alpha$  on the probability of entering a node child corresponding to a higher objective value. This immediately gives the upper bound of  $O(M \cdot N^{\log_2(1+\alpha)})$  per one query and  $O(M \cdot N^{1+\log_2(1+\alpha)})$  for the entire run, which is strictly faster than  $\Theta(N^2M)$  when  $\alpha < 1$ .

It is, however, possible to observe the  $\Theta(N^2M)$  running time of this algorithm on an input described by three numbers  $N$ ,  $M$  and  $k$ , where  $N$  is the number of points,  $M$  is the number of objectives and  $1 \leq k \leq M$  is the index of the “special” objective. The point  $P^{(i)}$ ,  $1 \leq i \leq N$ , of this input will have the objective value  $P_j^{(i)} = i$  for all  $j \neq k$  and  $P_k^{(i)} = N - i$ . The choice of  $k$  that degrades the performance most prominently depends on the implementation, but  $k = 1$  or  $k = M$  may be good choices. With this input, there will always be one front, and each ranking query will visit almost the entire tree.

### 3 The Proposed Algorithms

In this section, we first explain the problems which arise when adapting ENS-NDT, and many more algorithms, to serve as the replacements for HELPERA and HELPERB of the divide-and-conquer algorithm. Then we introduce ENS-NDT-ONE, a modification of ENS-NDT that uses only one k-d tree instance and is capable of working as HELPERA and HELPERB. Finally, we describe the hybrid algorithm.

#### 3.1 Loss of Monotonicity in HelperB

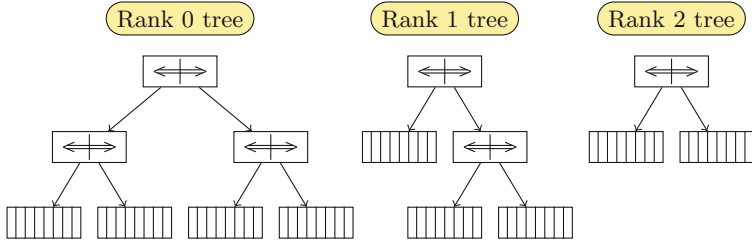
At the first glance, it should be trivial to adapt ENS-NDT, as well as other algorithms from the ENS family, to serve as HELPERB. Given the point sets  $L$  and  $R$ , one has to traverse their union in the lexicographical order. When a point from  $L$  is encountered, it is added to the data structure with its already known rank. When a point from  $R$  is encountered, one needs to query the data structure for the rank of this point, but one must not add this point to the data structure. This way, all necessary comparison between the points from  $L$  and from  $R$  will be performed.

The problem with this approach is that, in order to work correctly, the implementations of the algorithms shall stop relying on certain invariants that improve performance, and, as a result, the performance can significantly degrade. In the case of ENS-NDT and ENS-BS, this important invariant is *monotonicity*, which enables binary search. The invariant can be formulated as follows: *if the front  $k + 1$  dominates the point, then the front  $k$  also dominates it*. We now show that this invariant can be violated inside HELPERB.

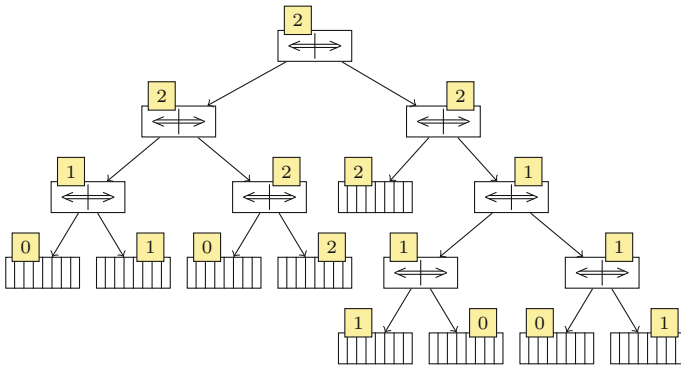
Consider points  $p_0 = (1, 3, 9, 1)$ ,  $p_1 = (1, 5, 5, 3)$ ,  $p_2 = (1, 6, 2, 4)$ ,  $p_3 = (1, 6, 7, 4)$ ,  $p_4 = (1, 6, 7, 7)$ ,  $p_5 = (1, 9, 1, 5)$ ,  $p_6 = (2, 1, 6, 7)$ ,  $p_7 = (2, 6, 5, 6)$ ,  $p_8 = (4, 8, 2, 7)$ ,  $p_9 = (5, 3, 3, 8)$ . The first call to HELPERA splits them into  $P_L = \{p_0, p_1, p_2, p_3\}$ ,  $P_M = \{p_5\}$ ,  $P_R = \{p_4, p_6, p_7, p_8, p_9\}$ . By the time HELPERB( $P_L \cup P_M, P_R, 3$ ) is called,  $p_0$  will have rank 0 and  $p_3$  will have rank 1. This call will partition these sets around the median of the third objective, which is 5, such that  $L_R = \{p_0, p_3\}$  and  $R_R = \{p_4, p_6\}$ . Once HELPERB( $L_R, R_R, 3$ ) is called, the point  $p_4$  will be found to be dominated by  $p_3$  of rank 1, but the front corresponding to rank 0 will consist only of point  $p_0$ , which does not dominate  $p_4$ . This means that there is no monotonicity anymore, and binary search for the rank is no longer valid.

The same problem makes it impossible for ENS-SS to test ranks in the increasing order. The original ENS-SS stops once a front is found that does *not* dominate the point. We now know that inside HELPERB this can result in a preliminary termination. The valid strategy in these conditions is to test ranks in the *decreasing* order, and to stop once the front is found that *does* dominate the point. Again, this reduces the performance of the ENS-SS algorithm, as, unlike the original version, most fronts are now traversed to their very end.

We overcome this problem by adapting ENS-NDT in such a way that it does not have to rely on monotonicity of fronts, while retaining a decent performance.



**Fig. 1.** The way ENS-NDT uses its k-d trees. Each tree is associated with a rank value, and stores only points with that rank.



**Fig. 2.** The way ENS-NDT-ONE uses its only k-d tree. All points reside in the same tree, and each node additionally stores the maximum rank of a point in its subtree.

### 3.2 The ENS-NDT-ONE Algorithm

We propose a new algorithm for non-dominated sorting, termed ENS-NDT-ONE, that is based on ENS-NDT, however, unlike its ancestor, it does not maintain separate trees for storing points of different ranks. Instead, all points now reside in a single k-d tree.

One of the performance advantages of ENS-NDT is that, while completing the rank query for a point  $p$ , once a point in a tree is found to dominate  $p$ , it is possible to quit that tree immediately, since no more points from that tree can influence the rank of  $p$ . This is not so in ENS-NDT-ONE, as there can be points with the same or a greater rank compared to the updated rank of  $p$ .

To compensate for this performance loss, we propose to store in each tree node the value of the maximum rank among all points in the subtree rooted at this node. With this information in hand, we can now refrain from visiting the node (and all nodes in its subtree) if the maximum rank is less than the current rank of the point  $p$  being queried. The maximum ranks of nodes are also straightforwardly updated on insertion of a point. See Figs. 1 and 2 for comparison of the principles beneath ENS-NDT and ENS-NDT-ONE.

The worst-case running time of this algorithm is  $\Theta(N^2M)$ , which is demonstrated by the same construction as we used for ENS-NDT. However, for many cases the running time is much smaller. For instance, if the points to be sorted are sampled uniformly from a hypercube  $[0; 1]^M$ , then we can see that  $\Theta(N)$  points will have a probability of at most  $1/2$  to enter both children of every particular branching node of the tree. This immediately gives the  $O(MN^{1+\log_2(3/2)}) \approx O(MN^{1.585})$  runtime bound. Similar results can be shown for other distributions, and the bounds can be further reduced by considering the distribution of ranks.

### 3.3 The Hybrid Algorithm

We can now formulate the hybrid algorithm. We take the divide-and-conquer algorithm as a basis, however, before we enter the main parts of HELPERA or HELPERB, we check whether the subproblem is *small enough*. If it is, we use the ENS-NDT-ONE algorithm to solve this subproblem. Since ENS-NDT-ONE is immune to the features of these subproblems, such as the loss of monotonicity, the resulting algorithm will always produce correct results.

More formally, we define, for every number of objectives, a threshold which signifies that every subproblem with this number of objectives and the size below the threshold should be delegated to ENS-NDT-ONE. For HELPERA, the size of the problem is the size of the set  $P$ , while for HELPERB this is the sum of sizes of the sets  $L$  and  $R$ .

We shall note that, since we define thresholds to be constants, the asymptotic estimation of the running time of this algorithm is still  $O(N(\log N)^{M-1})$ . However, we note that more careful choices for thresholds, that possibly depend on the number of objectives or on other properties of the subproblems, may possibly result in smaller runtime bounds. Due to the complexity of this issue, including strong dependency on inputs, we leave this for possible future work.

## 4 Experiments and Discussion

All mentioned algorithms were implemented in Java within the same algorithmic framework, which enabled sharing large code amounts between the algorithms. These implementations are available on GitHub<sup>1</sup> along with performance plots.

All the algorithms, except for the original divide-and-conquer algorithm, feature parameters that influence their performance. In particular, the ENS-NDT algorithm and its derivatives have the *split threshold* parameter which regulates the maximum possible size of the terminal node. In [11] this parameter was fixed to the value of 2, however, our preliminary experiments found that the value of 8 brings generally better performance. This difference can be attributed to the differences in implementations. We used the split threshold of 8 for ENS-NDT, ENS-NDT-ONE, as well as in the ENS-NDT-ONE part of the hybrid algorithm.

<sup>1</sup> <https://github.com/mbuzdalov/non-dominated-sorting/releases/tag/v0.1>.



**Table 1.** Average running times of the algorithms in seconds. The smallest running time, for each category, is marked grey. All standard deviations are less than 2%.

$N$	$M$	Divide&Conquer		ENS-NDT		ENS-NDT-ONE		Hybrid	
		hypercube	hyperplane	hypercube	hyperplane	hypercube	hyperplane	hypercube	hyperplane
$5 \cdot 10^5$	3	1.52	0.85	1.95	0.73	1.66	0.76	1.17	0.67
	$10^6$	2.82	1.60	5.25	1.61	4.25	1.65	2.63	1.50
$5 \cdot 10^5$	5	22.7	16.6	8.31	2.01	6.25	2.22	6.43	4.68
	$10^6$	45.2	33.0	26.3	5.22	18.2	5.82	17.2	12.8
$5 \cdot 10^5$	7	89.6	55.1	17.1	6.96	15.5	6.78	9.29	7.02
	$10^6$	191.5	120.2	55.4	19.4	46.1	18.9	26.8	20.1
$5 \cdot 10^5$	10	197.7	99.9	27.6	15.9	36.7	17.7	14.5	11.5
	$10^6$	478.8	228.6	84.8	48.1	104.8	55.0	41.0	33.0
$5 \cdot 10^5$	15	190.0	116.1	40.8	23.0	62.1	25.9	22.6	15.7
	$10^6$	587.9	337.5	135.4	76.3	206.8	85.4	64.5	46.0

The hybrid algorithm also depends on the switch-to-tree threshold values. Based on our preliminary investigations, we chose this threshold for three objectives to be 100, and for more than three objectives to be 20 000.

We have investigated the performance of all these algorithms, including the ENS-NDT-ONE alone, on several artificial inputs. We used two types of data. The first one is the “uniform hypercube”, which is also known as the “cloud dataset” in the literature, where points are sampled uniformly at random from the  $[0; 1]^M$  hypercube. The second one is the “uniform hyperplane”, where points are sampled uniformly at random from the piece of a hyperplane, such that all coordinates are non-negative and sum up to 1. The following values of  $N$  were tested:  $\{1, 2, 5\} \times \{10, 10^2, 10^3, 10^4, 10^5\}$  and  $10^6$ . We considered  $M$  to be from the set  $\{3, 5, 7, 10, 15\}$ , which covers the most widely used range.

For every input configuration, 10 instances were created with different but fixed random seeds. We measured the total times on all these instances and divided them by 10 to achieve an approximation of the average time. The time measurements were done using the Java Microbenchmark Harness suite with one warmup iteration of at least 6 seconds, which was enough for the entire bytecode to be translated to the native code, and one measurement iteration of at least one second. For each pair of algorithm and input, five measurement runs were conducted. A high-performance server with AMD Opteron™ 6380 processors and 512 GB of RAM was used, and the code was run with the OpenJDK virtual machine 1.8.0\_141.

The already mentioned GitHub release features the plots of the running times, which could not fit in this paper due to space restrictions. In Table 1, we show only the average results for two largest  $N$ ,  $5 \cdot 10^5$  and  $10^6$ . One can see that the hybrid algorithm wins in all cases except for  $M = 5$  and the hyperplane instance of  $M = 7$ . One more insight is that ENS-NDT-ONE runs faster than ENS-NDT on hypercube instances with  $M \leq 7$ , which means that the maximum subtree rank heuristic is indeed efficient. The implementation constant of ENS-NDT-ONE seems to be slightly larger, however.

## 5 Conclusion

We proposed a highly efficient algorithm for non-dominated sorting based on hybridisation of two previously known algorithms, the divide-and-conquer algorithm by Jensen and the non-dominated tree (ENS-NDT) by Gustavsson and Syberfeldt. It typically outperforms both of its parents on large population sizes, except for certain ranges of population sizes in several dimensions. Our modification of ENS-NDT is also of interest, as it can outperform the original ENS-NDT.

We are probably the first to report results on  $10^6$  solutions. Some industrial applications of evolutionary multiobjective optimisation already require population sizes that are this large. As divide-and-conquer algorithms often offer parallelisation benefits, and our algorithm is not an exception, we hope to get further speed-ups by adapting our algorithm to multicore computers.

The optimal choice of thresholds to decide when to switch to ENS-NDT is an open and difficult question. We expect that adaptation of thresholds while running the algorithm can overcome this issue.

**Acknowledgment.** We would like to acknowledge the support of this research by the Russian Scientific Foundation, agreement No. 17-71-20178.

## References

1. Brockhoff, D., Wagner, T.: GECCO 2016 tutorial on evolutionary multiobjective optimization. In: Proceedings of Genetic and Evolutionary Computation Conference Companion, pp. 201–227 (2016)
2. Buzdalov, M., Shalyto, A.: A provably asymptotically fast version of the generalized jensen algorithm for non-dominated sorting. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) PPSN 2014. LNCS, vol. 8672, pp. 528–537. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10762-2\\_52](https://doi.org/10.1007/978-3-319-10762-2_52)
3. Coello Coello, C.A., Toscano Pulido, G.: A micro-genetic algorithm for multiobjective optimization. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 126–140. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44719-9\\_9](https://doi.org/10.1007/3-540-44719-9_9)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press, Cambridge (2001)
5. 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)
6. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Trans. Evol. Comput.* **18**(4), 577–601 (2013)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
8. Fang, H., Wang, Q., Tu, Y.C., Horstemeyer, M.F.: An efficient non-dominated sorting method for evolutionary algorithms. *Evol. Comput.* **16**(3), 355–384 (2008)

9. Fonseca, C.M., Fleming, P.J.: Nonlinear system identification with multiobjective genetic algorithm. In: Proceedings of the World Congress of the International Federation of Automatic Control, pp. 187–192 (1996)
10. Fortin, F.A., Grenier, S., Parizeau, M.: Generalizing the improved run-time complexity algorithm for non-dominated sorting. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 615–622. ACM (2013)
11. Gustavsson, P., Syberfeldt, A.: A new algorithm using the non-dominated tree to improve non-dominated sorting. *Evol. Comput.* **26**(1), 89–116 (2018)
12. Jensen, M.T.: Reducing the run-time complexity of multiobjective EAs: the NSGA-II and other algorithms. *IEEE Trans. Evol. Comput.* **7**(5), 503–515 (2003)
13. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the pareto archived evolution strategy. *Evol. Comput.* **8**(2), 149–172 (2000)
14. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. *J. ACM* **22**(4), 469–476 (1975)
15. Markina, M., Buzdalov, M.: Hybridizing non-dominated sorting algorithms: divide-and-conquer meets best order sort. In: Proceedings of Genetic and Evolutionary Computation Conference Companion, pp. 153–154 (2017)
16. McClymont, K., Keedwell, E.: Deductive sort and climbing sort: new methods for non-dominated sorting. *Evol. Comput.* **20**(1), 1–26 (2012)
17. Nekrich, Y.: A fast algorithm for three-dimensional layers of maxima problem. In: Dehne, F., Iacono, J., Sack, J.-R. (eds.) WADS 2011. LNCS, vol. 6844, pp. 607–618. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22300-6\\_51](https://doi.org/10.1007/978-3-642-22300-6_51)
18. Roy, P.C., Islam, M.M., Deb, K.: Best Order Sort: a new algorithm to non-dominated sorting for evolutionary multi-objective optimization. In: Proceedings of Genetic and Evolutionary Computation Conference Companion, pp. 1113–1120 (2016)
19. Schlünz, E.B.: Multiobjective in-core fuel management optimisation for nuclear research reactors. Ph.D. thesis, Stellenbosch University, December 2016
20. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.* **2**(3), 221–248 (1994)
21. Stevens, J., Smith, K., Rempe, K., Downar, T.: Optimization of pressurized water reactor shuffling by simulated annealing with heuristics. *Nucl. Sci. Eng.* **121**(1), 67–88 (1995)
22. Wang, H., Yao, X.: Corner sort for pareto-based many-objective optimization. *IEEE Trans. Cybern.* **44**(1), 92–102 (2014)
23. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
24. Zhang, X., Tian, Y., Cheng, R., Jin, Y.: An efficient approach to nondominated sorting for evolutionary multiobjective optimization. *IEEE Trans. Evol. Comput.* **19**(2), 201–213 (2015)
25. 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). [https://doi.org/10.1007/978-3-540-30217-9\\_84](https://doi.org/10.1007/978-3-540-30217-9_84)
26. 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)



# Tree-Structured Decomposition and Adaptation in MOEA/D

Hanwei Zhang<sup>1,2</sup> and Aimin Zhou<sup>1</sup>(✉)

<sup>1</sup> Shanghai Key Laboratory of Multidimensional Information Processing,  
Department of Computer Science and Technology, East China Normal University,  
Shanghai, China

amzhou@cs.ecnu.edu.cn

<sup>2</sup> CNRS-IRISA, Rennes, France

hanwei.zhang@irisa.fr

**Abstract.** The *multiobjective evolutionary algorithm based on decomposition (MOEA/D)* converts a *multiobjective optimization problem (MOP)* into a set of simple subproblems, and deals with them simultaneously to approximate the Pareto optimal set (PS) of the original MOP. Normally in MOEA/D, a set of weight vectors are predefined and kept unchanged during the search process. In the last few years, it has been demonstrated in some cases that a set of predefined subproblems may fail to achieve a good approximation to the Pareto optimal set. The major reason is that it is usually unable to define a proper set of subproblems, which take full consideration of the characteristics of the MOP beforehand. Therefore, it is imperative to develop a way to adaptively redefine the subproblems during the search process. This paper proposes a *tree-structured decomposition and adaptation (TDA)* strategy to achieve this goal. The basic idea is to use a tree structure to decompose the search domain into a set of subdomains that are related with some subproblems, and adaptively maintain these subdomains by analyzing the search behaviors of MOEA/D in these subdomains. The TDA strategy has been applied to a variety of test instances. Experimental results show the advantages of TDA on improving MOEA/D in dealing with MOPs with different characteristics.

## 1 Introduction

Decomposition based *multiobjective evolutionary algorithms (MOEAs)* [1–3] have recently been attracting much attention for dealing with *multiobjective optimization problems (MOPs)*. A main difference between the decomposition based MOEAs and the other two major MOEA paradigms, i.e., the Pareto domination based approaches [4, 5] and the indicator based approaches [6–8], lies in the environmental selection. To differentiate solutions in the environmental selection, the Pareto domination based approaches use the Pareto domination relationship and a density estimation strategy to define a complete ranking order of the solutions, and the indicator based approaches utilize a performance indicator to score a

solution or a subpopulation. Since the decomposition based approaches convert an MOP into a set of subproblems, the environmental selection is implemented for each subproblem [9], i.e., if the subproblem is a scalar-objective problem, the subproblem objective value can be directly used to do selection; if the subproblem is a multiobjective problem, the above two selection approaches can be used. For both scalar-objective and multiobjective subproblems, they are tackled simultaneously.

The *multiobjective evolutionary algorithm based on decomposition (MOEA/D)* is a typical decomposition based MOEA [10]. The combination of the found optimal solutions of the subproblems will constitute an approximation to the Pareto optimal set of the original MOP. A variety of methods have been proposed to decompose an MOP into a set of subproblems in MOEA/D [10, 11]. Let  $\min_{x \in \Omega} F(x) = \{f_1(x), \dots, f_m(x)\}$  be a general MOP, where  $x$  is a decision variable vector,  $\Omega$  denotes the feasible region of the search space, and  $f_i(x)$  is the  $i$ th objective. This paper considers the Tchebycheff approach [10] that defines a parameterized scalar-objective subproblem  $g(x|w, z^*) = \max_{1 \leq j \leq m} w^j |f_j(x) - z_j^*|$  with reference point  $z^* = (z_1^*, \dots, z_m^*)$  and weight vector  $w = (w_1, \dots, w_m)$ , which is required that  $w_i \geq 0$ , for  $i = 1, \dots, m$ , and  $\sum_{i=1}^m w_i = 1$ . It is clear that all the weight vectors are from an  $(m - 1)$ -dimensional simplex. For simplicity, we use  $g^i$  to denote  $g(x|w^i, z^*)$  in the sequel.

The approximation quality is determined by the weight vectors and the reference point. In different MOEA/D variants, the reference point is adaptively updated by the best solutions found so far and this strategy works well. However, when the Pareto Front (PF) shape of an MOP is complicated (e.g. disconnected, ill-scaled), the uniform sampling strategy may fail to find a good approximation of the PF. A natural way to deal with this problem is to adaptively adjust the weight vectors during the search process. Several works have been done along this direction [12–16]. This paper proposes a new way to adjust the weight vectors dynamically, called *tree-structured decomposition and adaptation (TDA)*, and name MOEA/D with TDA as MOEA/D-TDA. The basic idea is to maintain a tree structure to decompose the search domain into a set of subdomains that are related to some subproblems, and adaptively adjust the subdomains to find the Pareto optimal solutions of an MOP. The search domain, in both the objective and decision spaces, is recursively decomposed into a set of simplexes through a set of weight vectors in the weight space. The simplex is regarded as a basic unit of the search process. Each simplex is represented as a node in the tree structure. The sparseness of the simplexes is measured along the search process. According to the measurement, some new simplexes are added in the sparse areas and some old ones are removed from the dense areas. And a tree structure makes it efficient for these operations. Since it is hard to find a good approximation to both the PF and the PS, we utilize two populations: an internal working population that approximates the PS and an external archive that approximates the PF.

The rest of the paper is organized as follows. Section 2 presents the proposed method in detail. Sections 3 and 4 study the major components in the new approach. Section 5 gives the experimental studies. Finally, the paper is concluded in Sect. 6 with some suggestions for future work.

## 2 The Proposed Method

In this section, we introduce the strategy *tree-structured decomposition and adaptation (TDA)* in details. Firstly, the framework of the algorithm is given. After that, we explain how to partition the decision, the objective, and the weight domains into some subdomains by using a tree structure named decomposition tree. Last but not least, we depict the approach to adaptively change the weight vectors by adding or removing subdomains according to the search behaviors.

### 2.1 Algorithm Framework

MOEA/D-TDA maintains a set of scalar-objective subproblems, and the  $i$ th ( $i = 1, \dots, N$ ) subproblem is with (a) its weight vector  $w^i$  and objective function  $g^i(x)$ , (b) its current solution  $x^i$  and its objective vector  $F^i = F(x^i)$ , and (c) the index set of its neighboring subproblems,  $B^i$ , of which the weight vectors are closest to  $w^i$ .

With TDA, the domains are decomposed recursively using a Tree-structured  $DT = \{D^k\}$ ,  $k = 1, \dots, K$ , which is called the decomposition tree. Each node in  $DT$  represents a subdomain and is defined as  $D^k = \langle p, O, W, E \rangle$  where  $p$  is the index of its parent node,  $O$  contains the indices of its child nodes,  $W$  contains the weight vectors of the subproblems that are directly related to the domain, and  $E$  is the set of all the edges of the simplex that forms the subdomain. It should be noted that

- Each domain is with  $m$  subproblems, i.e.,  $|W| = m$ , in which  $m$  is the number of objectives.
- $O = \emptyset$  if  $D^k$  is a leaf node or  $|O| = 2^{m-1}$  otherwise.
- $\bigcup_{D \in DT} D.W$  contains the weight vectors of all the subproblems.
- Let  $N = |\bigcup_{D \in DT} D.W|$  and  $K$  be the number of subproblems and the number of subdomains respectively.  $D.W$  denotes the weight vectors for the domain  $D$ . Neither  $N$  nor  $K$  is fixed throughout the run, and we discuss this in the next section.

---

#### Algorithm 1. Main Framework of MOEA/D-TDA

---

```

1 Initialize a decomposition tree  $DT$ .
2 Initialize all the subproblems according to the weight vectors  $\bigcup_{D \in DT} D.W$  with  $DT$ .
3 Initialize the reference point  $z^* = (z_1^*, \dots, z_m^*)$  as  $z_j^* = \min_{i=1, \dots, N} f_j(x^i)$  for  $j = 1, \dots, m$ .
4 while not terminate do
5     Update the decomposition tree  $DT$ .
6     Update the neighborhood structure according to the weight vectors.
7     foreach subproblem  $i \in \{1, \dots, N\}$  do
8         Generate a new solution  $y = \text{Generate}(i)$ .
9         Update the reference point  $z^*$  by resetting  $z_j^* = f_j(y)$  if  $z_j^* \geq f_j(y)$  for  $j = 1, \dots, m$ .
10        Update the population by the new trial solution  $y$ .
```

---

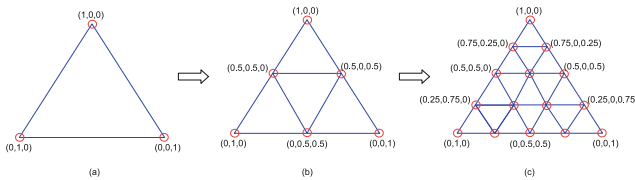
MOEA/D-TDA also needs to maintain a reference point  $z^* = (z_1^*, \dots, z_m^*)$ . The main framework of MOEA/D-TDA is shown in Algorithm 1. We would make the following comments on the framework.

- In *Line 2*, the weight vectors are generated in the decomposition tree initialization process. Each solution is initialized by a randomly sampled point from  $\Omega$  and is assigned to subproblems according to the weight vectors.
- The reference point  $z^*$  is initialized in *Line 3* and updated in *Line 9*.
- In *Line 4*, a maximum number of generations is used as the termination condition.
- In *Line 6*, the neighborhood structure needs to be updated since the weight vectors may change in *Line 5*.
- In *Line 7*, each subproblem is selected for offspring generation and population update in each generation.

Basically, the above algorithm framework is following the original MOEA/D framework [17]. *Line 8* generates a new solution  $y$ . There are various ways to implement it. It should be noted that, in this paper, this procedure is the same generation procedure as in MOEA/D-DE [17]. *Line 10* tries to replace one solution in the current population by the trial solution  $y$ . In this paper, we use the approach defined in [18]. In the next section, we emphasize the decomposition tree initialization in *Line 1*, the decomposition tree update in *Line 5*.

### 2.2 Domain Decomposition

Let  $N_0$  be the desired population size. The domain decomposition process starts by setting the decomposition tree as the weight domain, then recursively decomposes the subdomains until the number of weight vectors exceeds  $N_0$ . Figure 1 illustrates the decomposition tree initialization process in the case of tri-objective problems. Each edge of the weight simplex is cut into two equal-length edges and some subdomains with equal size are generated. It can be deduced easily that when decomposing a subdomain,  $2^{m-1}$  new subdomains and  $2^{m-1} - 1$  new weight vectors are generated.



**Fig. 1.** An illustration of decomposition tree initialization in the case of tri-objective problems.

Let  $e_i = (e_{i,1}, e_{i,2}, \dots, e_{i,m})$  denote the unit vector in the coordinate system where  $e_{i,j} = 0$  if  $j \neq i$ , and  $e_{i,i} = 1$ . The decomposition tree initialization process is shown in Algorithm 2.

---

**Algorithm 2.** Decomposition Tree Initialization

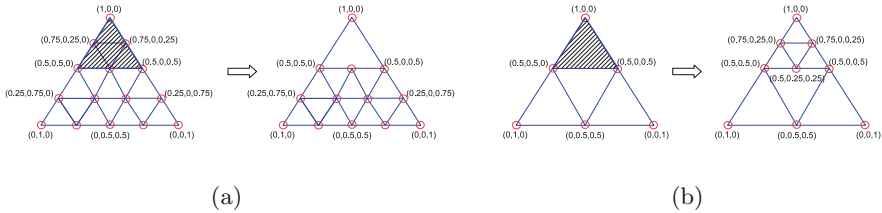
---

- 1 Set  $DT = \{D^1\}$  where  $D^1.p = 0$ ,  $D^1.O = \emptyset$ , and  $D^1.W = \{e_1, \dots, e_m\}$ .
  - 2 **while**  $|\bigcup_{D \in DT} D.W| < N_0$  **do**
  - 3     Let  $D \in DT$  be a randomly chosen leaf node that has the lowest depth.
  - 4     Decompose domain  $D$  into a set of subdomains, set the child nodes of  $D$  be these subdomains, and add them to  $DT$ .
- 

In Algorithm 2, we define the depth of the root node as 1, and the depth of a child node is the depth of its parent node plus 1. *Line 3* makes sure that it is always a leaf node, which is most closet to the root node, to be decomposed. It should also be noted that in *Line 3*, to keep the population size, not all the leaf nodes with the same depth will be decomposed.

### 2.3 Domain Adaptation

As discussed previously, MOEA/D can obtain a set of well-distributed solutions by setting proper weight vectors. To this end, we adaptively change the weight vectors by adding some nodes in sparse areas and removing some nodes in dense areas. This idea is implemented in TDA by adding some new subdomains and removing some old nodes respectively.



**Fig. 2.** An illustration of (a) deleting old domains and (b) inserting new domains in the case of tri-objective problems.

Figure 2(a) illustrates, in the case of tri-objective problems, how to remove a subdomain. It should be noted that not all subdomains can be removed, and a removable subdomain is the one that contains only one level of child subdomains. Once a subdomain is removed, some of the corresponding weight vectors and subproblems are removed as well. Since a weight vector may be shared by several subdomains, only the unused weights can be removed. Figure 2(b) illustrates, in the case of tri-objective problems, how to add some subdomains. Some new weight vectors and subproblems are added as well.

Let  $d(D)$  be a function that measures the search behavior, which is the density in this paper, of subdomain  $D$ . We assume a lower  $d(D)$  value denotes that subdomain  $D$  is dense while a higher  $d(D)$  value denotes that subdomain  $D$  is sparse. The decomposition tree adaptation process is shown in Algorithm 3.



---

**Algorithm 3.** Decomposition Tree Adaptation

---

```

1 Let  $D_1 \subset DT$  be the set of removable nodes, and sort them by an increasing order of their
   $d(\cdot)$  values.
2 Let  $D_2 \subset S$  be the set of leaf nodes, and sort them by a decreasing order of their  $d(\cdot)$  values.
3 Set  $d_1 = first(D_1)$  and  $d_2 = first(D_2)$ .
4 while  $|D_1| > 0$  and  $|D_2| > 0$  and  $d(d_1) < d(d_2)$  do
5   Delete node  $d_1$  from DT, and set  $D_1 = D_1 \setminus \{d_1\}$ .
6   Decompose  $d_2$ , add new nodes to DT, and set  $D_2 = D_2 \setminus \{d_2\}$ .
7   Remove the parent node of  $d_2$  from  $D_1$  by setting  $D_1 = D_1 \setminus \{parent(d_2)\}$ .
8   Resort  $D_1$  and  $D_2$ , set  $d_1 = first(D_1)$  and  $d_2 = first(D_2)$ .

```

---

We would like to make some comments on the algorithm.

- The process stops in *Line 4* when there is no removable subdomain to delete, or no subdomain to decompose, or the density of the subdomain to delete is bigger than that of the one to decompose. The target is to make all subdomains have the same density values and thus to obtain a set of well-distributed final solutions.
- In each step, one subdomain is removed and one is decomposed. The target is to keep a stable population size although the number of added weight vectors may not be the same as the number of removed weight vectors.
- When a subdomain is deleted from  $DT$  in *Line 5*, the corresponding weight vectors and subproblems are deleted as well if the weight vectors are not used by other subdomains.
- When a subdomain is added to  $DT$  in *Line 6*, some new weight vectors and subproblems are also added. Each new subproblem is initialized with a randomly generated solution and with infinite objective values.
- In *Line 7*, the parent node of  $d_2$  is removed from  $D_1$  to prevent the newly added subdomains to be deleted again in the next steps.
- $d(\cdot)$  is a function to measure subdomain by measuring its density. How to define the function will be discussed later.

### 3 Subdomain Measurement

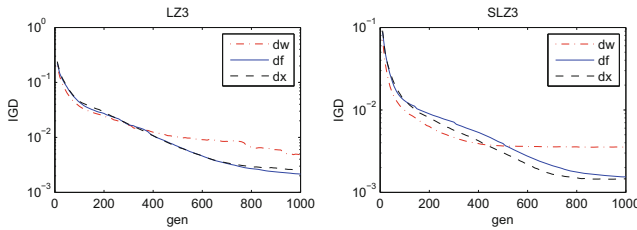
To implement MOEA/D-TDA, a key issue is on how to measure the subdomain. Density might be a good choice in this case. We define the density of a simplex as follows.

$$\begin{aligned}
 df(s) &= \sum_{w_i w_j \in s.E} \|F(x^i) - F(x^j)\|_2 \\
 dx(s) &= \sum_{w_i w_j \in s.E} \|x^i - x^j\|_2 \\
 dw(s) &= \sum_{w_i w_j \in s.E} \|w^i - w^j\|_2
 \end{aligned} \tag{1}$$

where  $w_i w_j$  is an edge in the simplex  $s$ , and  $\|\cdot\|_2$  denotes the  $L_2$  norm,  $x^i$  and  $x^j$  are two solutions with  $w_i$  and  $w_j$  respectively.  $df(\cdot)$ ,  $dx(\cdot)$ , and  $dw(\cdot)$  measure the density of the subdomain in the objective space, in the decision space, and in the weight space respectively. It is clear that, if  $dw(\cdot)$  is used, MOEA/D-TDA

is actually the original MOEA/D because the initial weight vectors are well-distributed; otherwise if  $df(\cdot)$  or  $dx(\cdot)$  is used, MOEA/D-TDA will emphasize the search behavior in either the objective space or the decision space. If we attach more importance to the objective space, we name this version as TDA-F while TDA-X for the version underlines the decision space.

It should be noted that the above measurements are just examples and other subdomain measurements could be defined and used in MOEA/D-TDA. In following, we study the influence of the three subdomain measurements defined in (1). We choose two problems, i.e., LZ3 [17] and its variant SLZ3<sup>1</sup>, as examples in the study. The parameter settings are as follows: the population size  $N = 300$ , the number of decision variables  $n = 30$ , and the neighborhood size  $T = 20$ . The parameters in offspring reproduction are  $\delta = 0.9$ ,  $F = 0.5$ , and  $\eta = 20$ . The maximum FE number is  $3 \times 10^5$  for all the algorithms. Each algorithm is executed in each problem with 50 independent runs. For quantitative comparison, the *Inverted Generational Distance (IGD)* metric [19] is used and the reference point set has 1000 points.



**Fig. 3.** The mean IGD metric values versus generations for MOEA/D-TDA with different density measurements on LZ3 and SLZ3.

The experimental results are shown in Fig. 3. From the figure, we can conclude that (a) it is hard to balance the population diversity in both the decision and the objective spaces if the diversity maintains strategy is used only in one space, and (b) in some cases, to keep the population diversity in the objective, it is necessary to keep the population diversity in the decision space.

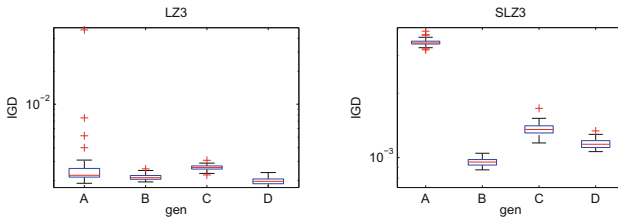
## 4 External Population

As discussed in the above section, in order to balance population diversity in both the objective and decision spaces we need an external population (archive) to MOEA/D-TDA. A step to maintain the external population should be added to Algorithm 1 after *Line 10*. The two populations are with different usages: the

<sup>1</sup> The LZ test instances are scaled by replace the original  $f_1(x)$  function by  $0.1f_1(x)$ .

*internal population* tries to approximate the PS in the decision space, and the *external population* tries to approximate the PF in the objective space.

In the new approach, the offspring generation operation is based on the internal population, and the density measurement  $dx$  is applied to tune the subproblems and thus to maintain the diversity of the internal population. The external population is initialized as the internal population. The newly generated solutions are used to update the external population. The solutions in the external population will not be used for offspring generation, but they will be output as the approximation result. It should be noted that any archive strategy can be integrated into MOEA/D-TDA. In the following experiment, we consider the following strategies: (a) NDS: the nondomination sorting scheme from NSGA-II [5], (b) HBS: the hypervolume based selection from SMS-EMOA [8], and (c) DBS: the population maintain strategy introduced in this paper with the density measurement  $df(\cdot)$ .



**Fig. 4.** Box-plots of IGD values of the final results obtained by the four algorithms over 50 independent runs.

To demonstrate the contribution of external population the corresponding maintain strategies, we empirically compare the following four algorithms on LZ3 and SLZ3: (a) A: MOEA/D-TDA with  $dw$  and without an external population, i.e., the original MOEA/D, (b) B: MOEA/D-TDA with  $dx$  and with an external population maintained by NDS, (c) C: MOEA/D-TDA with  $dx$  and with an external population maintained by HBS, and (d) D: MOEA/D-TDA with  $dx$  and with an external population maintained by DBS.

Figure 4 shows the box-plots of the IGD metric values of the final results obtained by the four algorithms. From the figure, we can see that by using external population, the approximation quality can be significantly improved. Comparing the three external population maintain strategies, the experimental results suggest that MOEA/D-TDA with NDS performs the best. The reason might be that it is more suitable to approximate the PF especially when the PF is scaled.

**Table 1.** The mean and standard deviation of IGD values obtained by five algorithms over 50 runs on the LZ and SLZ suites.

LZ1	TDA-X	$1.407e - 03_{1.681e-05}$ [4]	SLZ1	TDA-X	<b><math>8.847e - 04_{1.384e-05}</math></b> [1]
	TDA-F	$1.403e - 03_{1.288e-05}$ [3]		TDA-F	$9.476e - 04_{1.955e-05}$ [2]
	DE	<b><math>1.280e - 03_{3.029e-06}</math></b> [1]		DE	$3.339e - 03_{1.113e-05}$ [4]
	M2M	$1.391e - 03_{5.464e-05}$ [2]		M2M	$3.411e - 03_{1.655e-04}$ [5]
	AWA	$1.809e - 03_{7.184e-05}$ [5]		AWA	$1.017e - 03_{1.511e-05}$ [3]
LZ2	TDA-X	<b><math>2.243e - 03_{1.688e-04}</math></b> [1]	SLZ2	TDA-X	<b><math>9.417e - 04_{2.946e-05}</math></b> [1]
	TDA-F	$2.539e - 03_{1.767e-04}$ [3]		TDA-F	$1.117e - 03_{7.294e-05}$ [2]
	DE	$2.429e - 03_{2.395e-04}$ [2]		DE	$3.709e - 03_{2.853e-04}$ [3]
	M2M	$3.157e - 03_{7.434e-04}$ [4]		M2M	$5.760e - 03_{2.057e-03}$ [4]
	AWA	$3.109e - 02_{8.845e-03}$ [5]		AWA	$1.219e - 02_{5.529e-03}$ [5]
LZ3	TDA-X	<b><math>2.128e - 03_{1.131e-04}</math></b> [1]	SLZ3	TDA-X	<b><math>9.521e - 04_{2.799e-05}</math></b> [1]
	TDA-F	$2.160e - 03_{1.674e-04}$ [2]		TDA-F	$1.420e - 03_{9.339e-04}$ [2]
	DE	$2.549e - 03_{1.241e-03}$ [4]		DE	$3.518e - 03_{1.934e-04}$ [3]
	M2M	$2.327e - 03_{1.821e-04}$ [3]		M2M	$3.774e - 03_{2.733e-04}$ [4]
	AWA	$7.092e - 03_{2.240e-03}$ [5]		AWA	$5.082e - 03_{2.638e-03}$ [5]
LZ4	TDA-X	<b><math>2.010e - 03_{9.338e-05}</math></b> [1]	SLZ4	TDA-X	<b><math>9.371e - 04_{3.229e-05}</math></b> [1]
	TDA-F	$2.192e - 03_{2.031e-04}$ [2]		TDA-F	$1.273e - 03_{3.375e-04}$ [2]
	DE	$3.016e - 03_{1.512e-03}$ [4]		DE	$3.544e - 03_{1.481e-04}$ [4]
	M2M	$2.966e - 03_{6.310e-04}$ [3]		M2M	$3.767e - 03_{3.099e-04}$ [5]
	AWA	$3.119e - 03_{2.191e-04}$ [5]		AWA	$1.457e - 03_{2.386e-04}$ [3]
LZ5	TDA-X	$7.867e - 03_{2.919e-03}$ [4]	SLZ5	TDA-X	<b><math>2.967e - 03_{8.921e-04}</math></b> [1]
	TDA-F	$6.872e - 03_{1.391e-03}$ [2]		TDA-F	$3.858e - 03_{1.192e-03}$ [2]
	DE	$7.091e - 03_{1.537e-03}$ [3]		DE	$4.185e - 03_{8.924e-04}$ [3]
	M2M	<b><math>4.240e - 03_{4.501e-04}</math></b> [1]		M2M	$4.728e - 03_{6.535e-04}$ [4]
	AWA	$1.163e - 02_{2.879e-03}$ [5]		AWA	$6.727e - 03_{2.693e-03}$ [5]
LZ6	TDA-X	$1.764e - 01_{4.460e-02}$ [4]	SLZ6	TDA-X	$1.787e - 01_{9.593e-02}$ [4]
	TDA-F	$2.587e - 01_{4.757e-02}$ [5]		TDA-F	$2.015e - 01_{8.980e-02}$ [5]
	DE	<b><math>3.015e - 02_{5.592e-03}</math></b> [1]		DE	$8.681e - 02_{1.326e-02}$ [3]
	M2M	$6.748e - 02_{2.449e-02}$ [3]		M2M	$7.559e - 02_{9.218e-03}$ [2]
	AWA	$5.286e - 02_{4.625e-03}$ [2]		AWA	$3.230e - 02_{9.002e-03}$ [1]
LZ7	TDA-X	$2.410e - 01_{9.336e-02}$ [5]	SLZ7	TDA-X	$5.807e - 02_{5.907e-02}$ [3]
	TDA-F	$2.342e - 01_{8.378e-02}$ [4]		TDA-F	$6.217e - 02_{5.097e-02}$ [4]
	DE	$8.053e - 02_{8.047e-02}$ [3]		DE	$3.842e - 02_{2.723e-02}$ [2]
	M2M	$5.082e - 02_{6.167e-02}$ [2]		M2M	$1.037e - 01_{6.527e-02}$ [5]
	AWA	<b><math>2.452e - 03_{1.740e-04}</math></b> [1]		AWA	<b><math>1.087e - 03_{2.781e-05}</math></b> [1]
LZ8	TDA-X	$1.774e - 02_{1.737e-02}$ [3]	SLZ8	TDA-X	$3.048e - 03_{1.409e-03}$ [2]
	TDA-F	$2.086e - 02_{2.080e-02}$ [4]		TDA-F	<b><math>2.696e - 03_{1.727e-03}</math></b> [1]
	DE	<b><math>3.653e - 03_{5.015e-03}</math></b> [1]		DE	$5.008e - 03_{7.817e-04}$ [3]
	M2M	$1.060e - 02_{4.170e-03}$ [2]		M2M	$5.383e - 03_{7.574e-04}$ [4]
	AWA	$6.847e - 02_{2.812e-02}$ [5]		AWA	$1.444e - 02_{1.589e-02}$ [5]
LZ9	TDA-X	$2.499e - 03_{5.715e-04}$ [2]	SLZ9	TDA-X	<b><math>1.080e - 03_{9.872e-05}</math></b> [1]
	TDA-F	$3.998e - 03_{1.559e-03}$ [3]		TDA-F	$1.484e - 03_{1.236e-04}$ [2]
	DE	<b><math>2.311e - 03_{1.561e-04}</math></b> [1]		DE	$6.135e - 03_{1.927e-03}$ [3]
	M2M	$4.874e - 03_{1.894e-03}$ [4]		M2M	$6.569e - 03_{1.232e-03}$ [4]
	AWA	$1.844e - 01_{1.558e-02}$ [5]		AWA	$3.887e - 02_{2.324e-02}$ [5]
Mean rank	TDA-X	2.8	Mean rank	TDA-X	1.7
	TDA-F	3.1		TDA-F	2.4
	DE	2.2		DE	3.1
	M2M	2.7		M2M	4.1
	AWA	4.2		AWA	4.7

**Table 2.** The mean and standard deviation of IGD values obtained by five algorithms after different percentages of function evaluations over 50 runs on the GLT suite.

		20%	60%	100%
GLT1	TDA-X	1.909e - 02 <sub>1.528e-02</sub> [4]	3.881e - 03 <sub>4.800e-03</sub> [4]	2.785e - 03 <sub>3.607e-03</sub> [4]
	TDA-F	<b>7.517e - 04</b> <sub><b>4.872e-05</b></sub> [1]	<b>6.961e - 04</b> <sub><b>3.897e-05</b></sub> [1]	<b>6.619e - 04</b> <sub><b>3.406e-05</b></sub> [1]
	DE	1.259e - 03 <sub>3.918e-04</sub> [3]	1.178e - 03 <sub>4.929e-07</sub> [3]	1.177e - 03 <sub>1.519e-07</sub> [3]
	M2M	1.182e - 03 <sub>8.326e-06</sub> [2]	1.160e - 03 <sub>5.941e-06</sub> [2]	1.146e - 03 <sub>6.267e-06</sub> [2]
	AWA	4.146e - 01 <sub>1.495e-01</sub> [5]	3.395e - 02 <sub>2.862e-02</sub> [5]	1.612e - 02 <sub>2.325e-02</sub> [5]
GLT2	TDA-X	1.730e - 01 <sub>1.445e-01</sub> [3]	5.778e - 02 <sub>4.546e-02</sub> [2]	4.960e - 02 <sub>3.610e-02</sub> [3]
	TDA-F	<b>5.411e - 02</b> <sub><b>6.426e-02</b></sub> [1]	<b>1.187e - 02</b> <sub><b>3.297e-03</b></sub> [1]	<b>1.024e - 02</b> <sub><b>1.009e-03</b></sub> [1]
	DE	1.506e - 01 <sub>1.592e-02</sub> [2]	1.524e - 01 <sub>5.333e-03</sub> [3]	1.527e - 01 <sub>3.876e-03</sub> [4]
	M2M	1.962e - 01 <sub>5.300e-02</sub> [4]	1.654e - 01 <sub>6.687e-04</sub> [4]	1.656e - 01 <sub>2.359e-04</sub> [5]
	AWA	2.040e + 00 <sub>1.076e+00</sub> [5]	2.386e - 01 <sub>2.329e-01</sub> [5]	1.569e - 02 <sub>1.060e-03</sub> [2]
GLT3	TDA-X	2.482e - 02 <sub>7.303e-03</sub> [4]	1.417e - 02 <sub>9.307e-03</sub> [4]	9.753e - 03 <sub>8.699e-03</sub> [4]
	TDA-F	1.943e - 02 <sub>9.349e-03</sub> [3]	6.295e - 03 <sub>6.039e-03</sub> [2]	<b>3.197e - 03</b> <sub><b>3.577e-03</b></sub> [1]
	DE	1.607e - 02 <sub>1.004e-02</sub> [2]	8.553e - 03 <sub>5.942e-03</sub> [3]	8.115e - 03 <sub>5.284e-03</sub> [3]
	M2M	<b>6.466e - 03</b> <sub><b>4.056e-04</b></sub> [1]	<b>5.982e - 03</b> <sub><b>2.015e-04</b></sub> [1]	5.881e - 03 <sub>9.950e-05</sub> [2]
	AWA	2.151e - 01 <sub>4.782e-02</sub> [5]	1.094e - 01 <sub>4.817e-02</sub> [5]	6.073e - 02 <sub>2.317e-02</sub> [5]
GLT4	TDA-X	3.734e - 02 <sub>8.125e-02</sub> [4]	2.913e - 02 <sub>8.167e-02</sub> [4]	2.225e - 02 <sub>6.917e-02</sub> [4]
	TDA-F	<b>2.487e - 03</b> <sub><b>2.644e-03</b></sub> [1]	<b>1.891e - 03</b> <sub><b>4.997e-05</b></sub> [1]	<b>1.874e - 03</b> <sub><b>3.510e-05</b></sub> [1]
	DE	1.218e - 02 <sub>4.398e-02</sub> [3]	5.185e - 03 <sub>1.110e-04</sub> [2]	5.167e - 03 <sub>1.129e-04</sub> [3]
	M2M	5.550e - 03 <sub>4.575e-04</sub> [2]	5.217e - 03 <sub>2.316e-04</sub> [3]	5.155e - 03 <sub>1.298e-05</sub> [2]
	AWA	4.843e - 01 <sub>1.649e-01</sub> [5]	6.700e - 02 <sub>5.839e-02</sub> [5]	2.883e - 02 <sub>5.044e-02</sub> [5]
GLT5	TDA-X	3.749e - 02 <sub>9.307e-03</sub> [2]	2.302e - 02 <sub>3.629e-03</sub> [3]	2.177e - 02 <sub>1.744e-03</sub> [3]
	TDA-F	4.794e - 02 <sub>9.241e-03</sub> [3]	2.279e - 02 <sub>4.658e-03</sub> [2]	<b>2.086e - 02</b> <sub><b>9.415e-04</b></sub> [1]
	DE	<b>2.589e - 02</b> <sub><b>1.601e-03</b></sub> [1]	<b>2.187e - 02</b> <sub><b>1.618e-03</b></sub> [1]	2.098e - 02 <sub>1.171e-03</sub> [2]
	M2M	5.220e - 02 <sub>6.313e-03</sub> [4]	5.402e - 02 <sub>6.722e-03</sub> [4]	5.471e - 02 <sub>6.731e-03</sub> [4]
	AWA	2.404e - 01 <sub>2.345e-02</sub> [5]	1.917e - 01 <sub>2.291e-03</sub> [5]	1.860e - 01 <sub>1.156e-03</sub> [5]
GLT6	TDA-X	1.632e - 01 <sub>8.498e-03</sub> [3]	1.617e - 01 <sub>7.846e-03</sub> [3]	1.616e - 01 <sub>7.832e-03</sub> [3]
	TDA-F	1.631e - 01 <sub>1.011e-03</sub> [2]	1.618e - 01 <sub>4.125e-04</sub> [4]	1.616e - 01 <sub>3.808e-04</sub> [4]
	DE	1.636e - 01 <sub>4.821e-02</sub> [4]	1.496e - 01 <sub>5.359e-02</sub> [2]	1.436e - 01 <sub>5.559e-02</sub> [2]
	M2M	<b>3.800e - 02</b> <sub><b>5.870e-03</b></sub> [1]	<b>3.813e - 02</b> <sub><b>6.428e-03</b></sub> [1]	<b>4.077e - 02</b> <sub><b>6.389e-03</b></sub> [1]
	AWA	3.375e - 01 <sub>4.023e-02</sub> [5]	2.354e - 01 <sub>1.011e-02</sub> [5]	2.301e - 01 <sub>9.980e-03</sub> [5]
Mean rank	TDA-X	3.3	3.3	3.5
	TDA-F	1.8	1.8	1.5
	DE	2.5	2.3	2.8
	M2M	2.3	2.5	2.7
	AWA	5.0	5.0	4.5

## 5 Comparison Study

In this section, we study the performance of the proposed strategy with some state-of-the-art algorithms on some test suites. The following algorithms are compared: (a) *TDA*: MOEA/D-TDA with  $dx$  and with an external population maintained by NDS, (b) *DE*: MOEA/D-DE [20], which is a conceptual MOEA/D algorithm and is similar to MOEA/D-TDA with  $dw$ , (c) *AWA*: MOEA/D-AWA [13], which is a variation of MOEA/D by adapting weight vectors in evolution, and (d) *M2M*: MOEA/D-M2M [15], which decomposes an MOP into a set of MOPs and tackle these MOPs simultaneously.

The first five instances in the LZ test suite [17], their variants, in which  $f_1$  is scaled to  $10f_1$ , and the GLT test suite [21] are used in the comparison study.

The variants of LZ1-LZ5 are called SLZ1-SLZ5 respectively. The experimental settings are as follows. For MOEA/D-TDA and MOEA/D-DE the experimental settings are the same as it is in Sect. 3. And for MOEA/D-AWA and MOEA/D-M2M, the experimental settings are the same as it is in the original paper.

Table 1 presents the mean and variance of IGD values obtained over 50 runs on LZ and SLZ test suites. On the LZ test suite, DE works the best and TDA-X achieves the best performance on LZ2, LZ3, and LZ4. In the SLZ test suite, TDA-X performs the best on all problems. The rank values obtained by the algorithms also indicate similar results. Comparing to LZ, the SLZ problems have more complex PF. This might be the reason that maintaining a well distributed population in the decision space is helpful for approximating the PF in the objective space especially when problems are with complicated PFs. Table 2 presents the mean and variance of IGD values obtained by the algorithms with different percentages of function evaluations. TDA-F achieves the best performance on all problems except on GLT6. Besides, TDA-F always gains the best rank value in every stage. The results indicate that TDA-F has better performance in the problems complicated in objective space than other state-of-art evolutionary algorithms.

## 6 Conclusions

This paper proposed a new adaptive strategy, called *domain decomposition and adaptation (TDA)*, to tune the weight vectors online in MOEA/D so as to find good approximations to both PF and PS. The empirical studies indicated that the search behavior measurement in the decision space is helpful and necessary to maintain a good approximation to the PS. Since it is hard to approximate both PS and PF well with a single population, an external archive is added to maintain a good approximation to the PF. Therefore, the proposed algorithm, called MOEA/D-TDA, has two populations: an internal population, which is with the subproblems that are adjusted by TDA, to approximate PS, and an external population to approximate PF. Comparing to the basic algorithm MOEA/D-DE, MOEA/D-TDA does not introduce additional control parameters.

The experimental study has demonstrated: (a) MOEA/D with a single population is hard to approximate both PS and PF, and a good approximation to the PS is necessary to find a good approximation to the PF; (b) TDA with a search behavior measurement in the decision space is helpful to find a good approximation to the PS; and (c) an external archive is helpful to find a good approximation to PF. A further systematic comparison study on several test suites has indicated the advantages of MOEA/D-TDA over some state-of-the-art MOEAs.

The success of TDA depends on two key issues: one is the domain decomposition strategy, and the other is the search behavior measurement. For the former, we use a simplex to represent the domain, and for the latter, we give an initial study based on L2 norm in the three domains. It is no doubt that there might be better ways to do so, and this is the target for future work. Besides,

in the proposed approach, the fineness of the weight vector distribution has a fix pattern and the number of subdomains increases rapidly when the number of objective increasing. These are the issues to be improved in the future.

**Acknowledgement.** This work is supported by the National Natural Science Foundation of China under Grant No. 61731009, 61673180, and 61703382.

## References

1. Deb, K.: *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Hoboken (2001)
2. Cartos Coelle Coello, G.B.L., Van Veldhuizen, D.A.: *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, Heidelberg (2002). <https://doi.org/10.1007/978-0-387-36797-2>
3. Tan, K.C., Khor, E.F., Lee, T.H.: *Multiobjective Evolutionary Algorithms and Applications*. Springer, Heidelberg (2006). <https://doi.org/10.1007/1-84628-132-6>
4. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength Pareto evolutionary algorithm. *Evolutionary Methods for Design Optimisation and Control*, pp. 95–100 (2001)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
6. Rostami, S., Neri, F.: Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. *Integr. Comput.-Aided Eng.* **23**(4), 313–329 (2016)
7. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X. (ed.) *PPSN 2004. LNCS*, vol. 3242, pp. 832–842. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30217-9\\_84](https://doi.org/10.1007/978-3-540-30217-9_84)
8. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
9. Liu, H.-L., Gu, F., Cheung, Y.: T-MOEA/D: MOEA/D with objective transform in multi-objective problems. In: *2010 International Conference of Information Science and Management Engineering*, vol. 2, pp. 282–285. IEEE (2010)
10. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
11. Trivedi, A., Srinivasan, D., Sanyal, K., Ghosh, A.: A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Trans. Evol. Comput.* **PP**(99), 1–23 (2016)
12. Li, H., Landa-Silva, D.: An adaptive evolutionary multi-objective approach based on simulated annealing. *Evol. Comput.* **19**(4), 561–595 (2011)
13. Qi, Y., Ma, X., Liu, F., Jiao, L., Sun, J., Wu, J.: MOEA/D with adaptive weight adjustment. *Evol. Comput.* **22**(2), 231–264 (2014)
14. Jiang, S., Cai, Z., Zhang, J., Ong, Y.-S.: Multiobjective optimization by decomposition with Pareto-adaptive weight vectors. In: *2011 Seventh International Conference on Natural Computation (ICNC)*, vol. 3, pp. 1260–1264. IEEE (2011)
15. Liu, H.-L., Gu, F., Zhang, Q.: Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Trans. Evol. Comput.* **18**(3), 450–455 (2014)

16. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: handling constraints and extending to an adaptive approach. *IEEE Trans. Evol. Comput.* **18**(4), 602–622 (2014)
17. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Trans. Evol. Comput.* **13**(2), 284–302 (2009)
18. Zhou, A., Zhang, Q.: Are all the subproblems equally important? Resource allocation in decomposition based multiobjective evolutionary algorithms. *IEEE Trans. Evol. Comput.* **20**(1), 52–64 (2016)
19. Zhou, A., Zhang, Q., Jin, Y., Tsang, E., Okabe, T.: A model-based evolutionary algorithm for bi-objective optimization. In: *IEEE Congress on Evolutionary Computation (CEC)*, vol. 3, pp. 2568–2575 (2005)
20. Li, Y., Zhou, A., Zhang, G.: An MOEA/D with multiple differential evolution mutation operators. In: *IEEE Congress on Evolutionary Computation (CEC)*, pp. 397–404 (2014)
21. Zhang, H., Zhou, A., Song, S., Zhang, Q., Gao, X.-Z., Zhang, J.: A self-organizing multiobjective evolutionary algorithm. *IEEE Trans. Evol. Comput.* **20**(5), 792–806 (2016)





# Use of Reference Point Sets in a Decomposition-Based Multi-Objective Evolutionary Algorithm

Edgar Manoatl Lopez<sup>(✉)</sup> and Carlos A. Coello Coello

Departamento de Computación, CINVESTAV-IPN  
(Evolutionary Computation Group), 07300 México D.F., Mexico  
emanoatl@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

**Abstract.** In recent years, decomposition-based multi-objective evolutionary algorithms (MOEAs) have gained increasing popularity. However, these MOEAs depend on the consistency between the Pareto front shape and the distribution of the reference weight vectors. In this paper, we propose a decomposition-based MOEA, which uses the modified Euclidean distance ( $d^+$ ) as a scalar aggregation function. The proposed approach adopts a novel method for approximating the reference set, based on an hypercube-based method, in order to adapt the reference set for leading the evolutionary process. Our preliminary results indicate that our proposed approach is able to obtain solutions of a similar quality to those obtained by state-of-the-art MOEAs such as MOMBII, NSGA-III, RVEA and MOEA/DD in several MOPs, and is able to outperform them in problems with complicated Pareto fronts.

## 1 Introduction

Many real-world problems have several (often conflicting) objectives which need to be optimized at the same time. They are known as Multi-objective Optimization Problems (MOPs) and their solution gives rise to a set of solutions that represent the best possible trade-offs among the objectives. These solutions constitute the so-called *Pareto optimal set* and their image is called the *Pareto Optimal Front* (POF). Over the years, Multi-Objective Evolutionary Algorithms (MOEAs) have become an increasingly common approach for solving MOPs, mainly because of their conceptual simplicity, ease of use and efficiency.

Decomposition-based MOEAs transform a MOP into a group of sub-problems, in such a way that each sub-subproblem is defined by a reference weight point. Then, all these sub-problems are simultaneously solved using a single-objective optimizer [16]. Because of their effectiveness (e.g., with respect

---

The first author acknowledges support from CONACyT and CINVESTAV-IPN to pursue graduate studies in Computer Science. The second author gratefully acknowledges support from CONACyT project no. 221551.

to Pareto-based MOEAs<sup>1</sup>) and efficiency,<sup>2</sup> decomposition-based MOEAs have become quite popular in recent years both in traditional MOPs and in many-objective problems (i.e., MOPs having four or more objectives).

However, the main disadvantage of decomposition-based MOEAs is that the diversity of its selection mechanism is led explicitly by the reference weight vectors (normally the weight vectors are distributed in a unit simplex). This makes them unable to properly solve MOPs with complicated Pareto fronts (i.e., Pareto fronts with irregular shapes).

Decomposition-based MOEAs are appropriate for solving MOPs with regular Pareto front (i.e., those sharing the same shape of a unit simplex). There is experimental evidence that indicates that decomposition-based MOEAs are not able to generate good approximations to MOPs having disconnected, degenerated, badly-scaled or other irregular Pareto front shapes [2, 5].

Here, we propose a decomposition-based MOEA, which adopts the modified Euclidean distance ( $d^+$ ) as a scalar aggregation function. This approach is able to switch between a PBI scalar aggregation function and the  $d^+$  distance in order to lead the optimization process. In order to adopt the  $d^+$  distance, we also incorporate an adaptive method for building the reference set. This method is based on the creation of hypercubes, which uses an archive for preserving good candidate solutions. We show that the resulting decomposition-based MOEA has a competitive performance with respect to state-of-the-art MOEAs, and that is able to properly deal with MOPs having complicated Pareto fronts.

The remainder of this paper is organized as follows. Section 2 provides some basic concepts related to multi-objective optimization. Our decomposition-based MOEA is described in Sect. 3. In Sect. 4, we present our methodology and a short discussion of our preliminary results. Finally, our conclusions and some possible paths for future research are provided in Sect. 5.

## 2 Basic Concepts

Formally a MOP in terms of minimization is defined as:

$$\text{minimize } \mathbf{f}(\mathbf{x}) := [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})]^T \quad (1)$$

subject to:

$$g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \dots, p \quad (2)$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \dots, q \quad (3)$$

where  $\mathbf{x} = [x_1, x_2, \dots, x_n]$  is the vector of decision variables,  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $i = 1, \dots, m$  are the objective functions and  $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $i = 0, \dots, p$ ,  $j = 1, \dots, q$  are the constraint functions of the problem.

<sup>1</sup> It is well-known that Pareto-based MOEAs cannot properly solve many-objective problems [12].

<sup>2</sup> The running time of decomposition-based MOEAs is lower than that of indicator-based MOEAs [1, 9] and reference-based MOEAs [14].

We also need to provide more details about the  $\text{IGD}^+$  indicator, which uses the modified Euclidean distance that we adopt in our proposal. According to [11], the  $\text{IGD}^+$  indicator can be described as follows:

$$\text{IGD}^+(\mathcal{A}, \mathcal{Z}) = \frac{1}{|\mathcal{Z}|} \left( \sum_{j=1}^{|\mathcal{Z}|} d_j^+(\mathbf{z}, \mathbf{a})^p \right)^{1/p} \quad (4)$$

where  $\mathbf{a} \in \mathcal{A} \subset \mathbb{R}^m$ ,  $\mathbf{z} \in \mathcal{Z} \subset \mathbb{R}^m$ ,  $\mathcal{A}$  is the Pareto front set approximation and  $\mathcal{Z}$  is the reference set.  $d^+(\mathbf{a}, \mathbf{z})$  is defined as:

$$d^+(\mathbf{z}, \mathbf{a}) = \sqrt{(\max\{a_1 - z_1, 0\})^2, \dots, (\max\{a_m - z_m, 0\})^2}. \quad (5)$$

Therefore, we can see that the set  $\mathcal{A}$  represents a better approximation to the real  $\mathcal{PF}$  when we obtain a lower  $\text{IGD}^+$  value, if we consider the reference set as  $\mathcal{PF}_{True}$ .  $\text{IGD}^+$  was shown to be weakly Pareto complaint, and this indicator presents some advantages with respect to the original Inverted Generational Distance (for more details about  $\text{IGD}$  and  $\text{IGD}^+$ , see [4] and [11] respectively).

### 3 Our Proposed Approach

#### 3.1 General Framework

Our approach adopts the same structure of the original MOEA/D [16], but we include some improvements in order to solve MOPs with complicated Pareto fronts. Our approach has the following features: (1) An archiving process for preserving candidate solutions which will form the reference set; (2) a method for adapting the reference set in order to sample uniformly the Pareto front; and (3) a rule for updating the reference set. Algorithm 1 shows the details of our proposed approach. Our proposed MOEA decomposes the MOP into scalar optimization subproblems, where each subproblem is solved simultaneously by an evolutionary algorithm (same as the original MOEA/D). The population, at each generation, is composed by the best solution found so far for each subproblem. Each subproblem is solved by using information only from its neighborhood, where each neighborhood is defined by the  $n$  candidate solutions which have the nearest distance based on the scalar aggregation function. The reference update process is launched when certain percentage of the evolutionary process (defined by “UpdatePercent”) is reached. The reference update process starts to store the non-dominated solutions in order to sample the shape of the Pareto front. When the cardinality of the set  $|\mathcal{A}|$  is equal to “ArchiveSize”, the reference method is launched for selecting the best candidate solutions, which will form the *new reference set*. Once this is done, the scalar aggregation function is updated by choosing the modified Euclidean distance ( $d^+$ ) (see Eq. (4)), and the set  $\mathcal{A}$  is cleaned up. The number of allowable updates is controlled by the variable “maxUpdates”.

---

**Algorithm 1.** General Framework

---

**Input:** A MOP, a stopping criterion,  $N$  subproblems, a uniform spread of  $N$  reference vectors:  $\lambda^1 \dots \lambda^N$ , number of solutions in the neighborhood and a scalar aggregation function ( $g$ ).

**Output:** Approximation of the MOP

- 1: Create each neighborhood for every reference vector:  $B(i)$ ;
- 2: Generate an initial population randomly  $(x_i, \dots, x_N) \in X$  ;
- 3:  $t \leftarrow 0$ ;
- 4:  $\mathcal{A} \leftarrow \{\}$ ;
- 5: **while**  $t < gen_{max}$  **do**
- 6:   **for** each  $B(i) \in \mathcal{B}$  **do**
- 7:     Apply evolutionary operators: Randomly select two parents from  $B(i)$  and create an individual  $y$ ;
- 8:     Improvement: Apply a problem-specific repair/improvement heuristic on  $y$  to produce  $y'$ ;
- 9:     **for** each  $j \in B(i)$  **do**
- 10:       **if**  $g(F(y'), \lambda^j) < g(F(x_j), \lambda^j)$  **then**
- 11:          $x_j \leftarrow y'$ ;
- 12:       **end if**
- 13:     **end for**
- 14:     Update of Neighboring Solutions: For each index in  $B(i)$  ;
- 15:     **if**  $t > UpdatePercent$  **then**
- 16:       **if**  $|\mathcal{A}| < ArchiveSize$  **then**
- 17:          $\mathcal{A} \leftarrow nonDominated(\mathcal{A} \cup y')$ ;
- 18:          $y_{ref} \leftarrow getNadirPoint(\mathcal{A})$ ;
- 19:       **end if**
- 20:       **if**  $|\mathcal{A}| == ArchiveSize$  and  $Updates < maxUpdates$  **then**
- 21:          $\lambda^1 \dots \lambda^N \leftarrow ComputeReferenceSet(\mathcal{A}, y_{ref}, z_{size})$ ;
- 22:          $g(\cdot) \leftarrow d^+$ ;
- 23:          $\mathcal{A} \leftarrow \{\}$ ;
- 24:          $Updates \leftarrow Updates + 1$ ;
- 25:       **end if**
- 26:     **end if**
- 27:   **end for**
- 28:    $t \leftarrow t + 1$ ;
- 29: **end while**
- 30:  $\mathcal{Q} \leftarrow non-Dominated(F(X))$ ;
- 31: **return**  $\mathcal{Q}, X$ ;

---

### 3.2 Archiving Process

As mentioned before, the archive stores non-dominated solutions, up to a maximum number of solutions defined by the “ArchiveSize” value. When the archive reaches its maximum capacity, the approximation reference algorithm is executed for selecting candidate solutions (these candidate solutions will form the so-called *candidate reference set*). After that, the archive is cleaned and the archiving process continues until reaching a maximum number of updates. The archiving process is applied after a 60% of the total number of generations. It is worth mentioning that the *candidate reference set* is not compatible with the

weight relation rule<sup>3</sup>, which implies that it is not possible to use the Tchebycheff scalar aggregation function for leading the search. However, the PBI function works because it only requires directions (for more details see [16]).

### 3.3 Reference Set

In our approach, we aim to select the best candidate points whose directions are promising (these candidate solutions will sample the Pareto front as uniformly as possible). The main idea is to apply a density estimator. For this reason, we propose to use an algorithm based on the hypercube contributions to select a certain number of reference points from the archive. Algorithm 2 provides the pseudo-code of an approach that is invoked with a set of non-dominated candidate points (called  $\mathcal{A}$  set) and the maximum number of reference points that we aim to find. The algorithm is organized in two main parts. In the first loop, we create a set of initial candidate solutions to form the so-called  $\mathcal{Q}$  set. Thus, the solutions from  $\mathcal{A}$  that form part of  $\mathcal{Q}$  will be removed from  $\mathcal{A}$ . After that, the greedy algorithm starts to find the best candidate solutions which will form the reference set  $\mathcal{Z}$ . In order to find the candidate reference points, the selection mechanism computes the hypercube contributions of the current reference set  $\mathcal{Q}$ . Once this is done, we remove the  $i^{th}$  solution that minimizes the hypercube value and we add a new candidate solution from  $\mathcal{A}$  to  $\mathcal{Q}$ . This process is executed until the cardinality of  $\mathcal{A}$  is equal to zero. In the line 21 of Algorithm 2, we apply the *expand* and *translate* operations. A hypercube is generated by the union of all the maximum volumes covered by a reference point. The  $i^{th}$  maximum volume is described as “the maximum volume generated by a set of candidate points” (these candidate points are obtained from the archive using a reference point  $y_{ref}$ ). The hypercube is computed using Algorithm 3. The main idea of this algorithm is to add all the maximum volumes, which are defined by the maximum point and the reference point ( $y_{ref}$ ). When a certain point is considered to be the maximum point, the objective space is split between  $m$  parts. The maximum point is removed from the set  $\mathcal{Q}$ . This process is repeated until  $\mathcal{Q}$  is empty.

In the first part of Algorithm 3, we validate if  $\mathcal{Q}$  contains one element. If that is the case, we compute the volume generated by  $y_{ref}$  and  $\mathbf{q} \in \mathcal{Q}$ . Otherwise, we compute the union of all the maximum hypercubes. In order to apply this procedure, we find the vector  $\mathbf{q}_{max}$  that maximizes the hypercube. Once this is done, we create  $m$  reference points which will form the so-called  $\mathcal{Y}$  set. For each reference point from  $\mathcal{Y}$ , we reduce the set  $\mathcal{Q}$  into a small subset in order to form the set  $\mathcal{Q}_{new}$ . Once this is done, we proceed to compute recursively the hypercube value of the new set formed by the subset  $\mathcal{Q}_{new}$  and the new reference point  $y_{new}$ . It is worth noting that this value allows to measure the relationship among each element of a non-dominated set.

---

<sup>3</sup> The weights of the reference point problem should be  $\sum_{i=0}^m \lambda_i = 1$ .

---

**Algorithm 2.** ComputeReferenceSet( $\mathcal{A}, z_{size}$ )

---

**Input:** A current non-dominated set  $\mathcal{A} \subset \mathbb{R}^m$  and maximum number of reference points  $z_{size}$ .

**Output:** Reference point set  $\mathcal{Z} \subset \mathbb{R}^m$  with  $|\mathcal{Z}| = z_{size}$

- 1:  $y_{ref} \leftarrow FindMaxValue(\mathcal{A}) + \epsilon$ ;
- 2:  $\mathcal{Q} \leftarrow \{\}$ ;
- 3: **while**  $|\mathcal{Q}| < (z_{size} + 1)$  **do**
- 4:    $\mathbf{a} \leftarrow pop(\mathcal{A})$ ;
- 5:    $\mathcal{Q} \cup \{\mathbf{a}\}$  ;
- 6: **end while**
- 7: **while**  $\mathcal{A}! = \{\}$  **do**
- 8:    $i \leftarrow 0$ ;
- 9:    $maxHypercube \leftarrow HCB(\mathcal{Q}, y_{ref})$ ;
- 10:   **for each**  $\mathbf{q} \in \mathcal{Q}$  **do**
- 11:      $ContHyperCube[i] \leftarrow maxHypercube - HCB(\mathcal{Q} \setminus \{\mathbf{q}\}, y_{ref})$ ;
- 12:      $i \leftarrow i + 1$ ;
- 13:   **end for**
- 14:    $i_{min} \leftarrow \operatorname{argmin} ContHyperCube$ ;
- 15:    $\mathcal{Q} \setminus \{q_{i_{min}}\}$ ;
- 16:    $\mathbf{a} \leftarrow pop(\mathcal{A})$ ;
- 17:    $\mathcal{Q} \cup \{\mathbf{a}\}$ ;
- 18: **end while**
- 19:  $\mathcal{Z} \leftarrow \{\}$ ;
- 20: **for each**  $\mathbf{q} \in \mathcal{Q}$  **do**
- 21:    $\mathcal{Z} \cup \{\mathbf{q} * \epsilon - \mathbf{l}\}$ ;
- 22: **end for**
- 23: **return**  $\mathcal{Z}$ ;

---

## 4 Experimental Results

We compare the performance of our approach with respect to that of four state-of-the-art MOEAs: MOEA/DD [13], NSGA-III [5], RVEA [2], and MOMBI-II [9]. These MOEAs had been found to be competitive in MOPs with a variety of Pareto front shapes. MOEA/DD [13] is an extension of MOEA/D which includes the Pareto dominance relation to select candidate solutions and is able to outperform the original MOEA/D, particularly in many-objective problems having up to 15 objectives. NSGA-III [5] uses a distributed set of reference points to manage the diversity of the candidate solutions, with the aim of improving convergence. The *Reference Vector Guided Evolutionary Algorithm* (RVEA) [2] provides very competitive results in MOPs with complicated Pareto fronts. *Many Objective Meta-heuristic Based on the R2 indicator* (MOMBI) [8] adopts the use of weight vectors and the *R2* indicator, and both mechanisms lead the optimization process. MOMBI is very competitive but it tends to lose diversity in high dimensionality. This study includes an improved version of this approach, called MOMBI-II [9].

**Algorithm 3.**  $HCB(Q, y_{ref})$ 


---

**Input:** A current set  $Q \subset \mathbb{R}^m$  and a reference point  $y_{ref}$   
**Output:** Hypercube value

- 1: **if**  $|Q| = 1$  **then**
- 2:   return  $\text{vol}(Q, y_{ref})$ ;
- 3: **end if**
- 4:  $VolList \leftarrow \{\}$ ;
- 5: **for** each  $p \in Q'$  **do**
- 6:    $VolList \cup \{\text{vol}(p, y_{ref})\}$ ;
- 7: **end for**
- 8:  $i_{max} \leftarrow \text{argmax } VolList$ ;
- 9:  $q_{max} \leftarrow Q[i_{max}]$ ;
- 10:  $\mathcal{Y} \leftarrow \text{SplitReferencePoint}(q_{max}, y_{ref})$ ;
- 11:  $Q \leftarrow Q \setminus \{q_{max}\}$ ;
- 12:  $hypercube \leftarrow 0$ ;
- 13: **for** each  $y_{new} \in \mathcal{Y}$  **do**
- 14:    $Q_{new} \leftarrow \text{CoverPoints}(Q, y_{new})$ ;
- 15:    $hypercube \leftarrow hypercube + HCB(Q_{new}, y_{new})$ ;
- 16: **end for**
- 17: return  $hypercube + \max(VolList)$ ;

---

## 4.1 Methodology

For our comparative study, we decided to adopt the Hypervolume indicator, due to this indicator is able to assess both convergence and maximum spread along the Pareto front. The reference points used in our preliminary study are shown in Table 1.

**Table 1.** Reference points used for the hypervolume indicator

Problem	Reference point	Problem	Reference point
DTLZ1	(1, 1, 1)	VNT1	(5, 6, 5)
DTLZ2-6	(2, 2, 2)	VNT2	(5, -15, -11)
DTLZ7	(2, 2, 7)	VNT3	(9, 18, 5)
MAF1-3	(2, 2, 2)	WFG1	(3, 5, 7)
MAF4	(3, 5, 9)	WFG2	(2, 4, 7)
MAF5	(9, 5, 3)	WFG3	(2, 3, 7)

We aimed to study the performance of our proposed approach when solving MOPs with complicated Pareto front shapes. For this reason, we selected 18 test problems with a variety of representative Pareto front shapes from some well-known and recently proposed test suites: the DTLZ [7], the WFG [10], the MAF [3] and the VNT test suites [15].

## 4.2 Parameterization

In the MAF and DTLZ test suites, the total number of decision variables is given by  $n = m + k - 1$ , where  $m$  is the number of objectives and  $k$  was set to 5 for DTLZ1 and MAF1, and to 10 for DTLZ2-6, and MAF2-5. The number of decision variables in the WFG test suite was set to 24, and the position-related parameter was set to  $m - 1$ . The distribution indexes for the Simulated Binary crossover and the polynomial-based mutation operators [6] adopted by all algorithms, were set to:  $\eta_c = 20$  and  $\eta_m = 20$ , respectively. The crossover probability was set to  $p_c = 0.9$  and the mutation probability was set to  $p_m = 1/L$ , where  $L$  is the number of decision variables. The total number of function evaluations was set in such a way that it did not exceed 60,000. In MOEA/DD, MOMBI-II and NSGA-III, the number of weight vectors was set to the same value as the population size. The population size  $N$  is dependent on  $H$ . For this reason, for all test problems, the population size was set to 120 for each MOEA. In RVEA, the rate of change of the penalty function and the frequency to conduct the reference vector adaptation were set to 2 and 0.1, respectively. Our approach was tested using a PBI scalar aggregation function and the modified Euclidean distance ( $d^+$ ). The maximum number of elements allowed in the archive was set to 500 and the maximum number of reference updates was set to 5.

## 4.3 Discussion of Results

Table 2 shows the average hypervolume values of 30 independent executions of each MOEA for each instance of the DTLZ, VNT, MAF and WFG test suites, where the best results are shown in **boldface** and grey-colored cells contain the second best results. The values in parentheses show the variance for each problem. We adopted the Wilcoxon rank sum test in order to compare the results obtained by our proposed MOEA and its competitors at a significance level of 0.05, where the symbol “+” indicates that the compared algorithm is significantly outperformed by our approach. On the other hand, the symbol “-” means that MOEA/DR is significantly outperformed by its competitor. Finally, “ $\approx$ ” indicates that there is no statistically significant difference between the results obtained by our approach and its competitor.

As can be seen in Table 2, our MOEA was able to outperform MOMBI-II, RVEA, MOEA/DD, and NSGA-III in seven instances and in several other cases, it obtained very similar results to those of the best performer. We can see that our approach outperformed its competitors in MOPs with degenerate Pareto fronts (DTLZ5-6 and VNT2-3). In this study, MOMBI-II is ranked as the second best overall performer, because it was able to outperform its competitors in four cases. It is worth mentioning that all the adopted MOEAs are very competitive because the final set of solutions obtained by them has similar quality in terms of the hypervolume indicator.

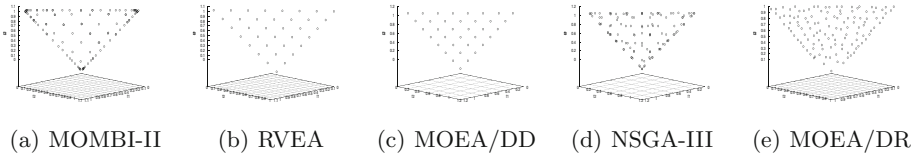
Figures 1, 2, 3 and 4 show a graphical representation of the final set of solutions obtained by each MOEA. On the MOPs with inverted Simplex-like Pareto fronts, our algorithm had a good performance (see Fig. 1). Figures 1a to e show



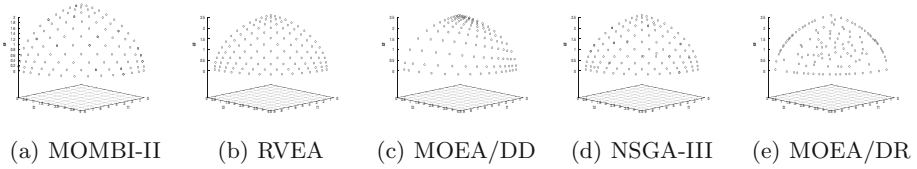
**Table 2.** Performance comparison among several MOEAs using the average hypervolume values obtained from 30 independent executions solving 18 benchmark problems for 3 objectives.

	MOMBI-II	RVEA	MOEA/DD	NSGA-III	MOEA/DR
DTLZ1	0.96622 (0.000001) +	0.66911 (0.000152) +	<b>0.97379 (0.000000)</b> ≈	0.96256 (0.001064) +	0.97265 (0.000007)
DTLZ2	7.36755 (0.000028) +	7.42224 (0.000000) ≈	7.42234 (0.000000) ≈	7.41893 (0.000000) +	<b>7.42684 (0.000143)</b>
DTLZ3	7.38843 (0.000084) -	7.40582 (0.000084) -	<b>7.4118 (0.000047)</b> -	7.38048 (0.000258) -	7.26131 (0.000248)
DTLZ4	7.3593 (0.036144) -	<b>7.42226 (0.000000)</b> -	7.42224 (0.000000) -	7.10506 (0.227356) ≈	7.10433 (1.093691)
DTLZ5	6.00978 (0.000000) +	5.9632 (0.000369) +	6.02456 (0.000062) +	5.84002 (0.05518) +	<b>6.10349 (0.000002)</b>
DTLZ6	5.79608 (0.00523) +	5.13815 (0.016264) +	5.6037 (0.006442) +	5.49135 (0.023354) +	<b>5.84857 (0.003765)</b>
DTLZ7	<b>13.37473 (0.000091)</b> -	13.0605 (1.283746) -	12.99409 (0.015542) ≈	13.32733 (0.002554) -	12.37989 (0.181549)
VNT1	61.44939 (0.000533) +	60.51323 (0.011862) +	60.55111 (0.021176) +	61.19214 (0.011932) +	<b>61.88114 (0.512056)</b>
VNT2	7.79702 (0.000001) +	7.7712 (0.000368) +	<b>7.80468 (0.000037)</b> +	7.77446 (0.000935) +	<b>7.84291 (0.000554)</b>
VNT3	15.11767 (0.000262) +	15.03082 (0.000422) +	15.06016 (0.000114) +	15.12629 (0.000502) +	<b>15.15149 (6.685422)</b>
MAF1	<b>5.44926 (0.000019)</b> -	5.37408 (0.000659) +	5.37139 (0.00009) +	5.4129 (0.000875) -	5.3986 (0.013358)
MAF2	5.08952 (0.000056) +	<b>5.1583 (0.000058)</b> ≈	5.11373 (0.000003) +	5.09758 (0.000043) +	5.14115 (0.000105)
MAF3	7.90637 (0.000043) -	<b>7.91154 (0.004847)</b> -	7.64261 (1.915744) +	7.89441 (0.00452) -	7.82731 (0.000558)
MAF4	<b>84.87316 (0.151259)</b> -	83.53436 (29.511151) -	51.80943 (1120.296924) +	83.73257 (1.377427) -	75.81219 (4.084039)
MAF5	95.97704 (52.294491) +	96.66782 (53.122845) +	<b>96.95207 (0.017991)</b> +	88.72762 (237.475764) +	<b>98.26977 (44.804422)</b>
WFG1	50.38691 (7.353216) -	<b>51.68413 (5.001739)</b> -	41.77398 (7.334821) +	44.95726 (10.36034) ≈	43.02462 (6.595565)
WFG2	48.72516 (12.06217) -	<b>51.14414 (0.045119)</b> -	44.23925 (3.146579) +	48.14747 (12.622738) -	46.87356 (1.171321)
WFG3	<b>24.28138 (0.007298)</b> -	22.12339 (0.086504) -	21.04349 (0.178677) -	23.54542 (0.037132) -	16.85662 (0.76122)

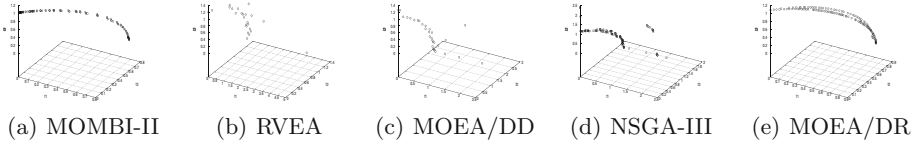
that the solutions produced by all the MOEAs adopted have a good coverage of the corresponding Pareto fronts. However, the solutions of MOMBI-II and NSGA-III are not distributed very uniformly, while the solutions of RVEA and MOEA/DD are distributed uniformly but their number is apparently less than their population size. On MOPs with badly-scaled Pareto fronts, our approach was able to obtain the best approximation (see Fig. 2). Figures 2a to e show that the solutions produced by all the MOEAs adopted are distributed very uniformly. On MOPs with degenerate Pareto fronts, it is clear that the winner in this category is our algorithm since the solutions of NSGA-III, RVEA and MOEA/DD are not distributed very uniformly, and they were not able to converge (see Fig. 3). On MOPs with disconnected Pareto fronts, our approach did not perform better than the other MOEAs. The reason is probably that the evolutionary operators were not able to generate solutions in the whole objective space, which makes the approximations produced by our approach to converge to a single region. Figure 4 shows that RVEA was able to obtain the best approximation in DTLZ7 since its approximation is distributed uniformly along the Pareto front.



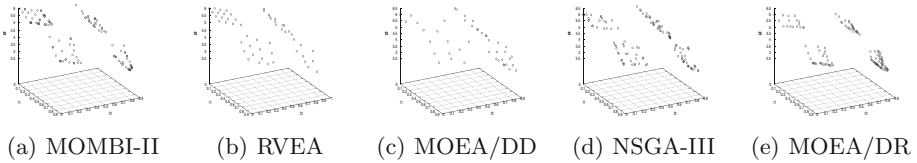
**Fig. 1.** Graphical representation of the final set of solutions obtained by each MOEA on MAF1 with 3 objectives



**Fig. 2.** Graphical representation of the final set of solutions obtained by each MOEA on MAF5 with 3 objectives



**Fig. 3.** Graphical representation of the final set of solutions obtained by each MOEA on DTLZ6 with 3 objectives



**Fig. 4.** Graphical representation of the final set of solutions obtained by each MOEA on DTLZ7 with 3 objectives

## 5 Conclusions and Future Work

We have proposed a decomposition-based MOEA for solving MOPs with different Pareto front shapes (i.e. those having complicated Pareto front shapes). The core idea of our proposed approach is to adopt the modified Euclidean distance ( $d^+$ ) as a scalar aggregation function. Additionally, our proposal introduces a novel method for approximating the reference set, based on an hypercube-based method, in order to adapt the reference set to address the evolutionary process. Our results show that our method for adapting the reference point set improves the performance of the original MOEA/D. As can be observed, the reference set is of utmost importance since our approach leads its search process using a set of reference points. Our preliminary results indicate that our approach is very competitive with respect to MOMBI-II, RVEA, MOEA/DD and NSGA-III, being able to outperform them in seven benchmark problems. Based on such results, we claim that our proposed approach is a competitive alternative to deal with MOPs having complicated Pareto front shapes. As part of our future work, we are interested in studying the sensitivity of our proposed approach to its parameters. We also intend to improve its performance in those cases in which it was not the best performer.

## References

1. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
2. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **20**(5), 773–791 (2016)
3. Cheng, R., et al.: A benchmark test suite for evolutionary many-objective optimization. *Complex Intell. Syst.* **3**(1), 67–81 (2017)
4. Coello Coello, C.A., Reyes Sierra, M.: A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) *MICAI 2004. LNCS (LNAI)*, vol. 2972, pp. 688–697. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24694-7\\_71](https://doi.org/10.1007/978-3-540-24694-7_71)
5. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Trans. Evol. Comput.* **18**(4), 577–601 (2014)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
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). [https://doi.org/10.1007/1-84628-137-7\\_6](https://doi.org/10.1007/1-84628-137-7_6)
8. Hernández Gómez, R., Coello Coello, C.A.: MOMBI: a new metaheuristic for many-objective optimization based on the R2 indicator. In: 2013 IEEE Congress on Evolutionary Computation (CEC 2013), Cancún, México, 20–23 June 2013, pp. 2488–2495. IEEE Press (2013). ISBN 978-1-4799-0454-9
9. Hernández Gómez, R., Coello Coello, C.A.: Improved metaheuristic based on the R2 indicator for many-objective optimization. In: 2015 Genetic and Evolutionary Computation Conference (GECCO 2015), Madrid, Spain, 11–15 July 2015, pp. 679–686. ACM Press (2015). ISBN 978-1-4503-3472-3
10. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **10**(5), 477–506 (2006)
11. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified distance calculation in generational distance and inverted generational distance. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) *EMO 2015. LNCS*, vol. 9019, pp. 110–125. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-15892-1\\_8](https://doi.org/10.1007/978-3-319-15892-1_8)
12. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: a short review. In: 2008 Congress on Evolutionary Computation (CEC 2008), Hong Kong, June 2008, pp. 2424–2431. IEEE Service Center (2008)
13. Li, K., Deb, K., Zhang, Q., Kwong, S.: An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Trans. Evol. Comput.* **19**(5), 694–716 (2015)
14. Manoatl Lopez, E., Coello Coello, C.A.: IGD<sup>+</sup>-EMOA: a multi-objective evolutionary algorithm based on IGD<sup>+</sup>. In: 2016 IEEE Congress on Evolutionary Computation (CEC 2016), Vancouver, Canada, 24–29 July 2016, pp. 999–1006. IEEE Press (2016). ISBN 978-1-5090-0623-9

15. Veldhuizen, D.A.V.: Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Ph.D. thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA, May 1999
16. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)



# Use of Two Reference Points in Hypervolume-Based Evolutionary Multiobjective Optimization Algorithms

Hisao Ishibuchi<sup>1</sup>(✉), Ryo Imada<sup>2</sup>, Naoki Masuyama<sup>2</sup>,  
and Yusuke Nojima<sup>2</sup>

<sup>1</sup> Shenzhen Key Laboratory of Computational Intelligence,  
Department of Computer Science and Engineering,  
Southern University of Science and Technology, Shenzhen 518055, China  
hisao@sustc.edu.cn

<sup>2</sup> 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  
ryo.imada@ci.cs.osakafu-u.ac.jp,  
{masuyama,nojima}@cs.osakafu-u.ac.jp

**Abstract.** Recently it was reported that the location of a reference point has a dominant effect on the optimal distribution of solutions for hypervolume maximization when multiobjective problems have inverted triangular Pareto fronts. This implies that the use of an appropriate reference point is indispensable when hypervolume-based EMO (evolutionary multiobjective optimization) algorithms are applied to such a problem. However, its appropriate reference point specification is difficult since it depends on various factors such as the shape of the Pareto front (e.g., triangular, inverted triangular), its curvature property (e.g., linear, convex, concave), the population size, and the number of objectives. To avoid this difficulty, we propose an idea of using two reference points: one is the nadir point, and the other is a point far away from the Pareto front. In this paper, first we demonstrate that the effect of the reference point is strongly problem-dependent. Next we propose an idea of using two reference points and its simple implementation. Then we examine the effectiveness of the proposed idea by comparing two hypervolume-based EMO algorithms: one with a single reference point and the other with two reference points.

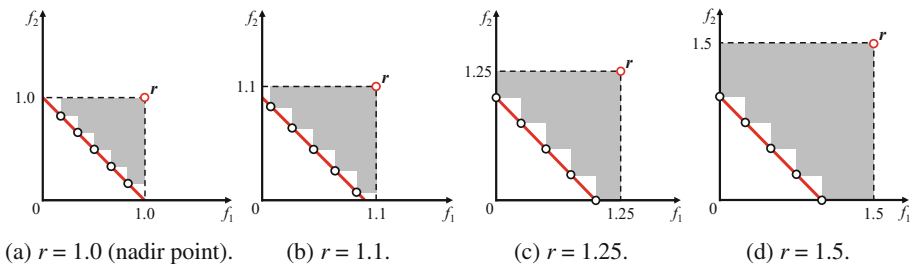
**Keywords:** Evolutionary multiobjective optimization (EMO)  
Hypervolume-based algorithms · Reference point specification  
Hypervolume contribution

## 1 Introduction

The hypervolume indicator [25] has been used for performance comparison in the EMO (evolutionary multiobjective optimization) community [26] due to its Pareto compliant property [24]. The hypervolume indicator has also been used in indicator-based EMO algorithms such as SMS-EMOA [3, 8], HypE [2], and FV-MOEA [18]. In

this paper, these algorithms are referred to as the hypervolume-based EMO algorithms. Their high performance on many-objective problems has been reported in the literature [10, 21, 22] in comparison with Pareto dominance-based EMO algorithms (e.g., NSGA-II [6]). Whereas the Pareto dominance-based selection pressure towards the Pareto front is severely weakened by the increase in the number of objectives, the hypervolume indicator can drive the population towards the Pareto front (usually at the cost of large computation load for many-objective problems [10]).

Properties of the hypervolume indicator can be visually examined by using the optimal distribution of solutions for hypervolume maximization. The optimal distribution has been theoretically derived for two-objective problems [1, 4] and empirically shown for multiobjective problems with three or more objectives [12–14]. Let us consider a two-objective minimization problem whose Pareto front is a straight line between (0, 1) and (1, 0) in a two-dimensional objective space. In Fig. 1, the Pareto front is shown by the red line. The optimal distribution of  $\mu$  solutions for hypervolume maximization is the equidistant distribution including (0, 1) and (1, 0) if the reference point  $\mathbf{r} = (r, r)$  for hypervolume calculation satisfies  $r \geq 1 + 1/(\mu - 1)$  [1, 4]. This condition is  $r \geq 1.25$  in Fig. 1 with  $\mu = 5$ . Thus the optimal distribution includes the two extreme points (0, 1) and (1, 0) of the Pareto front when  $r \geq 1.25$  as shown in Fig. 1(c) and (d). When  $r < 1.25$ , these two points are not included in the optimal distribution as shown in Fig. 1(a) and (b). It should be noted that the location of the reference point has no effect on the optimal distribution of solutions in Fig. 1 when  $r \geq 1.25$ . This observation suggests the use of a reference point which is far away from the Pareto front. Actually, the use of an infinitely large (i.e., distant) reference point in SMS-EMOA was mentioned in [8]. The reference point in SMS-EMOA in [3] was specified by adding 1.0 to the estimated nadir point in each generation (i.e., 2.0 in Fig. 1 if the true nadir point is correctly estimated).

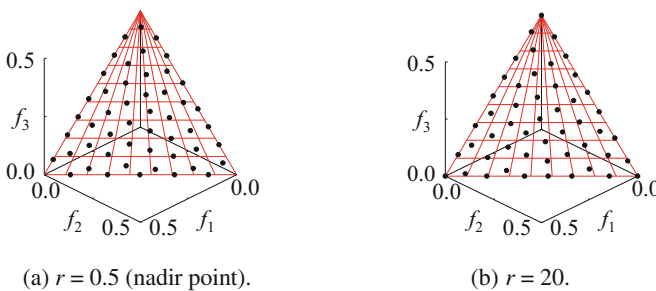


**Fig. 1.** The optimal distribution of five solutions ( $\mu = 5$ ) for each specification of the reference point  $\mathbf{r} = (r, r)$ . The shaded area shows the corresponding hypervolume. (Color figure online)

The above discussions imply that the reference point specification is not important in the hypervolume-based EMO algorithms. When the reference point is far away from the Pareto front of the two-objective minimization problem as in Fig. 1(d), the hypervolume-based EMO algorithms work well. In this case, the two extreme points (0, 1) and (1, 0) have much larger hypervolume contributions than the other three inside

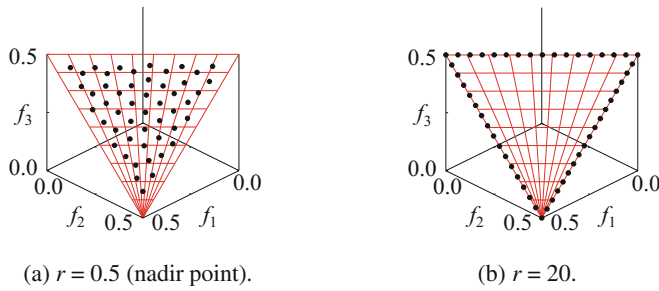
solutions. As a result, the two extreme points of the Pareto front are likely to be found. When the two extreme points are included in the current population, the location of the reference point has no effect on the hypervolume contributions of the other inside solutions. For example, the three inside solutions have the same hypervolume contributions in Fig. 1(c) with  $r = 1.25$  and Fig. 1(d) with  $r = 1.5$ .

A large reference point (which is far away from the Pareto front) can also be used for multiobjective minimization problems with triangular Pareto fronts such as the DTLZ1-4 [7] and WFG4-9 [9]. For example, in the case of three objectives, the hypervolume contributions of only the three extreme points of the Pareto front depend on the location of the reference point when they are included in the current population. Figure 2 shows approximately optimal distributions of 50 solutions of the three-objective DTLZ1 for two settings of the reference point  $\mathbf{r} = (r, r, r)$ :  $r = 0.5$  (i.e., nadir point) in Fig. 2(a) and  $r = 20$  in Fig. 2(b). These two distributions were obtained by SMS-EMOA with a large computation load (i.e., 1,000,000 generations) in our former study [12]. In Fig. 2(a) with  $r = 0.5$ , the three extreme points are not included in the obtained distribution since the nadir point is used as the reference point (i.e., since the hypervolume contributions of the three extreme points are zero when the nadir point is used as the reference point). In Fig. 2(b) with  $r = 20$ , the entire Pareto front is covered by the 50 solutions. Moreover, the two distributions in Fig. 2 are similar to each other whereas the totally different reference points are used. Figure 2 suggests that the use of a large reference point (which is far away from the Pareto front) works well on the three-objective DTLZ1. Figure 2 also suggests that the reference point specification is not important (since the similar results are obtained from the totally different reference points). Similar results are also obtained from the totally different reference points for the three-objective DTLZ2-4 and WFG4-9. It should be noted that the information of the true nadir point is used in those computational experiments (e.g., Fig. 2) to search for the optimal distribution of solutions.



**Fig. 2.** An approximately optimal distribution of 50 solutions ( $\mu = 50$ ) of the three-objective DTLZ1 test problem for each specification of the reference point  $\mathbf{r} = (r, r, r)$  [12].

Our discussions on Figs. 1 and 2 suggest the use of a large reference point in the hypervolume-based EMO algorithms. This is a good idea for two-objective minimization problems and multiobjective minimization problems with triangular Pareto fronts. However, this is not a good idea for multiobjective minimization problems with



**Fig. 3.** An approximately optimal distribution of 50 solutions ( $\mu = 50$ ) of the three-objective inverted DTLZ1 test problem for each specification of the reference point  $r = (r, r, r)$  [12].

inverted triangular Pareto fronts such as the inverted DTLZ1 [17], Minus-DTLZ1-4 [16] and Minus-WFG4-9 [16]. Figure 3 shows approximately optimal distributions of 50 solutions of the three-objective inverted DTLZ1 for the two settings of the reference point:  $r = 0.5$  (i.e., nadir point) in Fig. 3(a) and  $r = 20$  in Fig. 3(b). These two distributions were obtained by SMS-EMOA after 1,000,000 generations in our former study [12]. Figure 3(b) clearly shows that the use of a large reference point is not appropriate in the hypervolume-based EMO algorithms. The use of the nadir point is not appropriate as shown in Fig. 3(a), either.

An appropriate specification of the reference point was discussed from a viewpoint of fair performance comparison of EMO algorithms in our former studies [13, 14]. The basic idea is to specify the reference point so that uniformly distributed solutions over the entire Pareto front have similar hypervolume contributions (i.e., any solution should not have a dominantly large or negligibly small contribution). For the two-objective minimization problem with the linear Pareto front in Fig. 1, the suggested reference point in [13, 14] is  $r = 1 + 1/(\mu - 1)$  where  $\mu$  is the population size. In Fig. 1 with the population size 5,  $r$  is calculated as  $r = 1.25$ . This specification is used in Fig. 1(c) where each solution has exactly the same hypervolume contribution. By using an integer parameter  $H$  which denotes the number of intervals determined by  $\mu$  solutions (i.e.,  $H = \mu - 1$ ), the suggested specification is rewritten as  $r = 1 + 1/H$ . The integer parameter  $H$  in this formulation is the same as  $H$  in the weight vector specification mechanism in MOEA/D [23]. Using this fact, the reference point specification method by  $r = 1 + 1/H$  was extended to multiobjective minimization problems with linear Pareto fronts in [13, 14] where the value of  $H$  was determined from the number of objectives  $M$  and the population size  $\mu$  using the following formulation:

$${}_{H+M-1}C_{M-1} \leq \mu < {}_{H+M}C_{M-1}. \quad (1)$$

In this formulation,  ${}_nC_m$  denotes the number of combinations of selecting  $m$  elements from a set of  $n$  elements ( $n \geq m$ ):  ${}_nC_m = n!/m!(n-m)!$ .

The reference point specification method of  $r = 1 + 1/H$  with (1) is a good guideline for performance comparison of EMO algorithms. However, it does not always work well in the hypervolume-based EMO algorithms as we will show later in this paper. It is difficult to appropriately specify the reference point in the hypervolume-



based EMO algorithms especially for multiobjective problems with nonlinear inverted triangular Pareto fronts (e.g., Minus-DTLZ2-4 and Minus-WFG4-9 [16]). This is because the appropriate reference point specification depends on various factors such as the shape of the Pareto front and its curvature property in addition to the number of objectives ( $M$ ) and the population size ( $\mu$ ) used in (1). This is also because the true Pareto front is unknown (i.e., because the reference point specification should be based on the estimation nadir point, which is not always accurate).

To avoid the difficulty in appropriately specifying the reference point, we propose an idea of using two reference points. One is the estimated nadir point and the other is far away from it. Our idea is motivated by a simple intuition from Fig. 3: A good solution set would be obtained by combining the two solution sets in Fig. 3.

This paper is organized as follows. First, we demonstrate the difficulty in appropriately specifying the reference point in Sect. 2. Experimental results are explained using the hypervolume contributions of uniformly distributed solutions. Next, we propose an idea of using two reference points and its simple implementation in Sect. 3. Then, we examine the effectiveness of our idea in Sect. 4. Our two-point approach is compared with the standard single-point approach. Finally, we conclude this paper in Sect. 5 where a number of future research directions are suggested.

## 2 Empirical Discussions on Reference Point Specification

In this section, we show experimental results by FV-MOEA [18] on the three-objective DTLZ1 [7], DTLZ2 [7], Minus-DTLZ1 [16], Minus-DTLZ2 [16] and the car-side impact problem [17]. FV-MOEA is a recently-proposed fast hypervolume-based EMO algorithm. We use FV-MOEA in the same specifications as SMS-EMOA. Thus the same experimental results are obtained from FV-MOEA and SMS-EMOA. We use FV-MOEA because it is faster than SMS-EMOA (whereas we used SMS-EMOA in our former studies [12–14]).

FV-MOEA is applied to each three-objective minimization problem. During its execution, the objective space is normalized using non-dominated solutions in each generation as follows (e.g., see [11]). First, non-dominated solutions in the current population are selected. Next, the minimum and maximum values of each objective are found in the selected non-dominated solutions. Then, each objective is normalized so that the minimum and maximum values are 0 and 1, respectively. FV-MOEA with various specifications of the reference point is used under the following settings.

Population size ( $\mu$ ): 100,  
 Termination condition: 100,000 solution evaluations,  
 Crossover: SBX (Crossover probability: 1.0, Distribution index: 20),  
 Mutation: PM (Mutation probability:  $1/(\text{String length})$ , Distribution index: 20),  
 Number of runs: 11 runs.

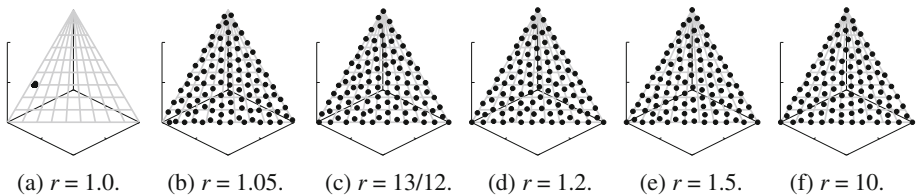
Among the 11 runs for each specification of the reference point, a single run with the median hypervolume is selected and shown as the experimental result in this paper.

Since the population size is 100 for the three-objective problems (i.e.,  $\mu = 100$  and  $M = 3$ ), the suggested reference point in [13, 14] is calculated from (1) as  $r = 1 + 1/$

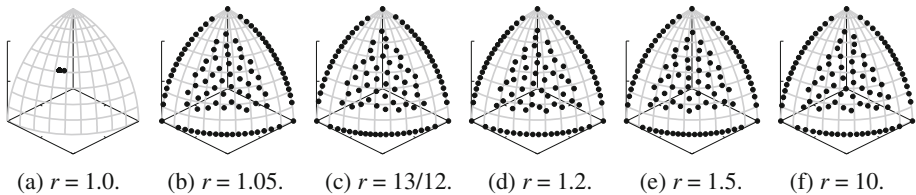
$H = 13/12$ . In addition to this specification, we also examine the following values:  $r = 1.0$  (the estimated nadir point), 1.05 (closer to the estimated nadir point than  $13/12$ ), 1.2 (slightly larger than  $13/12$ ), 1.5 (larger than  $13/12$ ) and 10 (far away from the Pareto front: much larger than the others). Experimental results are shown in Figs. 4, 5, 6, 7 and 8.

In Fig. 4 on DTLZ1 and Fig. 5 on DTLZ2, almost the same results are obtained when  $r \geq 1.05$ . These results suggest the use of a large reference point for multiobjective minimization problems with triangular Pareto fronts. These results also show that the reference point specification is not important for such a multiobjective problem as long as the reference point is not too close to the estimated nadir point.

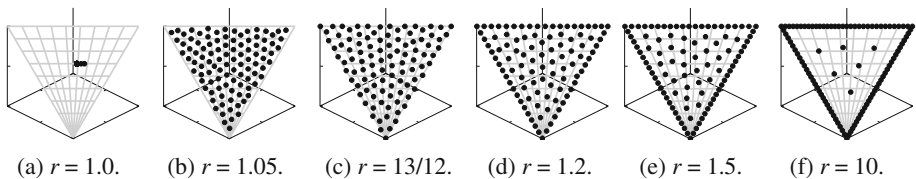
However, in Figs. 6, 7 and 8, totally different results are obtained from different specifications of the reference point. When the reference point is far away from the estimated nadir point (i.e.,  $r = 10$ ), many solutions are around the boundary of the Pareto front. In this case, only a small number of solutions are obtained inside the Pareto front. Thus we can see from Figs. 6, 7 and 8 that a large reference point is not appropriate.



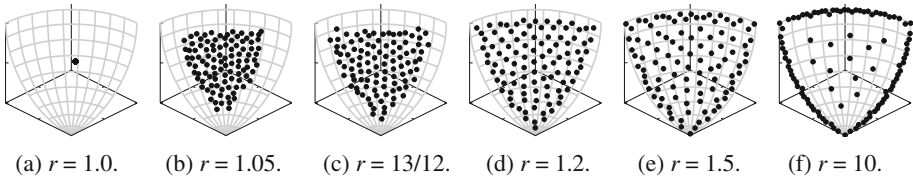
**Fig. 4.** Experimental results on DTLZ1 (median results over 11 runs).



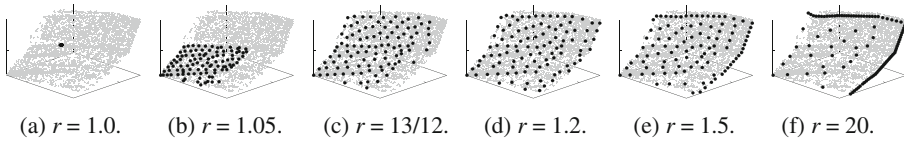
**Fig. 5.** Experimental results on DTLZ2 (median results over 11 runs).



**Fig. 6.** Experimental results on Minus-DTLZ1 (median results over 11 runs).



**Fig. 7.** Experimental results on Minus-DTLZ2 (median results over 11 runs).

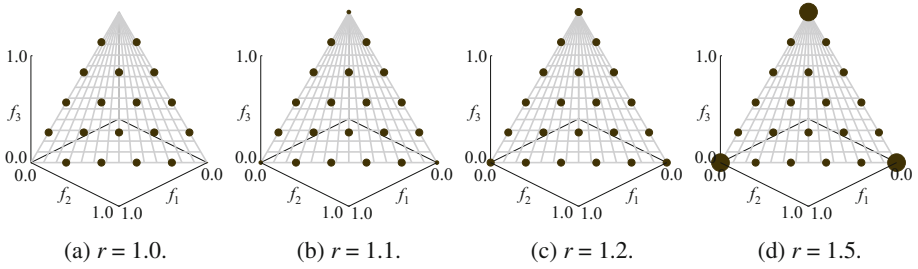


**Fig. 8.** Experimental results on the car-side impact problem (median results over 11 runs).

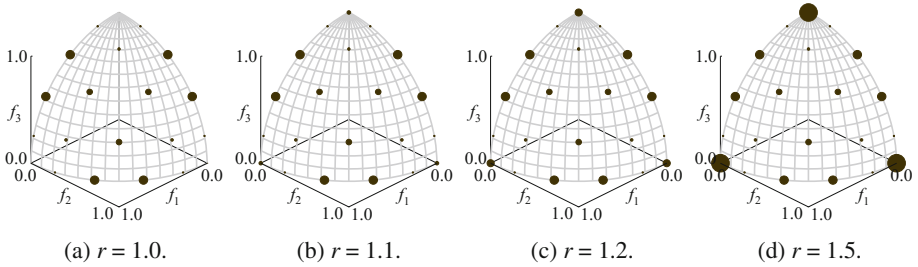
Independent of the shape of the Pareto front, the use of the estimated nadir point (i.e.,  $r = 1.0$  in Figs. 4, 5, 6, 7 and 8(a)) is not advisable since the diversity of the obtained solution sets is very small. It should be noted that the obtained solution sets in Figs. 4(a) and 5(a) are totally different from the approximately optimal solution sets in Figs. 1(a) and 2(a), respectively. This is because the true nadir point is used in Figs. 1 and 2 while the estimated nadir point is used in Figs. 4, 5, 6, 7 and 8.

As shown in Figs. 4 and 5, for multiobjective problems with triangular Pareto fronts, the reference point specification is not important since almost the same solution sets are obtained from different specifications of the reference point as far as it is not too close to the estimated nadir point. On the contrary, for multiobjective problems with inverted triangular Pareto fronts, the reference point specification is important (see Figs. 6 and 7). However, it is difficult to appropriately specify the reference point for such a problem. For example, whereas the suggested reference point by  $r = 1 + 1/H = 13/12$  works well on Minus-DTLZ1 in Fig. 6, it is too small for Minus-DTLZ2 in Fig. 7. In Fig. 7,  $r = 1.5$  seems to be appropriate. However, it seems to be too large in Fig. 6 (compare Fig. 6(e) with Fig. 6(c) and (d)).

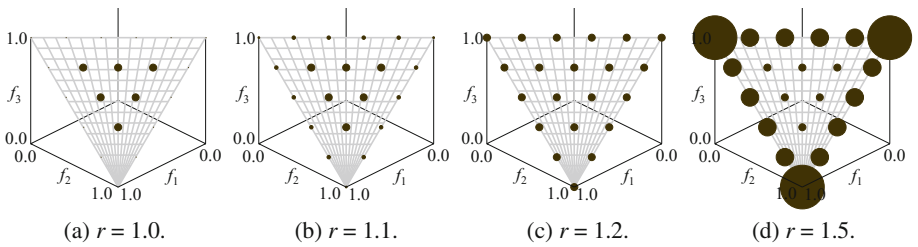
Our experimental results in Figs. 4, 5, 6, 7 and 8 can be explained using the hypervolume contributions of uniformly distributed solutions. In Figs. 9, 10, 11 and 12, we show the hypervolume contributions of 21 uniformly distributed solutions on the Pareto fronts. Each test problem in Figs. 9, 10, 11 and 12 is normalized so that the ideal and nadir points are  $(0, 0, 0)$  and  $(1, 1, 1)$ , respectively. The 21 solutions are generated in the same manner as the weight vector generation mechanism in MOEA/D with  $H = 5$ . The suggested reference point by  $r = 1 + 1/H$  is 1.2. In each figure, the size (i.e., area) of the closed circle is proportional to the hypervolume contribution of the corresponding solution. When the hypervolume contribution is zero, the corresponding solution is not shown.



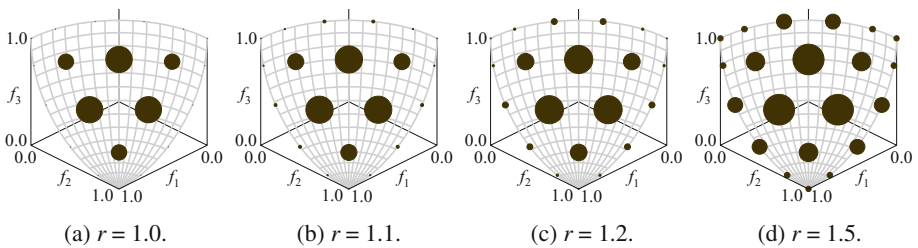
**Fig. 9.** Hypervolume contribution of each solution of DTLZ1.



**Fig. 10.** Hypervolume contribution of each solution of DTLZ2.



**Fig. 11.** Hypervolume contribution of each solution of Minus-DTLZ1.



**Fig. 12.** Hypervolume contribution of each solution of Minus-DTLZ2.

In Figs. 9, 10, 11 and 12(a) with  $r = 1.0$ , the hypervolume contributions of the three extreme points are zero. When the estimated nadir point is used as the reference point (i.e.,  $r = 1.0$ ) in the hypervolume-based EMO algorithms, the hypervolume contributions of the extreme points in the current population are zero. Thus they are likely to be removed from the current population through generation update. Then the diversity of the population gradually decreases, which increases the inaccuracy of the nadir point estimation. This is the reason for the very small diversity in Figs. 4, 5, 6, 7 and 8(a) with  $r = 1.0$ .

In Fig. 9 on DTLZ1 and Fig. 10 on DTLZ2, the hypervolume contributions of only the three extreme points depend on the reference point specification. This is the reason why almost the same results are obtained in Figs. 4 and 5 independent of the reference point specification except for the case where the reference point is too small. On the contrary, in Fig. 11 on Minus-DTLZ1 and Fig. 12 on Minus-DTLZ2, the reference point specification affects the hypervolume contributions of all boundary solutions. When the nadir point is used as the reference point in Figs. 11(a) and 12(a), the hypervolume contributions of all boundary solutions are zero. By increasing the distance between the reference point and the nadir point (i.e., by moving the reference point far away from the Pareto front), their hypervolume contributions increase. When the reference point is far away from the Pareto front, boundary solutions have large hypervolume contributions. This is the reason why only a small number of inside solutions are obtained in Figs. 6(f) and 7(f) with  $r = 10$ . The upper-right half of the Pareto front of the car-side impact problem in Fig. 8 has a similar property to the inverted triangular Pareto fronts of Minus-DTLZ1 and Minus-DTLZ2. Thus many solutions are obtained along the upper-right boundary of the Pareto front in Fig. 8(f).

When the suggested reference point (i.e.,  $r = 1.2$ ) is used for DTLZ1 in Fig. 9 and Minus-DTLZ1 in Fig. 11, all solutions in each figure have the same hypervolume contribution. This is the reason why the well-distributed solution sets are obtained for those test problems in Figs. 4(c) and 6(c). However, in Fig. 10 on DTLZ2 and Fig. 12 on Minus-DTLZ2, each solution has a different hypervolume contribution due to the nonlinearity of their Pareto fronts. As a result, well-distributed solution sets are not obtained in Figs. 5 and 7 independent of the reference point specification.

### 3 Proposed Idea and Its Simple Implementation

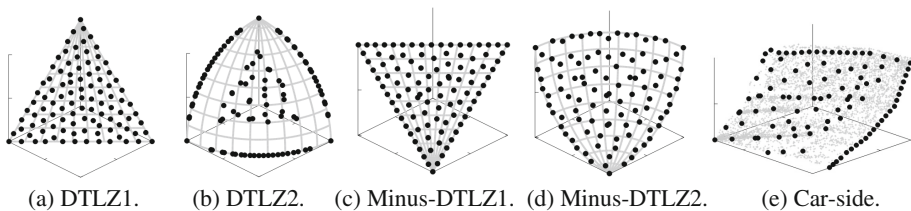
Our idea is to use two reference points in order to avoid the difficulty in appropriately specifying a single reference point for multiobjective problems with inverted triangular Pareto fronts. As the first attempt, we specify the two reference points as  $r = 1.0$  and  $r = 10$ , respectively. That is, one reference point is the estimated nadir point, and the other is far away from it. The population is divided into two subpopulations of the same size. A hypervolume-based EMO algorithm (FV-MOEA [18] in this paper) is applied to each subpopulation using a different reference point:  $r = 1.0$  for one subpopulation and  $r = 10$  for the other. The final result of the proposed idea is the merged solution set of the two subpopulations. The execution of FV-MOEA is performed in each subpopulation separately except for the following two procedures.

- (i) **Normalization:** The normalization of the objective space is performed in each generation using non-dominated solutions among all solutions in the two subpopulations. This is for accurately estimating the nadir point in each generation. If the normalization is performed separately, good results are not obtained from  $r = 1.0$  as we have already shown in Figs. 4, 5, 6, 7 and 8(a) in the previous section.
- (ii) **Periodical Subpopulation Comparison:** If the two subpopulations are similar, a good merged solution set cannot be obtained from them. In this case, it may be a good idea to merge them into a single population during the execution of FV-MOEA instead of merging them after its separate execution on each subpopulation. In this paper, we examine the similarity of the two subpopulations four times during its execution (after 20%, 40%, 60%, and 80% use of the available computation load, i.e., after 20,000th, 40,000th, 60,000th, and 80,000th solution evaluations). If the two subpopulations are similar, we merge them into a single population and FV-MOEA is applied to the merged population. The reference point is specified as  $r = 1 + 1/H$ . Once the two subpopulations are merged, the merged population is not divided again.

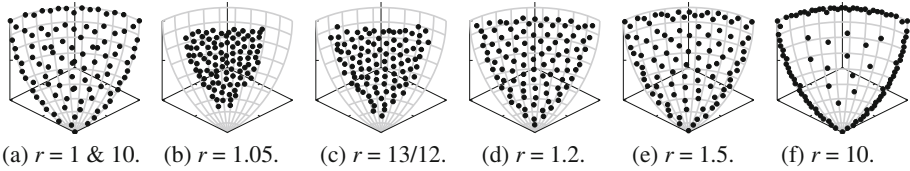
One important issue is how to measure the similarity of the two subpopulations. In this paper, we use the  $IGD^+$  indicator [15] where the subpopulation with  $r = 10$  is used as the  $IGD^+$  reference points to calculate  $IGD^+$  of the other subpopulation with  $r = 1.0$ . When the calculated  $IGD^+$  is smaller than  $2^{1/2}/5H$ , we merge the two subpopulations. The threshold value is specified as  $2^{1/2}/5H$  based on the following consideration. In the normalized three-objective DTLZ1, the length of each side of the triangular Pareto front is  $2^{1/2}$  (e.g., the distance of the line between (1, 0, 0) and (0, 1, 0)). When  $\mu =_{H+M-1} C_{M-1}$  solutions are uniformly distributed over the entire Pareto front, each side was divided into  $H$  intervals. Thus the distance between adjacent solutions on each side is  $2^{1/2}/H$ . The threshold value  $2^{1/2}/5H$  is 1/5 of the distance between adjacent solutions on each side in the uniformly distributed solutions. Of course, other indicators (e.g.,  $IGD$  [5, 20] and  $\Delta_p$  [19]) and/or other specifications of the threshold value can be used, which is an important future research topic.

## 4 Experimental Results by the Proposed Idea

Using the same parameter specifications as in Sect. 2, we apply FV-MOEA with two reference points ( $r = 1.0$  and  $r = 10$ ) to the five test problems. Median experimental results among 11 runs are shown in Fig. 13.



**Fig. 13.** Experimental results by FV-MOEA with two reference points.



**Fig. 14.** Comparison of the proposed idea with the standard single reference point approach.

In Fig. 13(c)–(d), we obtain the intended results. Many solutions around the boundary of the Pareto front of each test problem are obtained from  $r = 10$ . At the same time, many inside solutions are also obtained from  $r = 1.0$ . The effectiveness of the proposed idea is clearly shown in Fig. 13(d) for Minus-DTLZ2. In Fig. 14, we compare the obtained solution set by the proposed idea (i.e., Fig. 14(a) which is the same as Fig. 13(d)) with the results by the standard FV-MOEA with a single reference point (i.e., Fig. 14(b)–(f) which are the same as Fig. 7(b)–(f)). The solution set in Fig. 14(a) is similar to the solution set in Fig. 14(e) with  $r = 1.5$ . However, the boundary solutions in Fig. 14(a) are much closer to the boundary of the Pareto front than Fig. 14(e). That is, the solution set in Fig. 14(a) covers the wider region of the Pareto front than Fig. 14(e). Similar observations can be obtained from Fig. 13(c) and (e) by comparing them with the corresponding results of the standard FV-MOEA with a single reference point in Figs. 6 and 8, respectively.

The obtained solution set of DTLZ1 in Fig. 13(a) is almost the same as the solution sets in Fig. 4(c)–(f). This is because the two subpopulations are merged into a single population during the execution of FV-MOEA as intended. Once the two subpopulations are merged, FV-MOEA with the two reference points is exactly the same as FV-MOEA with  $r = 1 + 1/H$ . The obtained solution set of DTLZ2 in Fig. 13(b) seems to be inferior to the results in Fig. 5(b)–(f). This is because the two subpopulations are not merged in Fig. 13(b). By changing the threshold value from  $2^{1/2}/5H$  to  $2^{1/2}/2H$ , almost the same solution set as Fig. 5(b)–(f) is obtained from FV-MOEA with the two reference points. This is because the two subpopulations are merged and FV-MOEA with  $r = 1 + 1/H$  is used. This result suggests the necessity of further examinations about the parameter setting in the proposed idea.

## 5 Conclusions

In this paper, we proposed an idea of using two reference points in hypervolume-based EMO algorithms to avoid the difficulty in appropriately specifying a single reference point for multiobjective problems with inverted triangular Pareto fronts. Whereas promising results were obtained by a simple implementation of the proposed idea, a number of issues are left for future research to design a competent hypervolume-based EMO algorithm with two reference points. Among them are the choice of a similarity indicator and a threshold value, the timing of similarity check, and the specification of the two reference points (e.g., the use of an infinitely large reference point). Information exchange mechanisms between the two subpopulations should be further addressed.



Discussions are also needed on the estimation of the nadir point, the normalization of the objective space (e.g., see [11]), and the computational complexity of the proposed idea. Of course, performance comparison of the proposed idea with other EMO algorithms is needed. Another important future research topic is to examine the shape of the Pareto fronts of real-world multiobjective problems (e.g., triangular, inverted triangular or others; linear, concave or convex).

**Acknowledgments.** This work was supported by the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284).

## References

1. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Hypervolume-based multiobjective optimization: theoretical foundations and practical implications. *Theoret. Comput. Sci.* **425**, 75–103 (2012)
2. Bader, J., Zitzler, E.: HypE: an algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.* **19**, 45–76 (2011)
3. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**, 1653–1669 (2007)
4. Brockhoff, D.: Optimal  $\mu$ -distributions for the hypervolume indicator for problems with linear bi-objective fronts: exact and exhaustive results. In: Deb, K. (ed.) SEAL 2010. LNCS, vol. 6457, pp. 24–34. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17298-4\\_2](https://doi.org/10.1007/978-3-642-17298-4_2)
5. Coello Coello, C.A., Reyes Sierra, M.: A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 688–697. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24694-7\\_71](https://doi.org/10.1007/978-3-540-24694-7_71)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**, 182–197 (2002)
7. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of IEEE CEC 2002, pp. 825–830 (2002)
8. Emmerich, M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-31880-4\\_5](https://doi.org/10.1007/978-3-540-31880-4_5)
9. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **10**, 477–506 (2006)
10. Ishibuchi, H., Akedo, N., Nojima, Y.: Behavior of multi-objective evolutionary algorithms on many-objective knapsack problems. *IEEE Trans. Evol. Comput.* **19**, 264–283 (2015)
11. Ishibuchi, H., Doi, K., Nojima, Y.: On the effect of normalization in MOEA/D for multi-objective and many-objective optimization. *Complex Intell. Syst.* **3**, 279–294 (2017)
12. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: Hypervolume subset selection for triangular and inverted triangular Pareto fronts of three-objective problems. In: Proceedings of FOGA 2017, pp. 95–110 (2017)
13. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: Reference point specification in hypervolume calculation for fair comparison and efficient search. In: Proceedings of GECCO 2017, pp. 585–592 (2017)



14. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: How to Specify a Reference Point in Hypervolume Calculation for Fair Performance Comparison. *Evolutionary Computation* (in press)
15. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified distance calculation in generational distance and inverted generational distance. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) *EMO 2015. LNCS*, vol. 9019, pp. 110–125. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-15892-1\\_8](https://doi.org/10.1007/978-3-319-15892-1_8)
16. Ishibuchi, H., Setoguchi, Y., Masuda, H., Nojima, Y.: Performance of decomposition based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Trans. Evol. Comput.* **21**, 169–190 (2017)
17. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based non-dominated sorting approach, part II: handling constraints and extending to an adaptive approach. *IEEE Trans. Evol. Comput.* **18**, 602–622 (2014)
18. Jiang, S., Zhang, J., Ong, Y.-S., Zhang, A.N., Tan, P.S.: A simple and fast hypervolume indicator-based multiobjective evolutionary algorithm. *IEEE Trans. Cybern.* **45**, 2202–2213 (2015)
19. Schütze, O., Esquivel, X., Lara, A., Coello Coello, C.A.: Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Trans. Evol. Comput.* **16**, 504–522 (2012)
20. Sierra, M.R., Coello Coello, C.A.: A new multi-objective particle swarm optimizer with improved selection and diversity mechanisms. Technical report, CINVSTAV-IPN (2004)
21. Tanabe, R., Ishibuchi, H., Oyama, A.: Benchmarking multi- and many-objective evolutionary algorithms under two optimization scenarios. *IEEE Access* **5**, 19597–19619 (2017)
22. 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). [https://doi.org/10.1007/978-3-540-70928-2\\_56](https://doi.org/10.1007/978-3-540-70928-2_56)
23. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**, 712–731 (2007)
24. 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). [https://doi.org/10.1007/978-3-540-70928-2\\_64](https://doi.org/10.1007/978-3-540-70928-2_64)
25. 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). <https://doi.org/10.1007/BFb0056872>
26. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. Evol. Comput.* **7**, 117–132 (2007)

# **Parallel and Distributed Frameworks**



# Introducing an Event-Based Architecture for Concurrent and Distributed Evolutionary Algorithms

Juan J. Merelo Guervós<sup>1</sup>(✉)  and J. Mario García-Valdez<sup>2</sup> 

<sup>1</sup> Universidad de Granada, Granada, Spain  
jmerelo@geneura.ugr.es

<sup>2</sup> Instituto Tecnológico de Tijuana, Tijuana, BC, Mexico  
mario@tectijuana.edu.mx

**Abstract.** Cloud-native applications add a layer of abstraction to the underlying distributed computing system, defining a high-level, self-scaling and self-managed architecture of different microservices linked by a messaging bus. Creating new algorithms that tap these architectural patterns and at the same time employ distributed resources efficiently is a challenge we will be taking up in this paper. We introduce KafkEO, a cloud-native evolutionary algorithms framework that is prepared to work with different implementations of evolutionary algorithms and other population-based metaheuristics by using micro-populations and stateless services as the main building blocks; KafkEO is an attempt to map the traditional evolutionary algorithm to this new cloud-native format. As far as we know, this is the first architecture of this kind that has been published and tested, and is free software and vendor-independent, based on OpenWhisk and Kafka. This paper presents a proof of concept, examines its cost, and tests the impact on the algorithm of the design around cloud-native and asynchronous system by comparing it on the well known BBOB benchmarks with other pool-based architectures, with which it has a remarkable functional resemblance. KafkEO results are quite competitive with similar architectures.

**Keywords:** Cloud computing · Microservices  
Distributed computing · Event-based systems · Kappa architecture  
Stateless algorithms · Algorithm implementation  
Performance evaluation · Distributed computing · Pool-based systems  
Heterogeneous distributed systems · Serverless computing  
Functions as a service

## 1 Introduction

Cloud computing is increasingly becoming the dominant way of running the server side of most enterprise applications nowadays, the same as the browser is the standard platform for the client side. Besides the convenience of the pay-as-you-go model, it also offers a way of describing the infrastructure as part of

the code, so that it is much easier to reproduce results and has been a boon for scientific computing. However, programming the cloud means that monolithic applications, that is, applications built on a single stack of services that communicate by layers, are no longer an efficient architectural design for scientific workflows. Cloud architectures favor asynchronous communication over heterogeneous resources, and shifting from mostly sequential and monolithic to an asynchronously parallel architecture will also imply important reformulation of the algorithms in order to take full advantage of these technologies. Cloud-native applications add a layer of abstraction to the underlying distributed computing system, seamlessly integrating different elements in a single data flow, allowing the user to just focus on code and service connections. Services are *native* points in this new architecture, departing from a monolithic or even distributed paradigm to become a loosely collection of services, in fact *microservices* [19], which in many cases are stateless, reacting to some event and *living* only while they are doing some kind of processing. Reactive systems not only allow massive scaling and independent deployment they are also more economical than other monolithic options. Platform as a service (PaaS) or even Container as a Service (CaaS) approaches need to be running all the time in order to maintain their state, so they are paid for their size and time they remain active. At any rate, while one of the main selling points of Functions as a Service (FaaS) is their ultra-fast activation time, from our point of view their most interesting feature is the fact that they provide stateless processing. An important caveat of stateless processing is that algorithms must be adapted to this fact and turned, at least in part, into a series of stateless steps working on a data stream. It is also taken to an atomic extreme with the so-called serverless architectures [20], which allow vendors and users to deploy code as single, stateless functions, that get activated via *rules*, *triggers* or explicitly, reacting to events consumed from a message queue. The first commercial implementation of this kind of architecture was released by Amazon with its Lambda product, to be closely followed by releases by Azure (Microsoft) and Google and OpenWhisk, an open source implementation released by IBM [2].

In this paper we want to introduce KafkEO, a serverless framework for evolutionary algorithms and other population-based systems. The main design objective is to leverage the scaling capabilities of a serverless framework, as well as create a system that can be deployed on different platforms by using free software. Our intention has also been to create an algorithm that is functionally equivalent to an asynchronous, parallel, island-based, EA, which can use parallelism and at the same time reproduce mechanisms that are akin to migration. The island-based paradigm is relatively standard in distributed EA applications, but in our case, we have been using it since it allows for better parallelism and thus performance, at the same time it makes keeping diversity easier while needing fewer parameters to tune.

We will examine the results of this framework using the first five functions of the Noiseless Black-Box-Optimization-Benchmarking (BBOB) testbed [10] part of the COCO (COMparing Continuous Optimisers) platform for comparisons

of real-parameter global optimisers [10]. The framework is compared against another cloud-ready parallel pool based implementation. The implementation is also free software and can be downloaded from GitHub. The rest of the paper is organized as follows. Next we present the state of the art in cloud implementation of evolutionary algorithms, to be followed in Sect. 3 by an introduction to the serverless architecture we will be using as well as our mapping of the evolutionary algorithm to it. Section 4 will present the result of performing experiments with this proof of concept; finally in Sect. 5 will discuss the results, present conclusions and future lines of work.

## 2 State of the Art

In general, scientific computing has followed the trends of the computing industry, with new implementations published almost as soon as new technologies became commercially available, or even before. There were very early implementations of evolutionary algorithms on transputers [21], the world wide web [5] and the first generation of cloud services [12, 16, 17]. However, every new computing platform has its own view of computing, and in many cases that has made evolutionary algorithms move in new directions in order to make them work better in that platform while keeping the spirit of bio-inspiration. For instance, most evolutionary algorithms work in a synchronous way; although there were very early efforts to create asynchronous EAs [6], in general generations proceed one after the other and migration happens in all islands at the same time. However, this mode of working does not fit well with architectures that are heterogeneous and dynamic, which is why there have been many efforts from early on to adapt EAs to this kind of substrate [1, 3, 22].

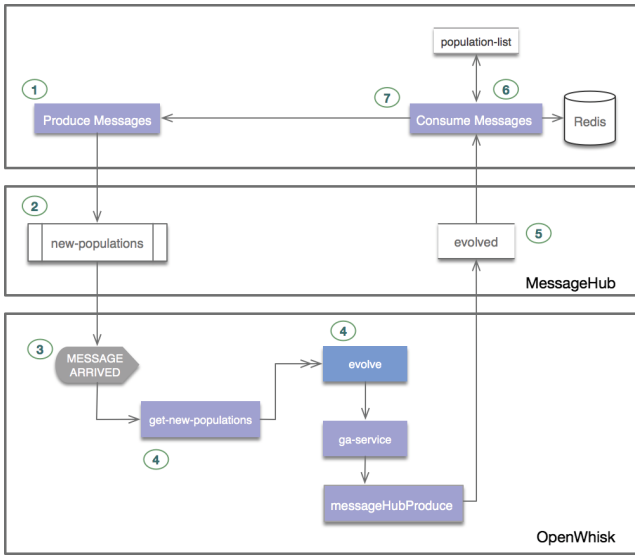
This kind of internet-native applications later on transitioned to using Service-Oriented Architectures (SoA) [14]. While monolithic, that is, including all services in a single computing node and application, SoA were better adapted to heterogeneous environments by distributing services across a network using standard protocols. Several authors implemented evolutionary algorithms over them [8, 13, 15]. However, scaling problems and the extension of cloud deployment and services had made this kind of architectures decline in popularity.

In general, frameworks based in SoA also tried to achieve functional equivalence with parallel or sequential versions of EAs. There is the same tension between functional equivalence and new design in new, cloud based approaches to evolutionary algorithms. Salza and collaborators [16, 17] explicitly and looking to optimize interoperability claim that there is very little need to change “traditional” code to port it to the cloud, implicitly claiming this *functional equivalence* with sequential evolutionary algorithms.

Besides these implementations using well known cloud services, there are new computation models for evolutionary algorithms that are not functionally equivalent to a canonical EA, but have proved to work well in these new environments. Pool based EAs, [4], with a persistent population that can be tapped to retrieve single individuals or pools of them and return evaluated or evolved

sub-populations, have been used for new frameworks such as EvoSpace [9], and proved to be able to accommodate all kinds of ephemeral and heterogeneous resources.

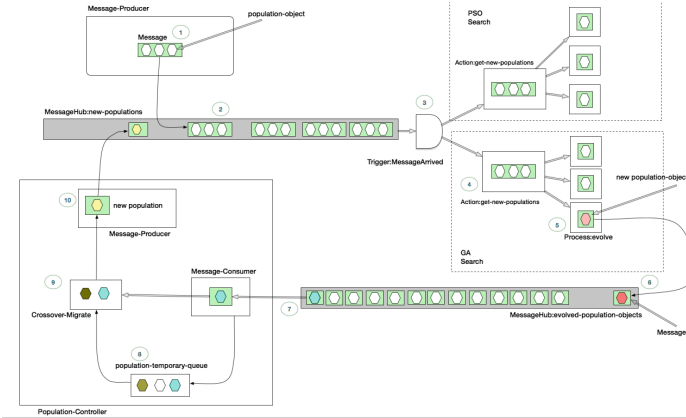
In the serverless, event-based architectures we are going to be targeting in this paper, there has been so far no work that we know of. Similar setups including microservices have been employed by Salza et al. [17]; however, the proposed serverless system adds a layer of abstraction to event-based queuing systems such as the one employed by Salza by reducing it to functions, messages and rules or triggers. We will explain in detail these architectures in the next section.



**Fig. 1.** Chart showing the general picture of the layers of a serverless architecture, including the messages and services that constitute KafkEO, with labels indicating message routes and the software components used for every part.

### 3 Event-Based Architectures and Implementing Evolutionary Algorithms over Them

Microservice architectures share the common trait of consisting of several services with a single concern, that is, providing a single processing value, in many cases stateless, and coupled using lightweight protocols such as REST and messaging buses that carry information from every service to the next. In this case, we are going to be using IBM’s BlueMix service, which includes OpenWhisk as a serverless framework and MessageHub, a vendor implementation of the Kafka messaging service; this last one gives name to the framework we are presenting, called KafkEO (EO stands for Evolving Objects).



**Fig. 2.** A flow diagram of KafkEO, showing message routes, MessageHub topics and the functions that are being used.

The main reason for choosing OpenWhisk and Kafka is availability of resources, but also the fact that all parts of the implementation are open source and can be deployed in desktop machines or other cloud providers by changing the configuration. It is also a good practice to implement free software using free software, making it widely available to the scientific community.

The layers and message flow in the application are shown in Fig. 1, which also includes the evolutionary components. We will focus for the time being in the general picture: a serverless architecture using a messaging service as a backbone, which in this case takes the shape of the Kafka/MessageHub service. These messages are produced and consumed by a service, which can also store them in an external database for their later use; in general, messaging systems are configured to keep messages only for a certain amount of time, and they disappear after that. Messaging queues are organized in *topics* and every topic uses a series of *partitions*, which can be increased for bigger throughput; the functions, hosted in OpenWhisk, execute *actions* triggered by the arrival of new messages; these actions also produce new messages that go back to the MessageHub to continue with the message loop. If all this is hosted in a cloud provider, as it is the case, the MessageHub service will be charged according to a particular cost structure, with partitions taking the most part of the cost, while messages have a relatively small impact.

The evolutionary algorithm mapped over this architecture is represented in Fig. 2. The main design challenge is to try and map an evolutionary algorithm to a serverless, and then stateless, architecture. That part is done in points 1 through 5 of Fig. 2. The beginning of the evolution is triggered from outside the serverless framework (1) by creating a series of Population objects, which we pack (2) to a message in the *new-populations* topic. Population objects are the equivalent to islands or samples in EvoSpace. If Population object is a self-contained population of individuals, represented as a JSON structure.

The arrival of a new population package sets off the `MessageArrived` trigger (3), that is bound to the actions that effectively perform a small number of generations. In this case we give as an example a GA and a PSO algorithms, although only the GA has been implemented for this paper. Any number of GA algorithms (*actions*) can be triggered in parallel by the same message, and new actions can be triggered while others are still working; this phase is then self-scaling and parallel by design.

Population objects are extracted from the message and, for each, a call to an *evolve* process is executed in parallel. The *evolve* process consists of two sequential *actions* (5), first, the *GA Service* function that runs a GA for a certain number of generations, producing a new evolved object, which is then sent to the second action called *Message Produce* responsible of sending the object to the *evolved-population-objects* message queue. The new Population object (6) includes the evolved population and also metadata such as a flag indicating whether the solution has been found, the best individual, and information about each generation. With this metadata a posterior analysis of the experiment can be achieved or simply generating the files used by the BBOB Post-processing scripts.

This queue is polled by a service outside the serverless framework, called *Population-Controller*. This service needs to be stateful, since it implements a buffer that needs to wait until several populations are ready to then mix them (in step #9 in Fig. 2) to produce a new population, that is the result of selection and crossover between several populations coming from the *evolved-population-objects* message queue. Eventually, these mixed populations are returned to the initial queue to return to the *serverless* part of the application. Another task of the *Population-Controller* is to start and stop the experiment. The service must keep the number of Population objects received, then after a certain number is reached, the controller stops sending new messages to the *new-populations topic*. It is important to note, that because of the asynchronous nature of the system, several messages could still arrived after the current experiment is over. The controller must only accept messages belonging to the current process.

This *merging* step before starting evolution takes the place of the *migration* phase and allows this type of framework to work in parallel, since several instances of the function might be working at the exact same time; the results of these instances are then received back by every one of the instances.

In fact, this kind of system would be more functionally equivalent to a pool-based architecture [4], since the queue acts as a pool from where populations are taken and where evolved populations return. Actually, the pool becomes a *stream* in this case, but in fact the pool also evolves, changing its composition, and has a finite size just like the pool. Since pool-based architectures have already proved they work with a good performance, we might expect this type of architecture, being functionally equivalent, to be at least just as efficient and the latter, and better adapted to a cloud-native application.

In this phase where we are creating a proof of concept, there is a single instance of this part. For the time being, it has not been detected as a bottleneck,



although eventually, when the number of functions are working in parallel, it might become one. There are several options for overcoming this problem, the easiest of which is to add more instances of this **Population-controller**. These instances will act in parallel, processing the message queue at different offsets and contributing to population diversity. This will eventually have its influence in the results of the algorithm, so it is better left as future work.

Since we are running just a few functions, the amount of code of KafkEO is quite small compared with other implementations. We use DEAP for all the evolutionary functions, which are written in Python and released in GitHub under the GPL license.

## 4 Experiments and Results

In this section we compare the performance of KafkEO against an implementation of the EvoSpace [9] pool-based architecture, using the first five functions of the Noiseless BBOB testbed [10], which are real-parameter, single-objective, separable functions, namely: Sphere, ellipsoidal, which is highly multimodal, Rastrigin, Bucke-Rastrigin, and the purely lineal function called linear slope. It is expected that the two algorithms achieve similar results as they are functionally equivalent. The EvoSpace implementation follows the basic EvoSpace model in which EvoWorkers asynchronously interact with the population pool by taking samples of the population to perform a standard evolutionary search on the samples, to then return newly evolved solutions back to the pool.

EvoWorkers were implemented in Python with the same code as KafkEO and using DEAP [7] for the GA service function. The code is in the following GitHub repository: <https://hidden.com>. Before each experiment, a script initializes the population on the server, creating the number of individuals specified by the *Pool Size* parameter, this size depends on the dimension of the problem according to the BBOB testbed. When starting each EvoWorker, the following parameters are used: first, the *Sample Size* indicating the number of individuals the worker would take from the server on each interaction, then the *Iterations per Sample* parameter specifies the number of generations or iterations the worker algorithm will run before sending back to the server the resulting population. Finally, the number of times an EvoWorker will take, evolve and return a sample, is indicated by the *Samples per Worker* parameter. The number of EvoWorkers instantiated for the experiment is given by the *GA Workers* parameter. The EvoSpace parameters are shown in Table 1. These parameters are set for each dimension and they indicate the effort in number of evaluations. In both experiments the maximum number of evaluations is  $10^5 \cdot D$ . For instance, for  $D = 2$ , the maximum number of evaluations is 200,000 which is obtained by multiplying the parameters in the first column of Table 1:  $50 \cdot 100 \cdot 20 \cdot 2$ . Also both algorithms limit the search space to  $[-5, 5]^D$ .

On the other hand, the parameters used for KafkEO are shown in Table 2. Every function runs an evolutionary algorithm for the shown number of iterations and with the population size also shown. The number of initial messages act as

**Table 1.** EvoWorker setup parameters,

Dimension	2	3	5	10	20	40
Iterations per Sample	50	50	50	50	50	50
Sample Size	100	100	100	200	200	200
Samples per Worker	20	30	25	25	25	25
GA Workers	2	2	4	5	8	16
Pool Size	250	250	500	1000	2000	4000

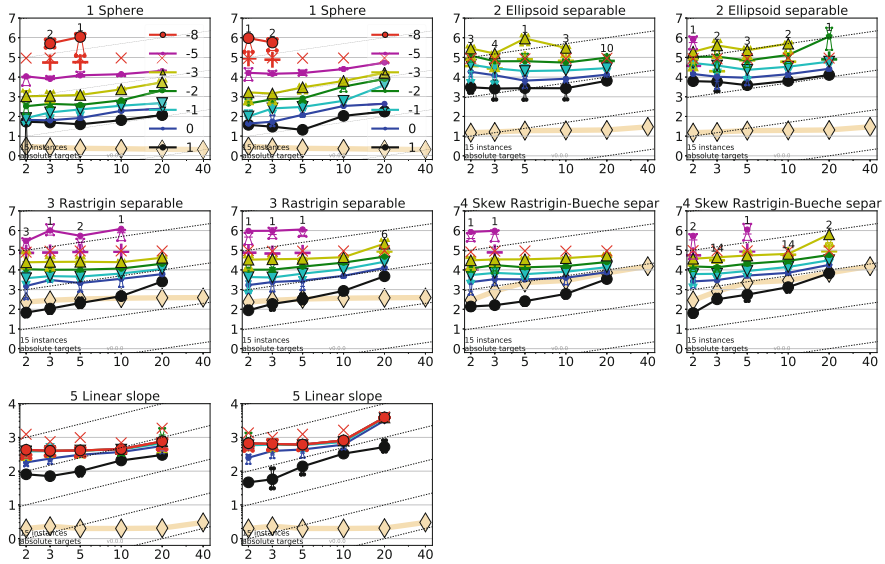
an initial trigger, being thus equivalent to the number of parallel functions or workers; this is the tunable parameter used for increasing performance when the problem dimension, and thus difficulty, increases; the population size is also increased, so that initial diversity is higher. Please note that every population is generated randomly, so that the population size would have to be multiplied by the number of initial messages to get to the initial population involved in the experiment. The effort is limited by the maximum number of messages consumed by the *Population-Controller* from the *evolved-population-objects* message queue. The maximum number is calculated by multiplying the *Maximum Iterations* and *Initial Messages* parameters. Again for a  $10^5 \cdot D$  maximum number of evaluations.

**Table 2.** KafkEO parameters for the BBOB benchmark. Dimensions are the independent variable, the rest of the parameters are changed to adapt to the increasing difficulty.

Dimension	2	3	5	10	20	40
Iterations	50	50	50	50	50	50
Population Size	100	100	100	200	200	200
Initial Messages	2	2	4	5	8	16
Maximum Iterations	2	2	4	5	8	16

The evolutionary algorithm implemented in KafkEO used the same code, also delegating the evolutionary operations to the standard DEAP library, written in Python [7], using 12 for tournament size, a Gaussian mutation with  $\sigma = 0.05$  and a probability between 0.1 and 0.6, plus two point crossover with probability between 0.8 and 1; these are the default parameters. In particular, the tournament size injects a high selective pressure which is known to decrease diversity. The system also allows to set different parameters for every instance; in this proof of concept only two parameters were randomly set, *Mutation Probability* uniformly random in the  $[.1, .6]$  range, and *Crossover Probability* random on  $[.8, 1]$ . This is one deviation from the standard evolutionary algorithm, but has been proved in the past to provide good results without needing to fine tune different parameters [18].

The experiments were performed during the month of January using a paid IBM BlueMix subscription. The totality of experiments costed about \$12. Most of the cost is due to the MessageHub *partitions*, that is, the hosted messaging service itself. The amount paid for the messages in the BlueMix platform is less than one dollar in total; messages are paid by the hundreds of thousands delivered, and are actually not the most expensive part of the implementation of the algorithm. Partitions are essential for a high throughput; a messaging queue will be able to process as many messages as the partitions are able to get through in parallel; this means that cost will scale with the number of messages in a complex way, not simply linearly, and design decisions will have to be taken. The baseline is that the best option is to maximize the number of messages that can be borne by a particular partition, and try to minimize the number of partitions to avoid scaling costs.



**Fig. 3.** Scaling of the running time with dimension to reach certain target values  $\Delta f$ . Lines: average runtime (aRT); Cross (+): median runtime of successful runs to reach the most difficult target that was reached at least once (but not always); Cross (x): maximum number of  $f$ -evaluations in any trial. Notched boxes: interquartile range with median of simulated runs. All values are divided by dimension and plotted as  $\log_{10}$  values versus dimension. Shown is the aRT for fixed values of  $\Delta f = 10^k$  with  $k$  given in the legend. Numbers above aRT-symbols (if appearing) indicate the number of trials reaching the respective target. The light thick line with diamonds indicates the best algorithm from BBOB 2009 for the most difficult target. Horizontal lines mean linear scaling, slanted grid lines depict quadratic scaling. Odd columns (1, 3): EvoSpace; even columns (2, 4): KafkEO.

The results of the comparison are shown in Fig. 3, which follow the classical BBOB 2009 format, which includes the amount of effort devoted to finding a certain fitness level and time needed to do it. Figure 3 was generated by the post-processing script from COCO [10] used in the Black-box Optimization Benchmarking workshop series. The EvoSpace and KafkEO results are shown side by side, only for the sake of comparison, since we are only interested for the time being in the baseline performance of the proof of concept.

The results obtained show that the basic Genetic Algorithm implemented in KafkEO does not perform very well against the testbed, specially when compared against other nature inspired algorithms like PSO or other hybrid approaches [11]. However, both implementations, shown side by side, reach similar results with the same effort; and the results in this case have been obtained with fewer parameters, with out the need to specify an initial pool size and without tuning the evolutionary algorithm parameters.

This is a problem that pool-based algorithms have: we need to specify the initial number of individuals to place in the pool and have the burden of always keeping a minimum number of individuals in the pool. This is not the case in KafkEO, because there is no need to have a repository for the population. However, population size and the number of generations turned in by every instantiation of the functions have to be tuned, which is something that will have to be left as future work.

## 5 Conclusions

This paper is intended to introduce a simple proof of concept of a serverless implementation of an evolutionary algorithm. The main problem with this algorithm, shared by many others, is to turn something that has state (in the form of loop variables or anything else) into a stateless system. In this initial proof of concept we have opted to create a stateful *mixer* outside the serverless (and thus stateless) platform to be able to perform *migration* and mixing among populations. A straightforward first step would be to parallelize this service so that it can respond faster to incoming evolved populations; however, this scaling up should be done by hand and a second step will be to make the architecture totally serverless by using functions that perform this mixing in a stateless way. This might have the secondary effect of simplifying the messaging services to a single topic, and making deployment much easier by avoiding the desktop or server back-end we are using now for that purpose.

The proof of concept is a good adaptation of an evolutionary algorithm to the serverless architecture, with a performance that is comparable, in terms of number of evaluations, to pool-based architectures. Even if results right now are not competitive, the scalability of the architecture and also the possibilities it offers in terms of tuning parameters for the algorithm, even using heterogeneous functions tapping the same *topic* (channel), offer the chance of improving running time as well as the algorithm itself in terms of number of evaluations. This is an avenue that we will explore in the near future. The whole set of experiments, done

in the cloud with a desktop component, took more than running a single desktop experiment using EvoSpace. However, scaling was linear with problem difficulty, which at least mean that we are not adding an additional level of complexity to the algorithm and might indicate that horizontal or vertical scaling would solve the problem. This kind of scaling also indicates that the stateful part, run in a desktop, has not for this problem size become a bottleneck. Even so, we consider that it is essential to create an algorithm architecture that will be fully serverless and, thus, stateless.

Other changes will go in the direction of testing the performance of the system and computing the cost, so that we can increase the former without increasing the latter. Since there is room for increasing parallelism, we will try different ways of obtaining better algorithmic results by making a parameter sensitivity analysis, including population size, length of evolution runs, and other algorithmic parameters. Once those algorithmic baselines have been set, we will experiment with different metaheuristics such as particle swarm optimization, or even try for heterogeneous functions with different evolutionary algorithm parameters, with the purpose of reducing the number of parameters to set at the start.

**Acknowledgments.** Supported by projects TIN2014-56494-C4-3-P (Spanish Ministry of Economy and Competitiveness) and DeepBio (TIN2017-85727-C4-2-P).

## References

1. Atienza, J., Castillo, P.A., García, M., González, J., Merelo, J.: Jenetic: a distributed, fine-grained, asynchronous evolutionary algorithm using Jini. In: Wang, P.P. (ed.) Proceedings of JCIS 2000 (Joint Conference on Information Sciences), vol. I, pp. 1087–1089 (2000). ISBN: 0-9643456-9-2
2. Baldini, I., et al.: Cloud-native, event-based programming for mobile applications. In: Proceedings - International Conference on Mobile Software Engineering and Systems, MOBILESoft 2016, pp. 287–288 (2016)
3. Baugh, J.W., Kumar, S.V.: Asynchronous genetic algorithms for heterogeneous networks using coarse-grained dataflow. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 730–741. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-45105-6\\_88](https://doi.org/10.1007/3-540-45105-6_88)
4. Bollini, A., Piastra, M.: Distributed and persistent evolutionary algorithms: a design pattern. In: Poli, R., Nordin, P., Langdon, W.B., Fogarty, T.C. (eds.) EuroGP 1999. LNCS, vol. 1598, pp. 173–183. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-48885-5\\_14](https://doi.org/10.1007/3-540-48885-5_14)
5. Chong, F.S., Langdon, W.B.: Java based distributed genetic programming on the internet. In: Banzhaf, W., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, p. 1229. Morgan Kaufmann, Orlando, 13–17 July 1999. Full text in technical report CSRP-99-7
6. Coleman, V.: The DEME mode: an asynchronous genetic algorithm. Technical report, University of Massachusetts at Amherst, Department of Computer Science (1989). uM-CS-1989-035
7. Fortin, F.A., Rainville, F.M.D., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: evolutionary algorithms made easy. *J. Mach. Learn. Res.* **13**, 2171–2175 (2012)

8. García-Sánchez, P., González, J., Castillo, P.A., Arenas, M.G., Merelo-Guervós, J.: Service oriented evolutionary algorithms. *Soft Comput.* **17**(6), 1059–1075 (2013)
9. García-Valdez, M., Trujillo, L., Merelo, J.J., Fernández de Vega, F., Olague, G.: The EvoSpace model for pool-based evolutionary algorithms. *J. Grid Comput.* **13**(3), 329–349 (2015). <https://doi.org/10.1007/s10723-014-9319-2>
10. Hansen, N., Auger, A., Mersmann, O., Tusar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting (2016). arXiv preprint [arXiv:1603.08785](https://arxiv.org/abs/1603.08785)
11. 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 Companion on Genetic and Evolutionary Computation, pp. 1689–1696. ACM (2010)
12. Merelo-Guervós, J.J., Arenas, M.G., Mora, A.M., Castillo, P.A., Romero, G., Laredo, J.L.J.: Cloud-based evolutionary algorithms: an algorithmic study. *CoRR abs/1105.6205*, 1–7 (2011)
13. Munawar, A., Wahib, M., Munetomo, M., Akama, K.: The design, usage, and performance of GridUFO: a grid based unified framework for optimization. *Future Gener. Comput. Syst.* **26**(4), 633–644 (2010)
14. Papazoglou, M.P., van den Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. *VLDB J.* **16**(3), 389–415 (2007). <https://doi.org/10.1007/s00778-007-0044-3>
15. Rodríguez, L.G., Diosa, H.A., Rojas-Galeano, S.: Towards a component-based software architecture for genetic algorithms. In: 2014 9th Computing Colombian Conference (9CCC), pp. 1–6, September 2014
16. Salza, P.: Parallel genetic algorithms in the cloud. Ph.D. thesis, University of Salerno, Italy (2017). <https://goo.gl/sDx6mY>
17. Salza, P., Hemberg, E., Ferrucci, F., O'Reilly, U.M.: cCube: a cloud microservices architecture for evolutionary machine learning classification. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 137–138. ACM (2017)
18. Tanabe, R., Fukunaga, A.: Evaluation of a randomized parameter setting strategy for island-model evolutionary algorithms. In: 2013 IEEE Congress on Evolutionary Computation (CEC), pp. 1263–1270. IEEE (2013)
19. Thönes, J.: Microservices. *IEEE Softw.* **32**(1), 116–116 (2015)
20. Varghese, B., Buyya, R.: Next generation cloud computing: new trends and research directions. *Future Gener. Comput. Syst.* **79**, 849–861 (2018). Cited by 2
21. Voigt, H.-M., Born, J., Santibañez-Koref, I.: Modelling and simulation of distributed evolutionary search processes for function optimization. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 373–380. Springer, Heidelberg (1991). <https://doi.org/10.1007/BFb0029778>
22. Zorman, B., Kapfhammer, G.M., Roos, R.S.: Creation and analysis of a JavaSpace-based distributed genetic algorithm. In: PDPTA, pp. 1107–1112 (2002)



# Analyzing Resilience to Computational Glitches in Island-Based Evolutionary Algorithms

Rafael Noguera and Carlos Cotta<sup>(✉)</sup>

Dept. Lenguajes y Ciencias de la Computación,  
Universidad de Málaga, ETSI Informática, Campus de Teatinos,  
29071 Málaga, Spain  
ccottap@lcc.uma.es

**Abstract.** We consider the deployment of island-based evolutionary algorithms (EAs) on irregular computational environments plagued with different kind of glitches. In particular we consider the effect that factors such as network latency and transient process suspensions have on the performance of the algorithm. To this end, we have conducted an extensive experimental study on a simulated environment in which the performance of the island-based EA can be analyzed and studied under controlled conditions for a wide range of scenarios in terms of both the intensity of glitches and the topology of the island-based model (scale-free networks and von Neumann grids are considered). It is shown that the EA is resilient enough to withstand moderately high latency rates and is not significantly affected by temporary island deactivations unless a fixed time-frame is considered. Combining both kind of glitches has a higher toll on performance, but the EA still shows resilience over a broad range of scenarios.

## 1 Introduction

The great success of metaheuristics in the last decades comes partly from the fact that their underlying algorithmic models are very much amenable to deployment in parallel and distributed environments [1]. Indeed, nowadays the use of parallel environments is a key factor to approach the resolution of complex computational problems, and population-based metaheuristics are ideal tools in this context. Specifically, evolutionary algorithms (EAs) have been used on this kind of setting since the 1980s with excellent results and can greatly benefit from parallelism [2, 16]. Following this line, during the last years there has been a growing interest in the use of EAs in distributed computing environments that move away from classical dedicated networks so common in the past. Among such environments we can cite cloud computing [19], P2P networks [14, 28], or volunteer computing (VC) [7], just to name a few.

Some of these emerging computational scenarios –in particular P2P and VC systems– are characterized by several distinctive features which can be summarized under the umbrella term of *irregularity*. Such irregularity is the result of

their being part of interconnected techno-social systems [26] composed of heterogeneous layers of resources with a complex dynamics. Thus, the computational substrate may be composed of a collection of computing nodes with heterogeneous capabilities [4, 17], a feature that has to be accounted for, typically by finding an appropriate balancing of the computational load [6] or by distributing data appropriately [23]. The dynamism of the computational environment is another outstanding feature of these systems: computational nodes can have an uncontrollable dynamics caused by user interventions, interruptions of the network, eventual blockages, delays in communications, etc. The term *churn* is used to denote this phenomenon [24].

While sometimes it may be possible to try to hide these computational irregularities by adding intermediate layers, this can be a formidable challenge in multiple situations [8], and therefore algorithms may need being adapted to run natively (that is, irregularity-aware) on these computational systems. Focusing on EAs, these are fortunately very resilient, at least at a fine-grained scale – see [15, 18]. Furthermore, in cases in which they can be more sensitive to environmental disruptions (e.g., in coarse-grain settings such as the island model [12]), they can be augmented with the necessary functionality to endure some of the difficulties caused by the irregularity of the computational substrate. In line with this, previous work has studied the resilience of EAs in scenarios plagued with instability and heterogeneity [21, 22]. This does not exhaust the sources of irregularity though. Computational glitches can take place in additional different forms, such as traffic overloads or transient computational limitations, which to the best of our knowledge have not been analyzed in this context. Studying the performance of EAs in the presence of these is precisely the focus of this work. To this end we deploy an island-based EA on a simulated computational environment that allows experimenting in a controlled way with different intensities of such computational glitches, namely communication latency and temporary process deactivations. This will be described in more detail in Sect. 2.

## 2 Methodology

As anticipated in previous section, we consider an island-based EA working on a simulated environment, in order to have control on the different issues under study. Each island of the EA runs on a computational node of this environment. In the following we will describe the basic algorithmic details of the EA, as well as how the network and the computational glitches are modeled.

**Algorithmic Model.** The algorithm considered is a steady-state EA with one-point crossover, bit-flip mutation, binary tournament selection and replacement of the worst parent. This algorithm is deployed on a computational environment in which each node hosts an island running an instance of the previously mentioned EA. After each iteration of the basic EA, these islands perform migration (stochastically with probability  $p_{mig}$ ) of single individuals to neighboring islands.



---

**Algorithm 1.** Overview of the island-based evolutionary algorithm.

---

```

for  $i \in [1 \dots n_i]$  do in parallel
  | Initialize( $pop_i$ ) ;           // initialize  $i$ -th island population
  |  $buffer_i \leftarrow \emptyset$  ;   // initialize  $i$ -th migration buffer
end
while  $\neg$  BudgetExhausted() do
  | for  $i \in [1 \dots n_i]$  do in parallel           // basic evolutionary cycle
  | | CheckMigrants( $pop_i$ ,  $buffer_i$ ) ;           // accept migrants (if any)
  | | DoIteration( $pop_i$ ) ;                       // selection, reproduction, replacement
  | | if  $rand() < p_{mig}$  then
  | | | for  $j \in \mathcal{N}_i$  do
  | | | | SendMigrants( $pop_i$ ,  $buffer_j$ ) ;           // send migrants
  | | | end
  | | end
  | end
end
end

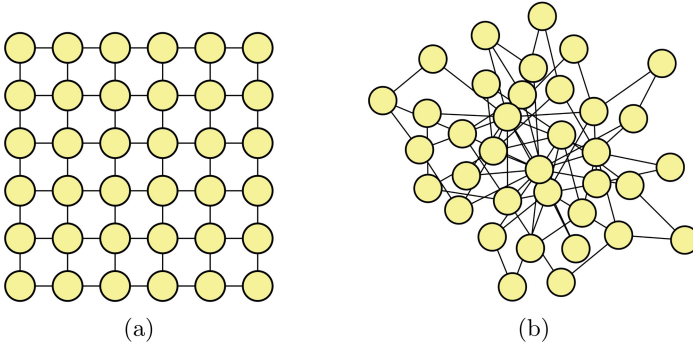
```

---

In each migration event the migrant is randomly selected from the current population and the receiving island inserts it in its population by replacing the worst individual [20]. The whole process is illustrated in Algorithm 1.

**Network Model.** We assume a network composed by  $n_i$  nodes interconnected following a certain topology. More precisely, we consider two possibilities for this purpose: a von Neumann (VN) grid and a scale-free (SF) network. The first one is a classical structure often used in spatially-structured EAs [10, 25] and can be described as a regular toroidal mesh in which each node is connected to four neighbors (those located at unit Manhattan distance), see Fig. 1a. As to the second one, it is a complex network structure commonly found in many natural and artificial systems (e.g., P2P networks) as a consequence of their growth dynamics, i.e., their continuous expansion by the addition of new nodes that attach preferentially to sites that are already well connected [5]. The result is a network topology in which node degrees are distributed following a power-law (i.e., the fraction  $p(d)$  of nodes with  $d$  neighbors goes as  $p(d) \sim d^{-\gamma}$  for some constant parameter  $\gamma$ ). To generate a network of this kind we use the Barabási-Albert model [3], whereby the network is grown from a clique of  $m + 1$  nodes by adding a node at a time, connecting it to  $m$  of the nodes previously added (selected with probability proportional to their degree) where  $m$  is a parameter of the model. Figure 1b shows an example of a SF network. As anticipated by the power-law distribution of node degrees, a few nodes will have a large connectivity and increasingly more nodes will have a smaller number of neighbors.

**Modeling Glitches.** The functioning of the island-based EA described before is disturbed by the presence of perturbations of two types: (i) communication delays and (ii) temporary process deactivations. Both of them can have a diverse



**Fig. 1.** (a) Example of a grid with von Neumann topology (toroidal links not shown for simplicity) (b) Example of a scale-free network with  $m = 2$ .

set of causes in real networks but in the specific context considered in this work, they can –from a very broad and abstract perspective– be considered to stem from the intrinsic properties of the underlying computational substrate, namely the fact it may be often composed of non-dedicated, low-end computational devices. Focusing firstly on network latency, it is a major issue on P2P systems: their decentralized nature makes them inherently more scalable than client-server architectures but also hampers effective communications due to bandwidth limitations and routing information maintenance overhead [13]. This can exert a strong influence on the performance of applications running on this kind of environments, e.g., [29]. To test the extent to which this factor also affects the performance of our island-based EA, we introduce a tunable delay in the communication between islands: whenever individuals are sent for migration purposes, they will only arrive to the destination island after some time  $\lambda$ , measured in a machine-independent way as a number of iterations of the basic evolutionary cycle in Algorithm 1.

The second factor considered is the temporary deactivation of a process. This can be due to a number of factors related to the way the operating system of a certain computational node schedules processes (e.g., the node can engage in swapping, or another high-priority process may kick in –recall we could be considering a VC scheme whereby our algorithm would be using just the spare CPU and bandwidth of a certain device– and the EA can be put to sleep). In such a case, we assume the computation process is still active but its execution is temporarily frozen. This means that it will not execute any evolutionary cycle nor it will send any individuals to neighboring islands (but it cannot prevent other islands from sending migrants to it; these migrants will be simply kept in the input buffer and processed later when the node wakes up). This is related to the issue of instability mentioned in Sect. 1 and can be considered as a slightly more benign form of churn, that is, the island is not completely lost as it would happen when a node goes out of the system and the process is terminated. In order to model this factor we need two parameters  $p_s$  and  $t_s$ : the first one indicates

the probability that each island is put to sleep in each iteration (assumed for simplicity to be constant and fixed for all islands), and the second one denotes the number of cycles it will remain in this dormant state.

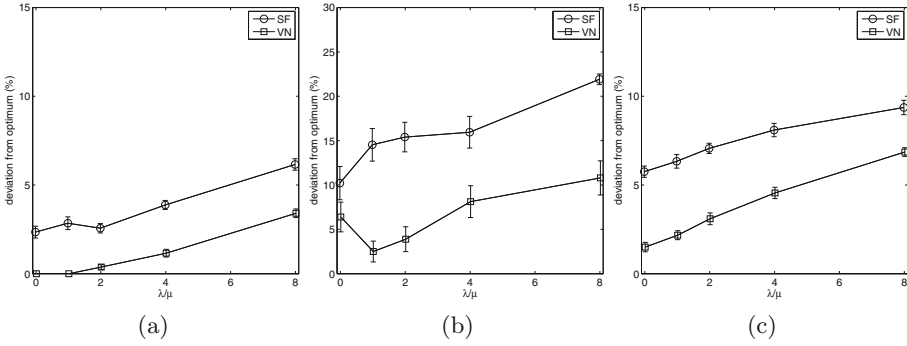
### 3 Experimentation

We consider  $n_i = 64$  islands of  $\mu = 32$  individuals each, and a total number of evaluations  $maxevals = 250,000$ . We use crossover probability  $p_X = 1.0$ , mutation probability  $p_M = 1/\ell$ , where  $\ell$  is the genotype length, and migration probability  $p_{mig} = 1/(5\mu) = 1/160$ . Regarding the network parameters, we use  $m = 2$  in the Barabási-Albert model in order to define the topology of the SF network; in the case of the VN topology, we consider a  $8 \times 8$  toroidal grid. As for the computational glitches, we consider the following settings:

- Latency values  $\lambda = k\mu$  for  $k \in \{0, 1, 2, 4, 8\}$ . Intuitively, these values indicate a communication delay analogous to  $k$  full generations elapsed on an island.
- Node deactivations are done with values  $p_s = k/(\mu n_i)$  and  $t_s = k\mu$ , with  $k \in \{0, 1, 2, 4, 8, 16, 32\}$ . Intuitively, a certain value of  $k$  would indicate both the average number of islands being deactivated per generation and the number of generations they would remain in that state.

The experimental benchmark comprises Deb’s trap function [9] (TRAP, concatenating 32 four-bit traps), Watson et al.’s Hierarchical-if-and-only-if function [27] (HIFF, using 128 bits) and Goldberg et al.’s Massively Multimodal Deceptive Problem [11] (MMDP, using 24 six-bit blocks). These functions provide a scalable benchmark exhibiting properties of interest such as multimodality and deception. We perform 20 simulations for each configuration and measure performance as the percentage deviation from the optimal solution in each case.

Firstly, let us analyze how the latency of communications affects the performance of the algorithm. Figure 2 and Table 1 show the results. As expected, the performance of the algorithm degrades as the latency of communications increases (that is, as we move to the right along the X axis). This can be interpreted in terms of the role of migration: when individuals are migrated the receiving island can benefit both from increased diversity and from quality genetic material. In fact these two factors are intertwined since good (in terms of fitness) fresh information is more likely to proliferate in the target population, and can hence re-focus the search conducted in the island or contribute to drive it out of stagnating states. To the extent that this information starts to constitute a glimpse from the past (as it happens when the latency is in the upper range of values considered), the migrants tend to be less significant in terms of fitness (since the receiving island has more time to evolve and advance in the mean time). They will still carry diversity (and actually in some cases this diversity might be probably higher in comparative terms, since the emitting island was in a less-converged state), but the impact on the receiving island will be less marked, at least in the cases in which latency is high (cf. Table 1), without excluding that for the lower range of latency values considered this diversity



**Fig. 2.** Average deviation from the optimal solution as a function of the latency parameter for SF and VN topologies. (a) TRAP (b) HIFF (c) MMDP.

**Table 1.** Results (20 runs) of the different EAs on SF (upper portion of the table) and VN (lower portion of the table) networks for different latency values. In this table and subsequent ones, the median ( $\tilde{x}$ ), mean ( $\bar{x}$ ) and standard error of the mean ( $\sigma_{\bar{x}}$ ) are indicated. A symbol  $\star|\bullet|o$  is used to indicate statistically significant differences at  $\alpha = 0.01|0.05|0.10$  with respect to the case  $\lambda = 0$  according to a Wilcoxon ranksum test.

SF	TRAP		H-IFF		MMDP		
Latency ( $\lambda$ )	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
0	2.50	2.34 $\pm$ 0.33	11.11	10.21 $\pm$ 1.87	5.99	5.75 $\pm$ 0.32	
$\mu$	2.50	2.84 $\pm$ 0.36	16.67	14.53 $\pm$ 1.83	$o$ 5.99	6.33 $\pm$ 0.38	
$2\mu$	2.50	2.56 $\pm$ 0.26	16.67	15.40 $\pm$ 1.67	$o$ 7.49	7.06 $\pm$ 0.29	$\bullet$
$4\mu$	3.75	3.88 $\pm$ 0.25	$\star$ 19.44	15.95 $\pm$ 1.79	$\bullet$ 7.49	8.10 $\pm$ 0.38	$\star$
$8\mu$	6.25	6.16 $\pm$ 0.32	$\star$ 21.88	21.92 $\pm$ 0.60	$\star$ 8.99	9.37 $\pm$ 0.41	$\star$
VN	TRAP		H-IFF		MMDP		
Latency ( $\lambda$ )	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
0	0.00	0.00 $\pm$ 0.00	0.00	6.39 $\pm$ 1.68	1.50	1.50 $\pm$ 0.27	
$\mu$	0.00	0.00 $\pm$ 0.00	0.00	2.50 $\pm$ 1.17	$o$ 1.50	2.17 $\pm$ 0.25	
$2\mu$	0.00	0.37 $\pm$ 0.13	$\star$ 0.00	3.89 $\pm$ 1.40	3.00	3.10 $\pm$ 0.33	$\star$
$4\mu$	1.25	1.16 $\pm$ 0.22	$\star$ 11.11	8.13 $\pm$ 1.79	4.49	4.55 $\pm$ 0.32	$\star$
$8\mu$	3.75	3.41 $\pm$ 0.24	$\star$ 13.89	10.80 $\pm$ 1.92	$o$ 7.32	6.86 $\pm$ 0.24	$\star$

boost can sometimes provide a minor improvement. The results are also qualitatively similar for both network topologies in which the degradation trend is analogous.

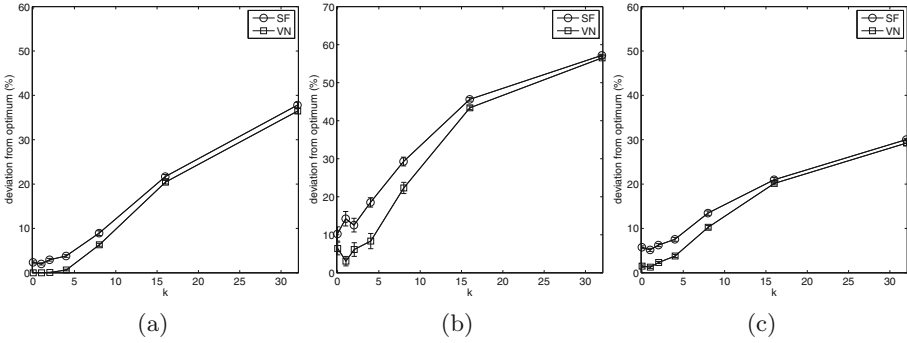
Let us now turn our attention to the effect of temporary island deactivations. The results for different intensities of this factor are shown in Table 2. As it can be seen, there is hardly a degradation of results even for large glitch rates. To interpret this, notice that the presence of dormant islands resembles

**Table 2.** Results (20 runs) of the different EAs on SF (upper portion of the table) and VN (lower portion of the table) networks for different deactivation parameters and a constant number of evaluations. The statistical comparison is done with respect to the case  $k = 0$ .

SF	TRAP		H-IFF		MMDP		
$k$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
0	2.50	$2.34 \pm 0.33$	11.11	$10.21 \pm 1.87$	5.99	$5.73 \pm 0.32$	
1	1.25	$1.94 \pm 0.27$	16.67	$13.11 \pm 1.98$	4.49	$5.00 \pm 0.36$	
2	2.50	$2.37 \pm 0.28$	13.89	$11.08 \pm 1.78$	5.99	$5.82 \pm 0.37$	
4	2.50	$2.34 \pm 0.29$	16.67	$14.57 \pm 1.35$	5.99	$5.90 \pm 0.39$	
8	2.50	$2.88 \pm 0.41$	16.67	$15.07 \pm 1.54$	●	$7.32$	$6.86 \pm 0.34$ ●
16	2.50	$2.66 \pm 0.36$	19.44	$18.44 \pm 1.25$	★	5.99	$6.11 \pm 0.36$
32	2.50	$2.22 \pm 0.28$	16.67	$16.05 \pm 1.71$	●	5.99	$6.03 \pm 0.35$
VN	TRAP		H-IFF		MMDP		
$k$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
0	0.00	$0.00 \pm 0.00$	0.00	$6.39 \pm 1.68$	1.50	$1.50 \pm 0.27$	
1	0.00	$0.00 \pm 0.00$	0.00	$3.06 \pm 1.24$	1.50	$1.26 \pm 0.29$	
2	0.00	$0.06 \pm 0.06$	0.00	$6.10 \pm 1.79$	1.50	$1.56 \pm 0.22$	
4	0.00	$0.12 \pm 0.09$	0.00	$5.42 \pm 1.81$	1.50	$1.48 \pm 0.28$	
8	0.00	$0.06 \pm 0.06$	11.11	$8.06 \pm 1.59$	1.50	$1.95 \pm 0.31$	
16	0.00	$0.25 \pm 0.11$	●	11.11	7.15	$7.15 \pm 1.58$	
32	0.00	$0.25 \pm 0.11$	●	11.11	8.89	$8.89 \pm 1.76$	
					2.83	$2.36 \pm 0.22$	●

transient heterogeneous computational capabilities: within a small time window, each island will have conducted a different number of cycles, which is to some extent analogous to assume they are running on nodes with different computational power; however, on the larger scale, these dormant periods distribute rather homogeneously over all islands, and thus they advance on average at the same rate. Of course, these perturbations become better smoothed out in the longer term the finer they are (hence some effects can be observed in the upper range of values of  $k$ , where glitches are more coarse-grained), but EAs are in any case resilient enough to withstand heterogeneous advance rates without dramatic performance losses [22].

A different perspective can be obtained if we approach these results from the point of view of a fixed time frame, as opposed to a fixed computational effort distributed over a variable time frame. Obviously, the presence of dormant islands contributes to dilute the computational effort exerted over a certain time frame, so studying the resilience of the EA to this dilution is in order. To do so, we consider a time frame dictated by the number of cycles performed by the EA in the base ( $k = 0$ ) case. The results under these conditions are shown in Table 3 and Fig. 3. As expected there is a clear trend of degradation in this case.



**Fig. 3.** Average deviation from the optimal solution as a function of the deactivation parameters for SF and VN topologies and a constant number of cycles. (a) TRAP (b) HIFF (c) MMDP.

**Table 3.** Results (20 runs) of the different EAs on SF (upper portion of the table) and VN (lower portion of the table) networks for different deactivation parameters and a constant number of cycles. The statistical comparison is done with respect to the case  $k = 0$ .

SF	TRAP		H-IFF		MMDP				
$k$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$			
0	2.50	$2.34 \pm 0.33$	11.11	$10.21 \pm 1.87$	5.99	$5.75 \pm 0.32$			
1	1.25	$2.00 \pm 0.25$	16.67	$14.21 \pm 1.91$	5.24	$5.17 \pm 0.35$			
2	2.81	$2.94 \pm 0.30$	16.67	$12.53 \pm 1.80$	5.99	$6.24 \pm 0.34$			
4	3.75	$3.78 \pm 0.30$	★	19.44	$18.51 \pm 1.23$	★	7.49	$7.54 \pm 0.43$	★
8	9.69	$8.94 \pm 0.53$	★	30.38	$29.26 \pm 1.15$	★	13.48	$13.45 \pm 0.50$	★
16	21.56	$21.69 \pm 0.47$	★	45.31	$45.62 \pm 0.46$	★	21.39	$20.98 \pm 0.49$	★
32	38.44	$37.81 \pm 0.60$	★	57.47	$57.20 \pm 0.29$	★	30.29	$30.03 \pm 0.27$	★
VN	TRAP		H-IFF		MMDP				
$k$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$			
0	0.00	$0.00 \pm 0.00$	0.00	$6.39 \pm 1.68$	1.50	$1.50 \pm 0.27$			
1	0.00	$0.00 \pm 0.00$	0.00	$3.06 \pm 1.24$	1.50	$1.27 \pm 0.29$			
2	0.00	$0.06 \pm 0.06$		0.00	$6.11 \pm 1.80$	3.00	$2.32 \pm 0.23$	●	
4	0.00	$0.62 \pm 0.19$	★	5.56	$8.33 \pm 2.01$	3.00	$3.79 \pm 0.33$	★	
8	6.25	$6.31 \pm 0.33$	★	22.05	$22.32 \pm 1.48$	★	10.15	$10.26 \pm 0.41$	★
16	20.63	$20.44 \pm 0.41$	★	43.40	$43.40 \pm 0.49$	★	20.31	$20.15 \pm 0.32$	★
32	37.19	$36.47 \pm 0.52$	★	56.60	$56.53 \pm 0.32$	★	29.29	$29.25 \pm 0.53$	★

Still, the EA can withstand scenarios in which islands remain deactivated for a number of cycles equivalent to twice the population size, although performance significantly degrades for larger deactivation rates/periods in which a much more significant part of the computational effort is lost (up from about 25% for  $k = 4$ ).

**Table 4.** Results (20 runs) of the different EAs on SF networks for different latency and deactivation parameters and a constant number of cycles. Three symbols are shown next to each entry indicating from left to right statistical comparisons with respect to (i)  $\lambda = 0, k = 0$ , (ii) same  $\lambda$  and  $k = 0$ , and (iii) same  $k$  and  $\lambda = 0$  using a Wilcoxon ranksum test. Blanks indicate no statistically significant difference for the corresponding comparison, and  $\star|\bullet|\circ$  have the same meaning as in Tables 1, 2 and 3.

SF		TRAP			H-IFF			MMDP		
$\lambda$	$k$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
0	0	2.50	2.34 ± 0.33		11.11	10.21 ± 1.87		5.99	5.75 ± 0.32	
	1	1.25	2.00 ± 0.25		16.67	14.21 ± 1.91		5.24	5.17 ± 0.35	
	2	2.50	2.41 ± 0.28		19.44	14.68 ± 1.99	●●	5.99	5.23 ± 0.36	
	4	3.13	3.00 ± 0.33		19.44	15.97 ± 1.81	●●	5.99	6.33 ± 0.31	
	8	2.50	2.81 ± 0.35		19.44	17.91 ± 1.34	★★	7.49	7.44 ± 0.46	★★
$\mu$	0	2.50	2.84 ± 0.36		16.67	14.53 ± 1.83	○ ○	5.99	6.33 ± 0.38	
	1	3.75	3.22 ± 0.34	●	16.67	16.58 ± 1.62	●	6.57	6.47 ± 0.40	●
	2	2.50	2.25 ± 0.32		16.67	14.86 ± 1.34	○	6.57	6.39 ± 0.43	○
	4	2.50	2.66 ± 0.22		18.06	15.00 ± 1.89	●	5.99	6.33 ± 0.35	
	8	3.44	2.97 ± 0.30		19.44	17.58 ± 1.62	★	8.07	7.56 ± 0.50	★○
$2\mu$	0	2.50	2.56 ± 0.26		16.67	15.40 ± 1.67	○ ○	7.49	7.06 ± 0.29	● ●
	1	3.44	3.09 ± 0.30	●	16.67	13.69 ± 1.94		7.49	7.16 ± 0.34	★ ★
	2	2.81	3.19 ± 0.23	○ ●	18.06	17.14 ± 1.58	★	5.99	6.41 ± 0.28	●
	4	4.37	3.97 ± 0.30	★★●	18.06	15.00 ± 2.14	●	7.49	7.27 ± 0.35	★ ○
	8	4.06	3.69 ± 0.41	●●	20.14	18.81 ± 1.35	★○	7.49	7.45 ± 0.41	★
$4\mu$	0	3.75	3.88 ± 0.25	★ ★	19.44	15.95 ± 1.79	● ●	7.49	8.10 ± 0.38	★ ★
	1	4.37	4.16 ± 0.25	★ ★	19.44	17.66 ± 1.16	★	7.49	7.59 ± 0.32	★ ★
	2	4.06	4.31 ± 0.32	★ ★	20.14	18.23 ± 1.16	★	8.99	8.26 ± 0.30	★ ★
	4	4.69	4.53 ± 0.39	★ ★	20.49	17.41 ± 1.61	★	8.66	8.30 ± 0.42	★ ★
	8	5.00	4.81 ± 0.24	★●★	20.83	18.17 ± 1.64	★	8.82	8.25 ± 0.27	★
$8\mu$	0	6.25	6.16 ± 0.32	★ ★	21.88	21.92 ± 0.60	★ ★	8.99	9.37 ± 0.41	★ ★
	1	5.94	5.75 ± 0.33	★ ★	21.70	22.35 ± 0.81	★ ★	10.15	9.79 ± 0.30	★ ★
	2	5.94	6.09 ± 0.26	★ ★	22.14	22.47 ± 0.87	★ ★	9.57	9.59 ± 0.33	★ ★
	4	6.25	6.25 ± 0.38	★ ★	21.88	22.76 ± 0.78	★ ★	10.48	10.51 ± 0.36	★★●
	8	5.00	5.47 ± 0.35	★○★	25.00	23.53 ± 1.52	★★★	10.48	10.58 ± 0.48	★○★

Finally, let us consider the cross-effect of having both types of computational glitches. To this end, we have focused on the lower range of deactivation rates ( $0 \leq k \leq 8$ ) in which degradation is moderate at most, leaving aside parameter settings for which extreme degradation already takes place on its own. Also, we have fixed  $t_s = \mu$  in order to have more fine-grained island deactivations and isolate the analysis on the interplay between  $p_s$  and  $\lambda$ . The results are shown

**Table 5.** Results (20 runs) of the different EAs on VN grids for different latency and deactivation parameters and a constant number of cycles. Statistical comparisons follow the same notation as in Table 4.

VN		TRAP			H-IFF			MMDP		
$\lambda$	$k$	$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$		$\tilde{x}$	$\bar{x} \pm \sigma_{\bar{x}}$	
0	0	0.00	0.00 $\pm$ 0.00		0.00	6.39 $\pm$ 1.68		1.50	1.50 $\pm$ 0.27	
	1	0.00	0.00 $\pm$ 0.00		0.00	3.06 $\pm$ 1.24		1.50	1.27 $\pm$ 0.29	
	2	0.00	0.00 $\pm$ 0.00		0.00	3.33 $\pm$ 1.36		1.50	1.76 $\pm$ 0.25	
	4	0.00	0.12 $\pm$ 0.09		0.00	4.72 $\pm$ 1.35		1.50	1.78 $\pm$ 0.23	
	8	0.00	0.19 $\pm$ 0.10	o o	11.11	8.06 $\pm$ 1.42		3.00	3.13 $\pm$ 0.25	**
$\mu$	0	0.00	0.00 $\pm$ 0.00		0.00	2.50 $\pm$ 1.17	oo	1.50	2.17 $\pm$ 0.25	
	1	0.00	0.25 $\pm$ 0.11	●●●	0.00	5.28 $\pm$ 1.37		3.00	2.77 $\pm$ 0.27	*o*
	2	0.00	0.19 $\pm$ 0.10	ooo	5.56	5.56 $\pm$ 1.27	o	2.83	2.42 $\pm$ 0.22	● o
	4	0.00	0.28 $\pm$ 0.13	●●	0.00	5.56 $\pm$ 1.61		3.00	2.50 $\pm$ 0.24	● ●
	8	0.00	0.37 $\pm$ 0.13	**	0.00	4.03 $\pm$ 1.63	o	3.00	3.56 $\pm$ 0.31	**
$2\mu$	0	0.00	0.37 $\pm$ 0.13	* *	0.00	3.89 $\pm$ 1.40		3.00	3.10 $\pm$ 0.33	* *
	1	0.00	0.44 $\pm$ 0.14	* *	0.00	4.44 $\pm$ 1.43		3.00	3.05 $\pm$ 0.25	* *
	2	0.00	0.50 $\pm$ 0.14	* *	0.00	3.33 $\pm$ 1.36		3.00	3.52 $\pm$ 0.29	* *
	4	0.00	0.56 $\pm$ 0.14	* ●	5.56	6.39 $\pm$ 1.52		3.00	3.34 $\pm$ 0.33	* *
	8	1.25	0.69 $\pm$ 0.14	* *	0.00	5.56 $\pm$ 1.45		4.49	4.68 $\pm$ 0.31	***
$4\mu$	0	1.25	1.16 $\pm$ 0.22	* *	11.11	8.13 $\pm$ 1.79		4.49	4.55 $\pm$ 0.32	* *
	1	1.25	1.34 $\pm$ 0.21	* *	0.00	2.22 $\pm$ 1.24	o●	4.49	4.89 $\pm$ 0.26	* *
	2	1.25	0.97 $\pm$ 0.16	* *	0.00	3.33 $\pm$ 1.36	●	5.66	5.35 $\pm$ 0.30	* *
	4	1.25	1.50 $\pm$ 0.19	* *	0.00	1.88 $\pm$ 1.30	●*o	4.49	4.64 $\pm$ 0.26	* *
	8	1.56	1.81 $\pm$ 0.23	*●*	5.56	7.57 $\pm$ 1.83		5.99	6.11 $\pm$ 0.32	***
$8\mu$	0	3.75	3.41 $\pm$ 0.24	* *	13.89	10.80 $\pm$ 1.92	o o	7.32	6.86 $\pm$ 0.24	* *
	1	3.75	3.66 $\pm$ 0.28	* *	13.89	12.26 $\pm$ 1.80	● *	7.49	7.51 $\pm$ 0.27	* *
	2	3.75	3.56 $\pm$ 0.29	* *	16.67	13.56 $\pm$ 1.91	* *	7.49	7.70 $\pm$ 0.24	***
	4	3.75	3.66 $\pm$ 0.27	* *	19.44	15.78 $\pm$ 1.69	*●*	7.49	7.41 $\pm$ 0.29	* *
	8	4.37	4.47 $\pm$ 0.26	***	16.67	14.51 $\pm$ 1.79	* *	8.66	8.44 $\pm$ 0.43	***

in Tables 4 and 5. As it can be seen, both factors strongly interact in degrading performance: if we inspect the first block in either table (corresponding to having no latency) we observe that performance differences only start to become significant for larger values of  $p_s$ ; however, remaining blocks are plagued with significant performance differences, even for small values of  $p_s$ . Furthermore, we can see that having  $p_s > 0$  can provoke significant differences in scenarios in which the mere presence of latency would not suffice, cf. Table 1. It is nevertheless interesting to observe that this latter factor, namely latency, seems to have a stronger influence in the performance of the algorithm in this scenario,



as indicated by the fact that turning off deactivations for a given latency value does not usually provide a significant difference (that is, unless the former is typically in the upper end of its range) whereas the converse is often the case.

## 4 Conclusions

Resilience is a property that any algorithm running on an irregular computational environment should feature. Evolutionary algorithms are in this sense well-prepared thanks to the intrinsic resilience provided by their population-based nature. In particular, we have shown in this work that an island-based EA can withstand significant computational glitches without major performance losses. Indeed, the range of latency values and deactivation rates for which noticeable degradation takes place can be considered at the very least moderately high (e.g., latency values larger than a couple of generations of the EA). This complements previous findings that showed both the sensitivity of these techniques to more serious disruptions (such as node failures) and their amenability for being endowed with mechanisms to endure such severe glitches. In this sense, it would be of the foremost interest to study harder scenarios integrating node failures with the computational perturbations considered in this work, analyzing how the EA can react to the corresponding variety of fluctuations in the computational landscape. Such a study could certainly encompass other algorithmic variants of EAs, as well as additional network topologies. The study could be also conducted along other dimensions such as the effect that the migration probability can have in order to counteract glitches.

**Acknowledgements.** This work is supported by the Spanish Ministerio de Economía and European FEDER under Projects EphemeCH (TIN2014-56494-C4-1-P) and DeepBIO (TIN2017-85727-C4-1-P) and by Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

## References

1. Alba, E.: *Parallel Metaheuristics: A New Class of Algorithms*. Wiley, Hoboken (2005)
2. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. *IEEE Trans. Evol. Comput.* **6**(5), 443–462 (2002)
3. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. *Rev. Mod. Phys.* **74**(1), 47–97 (2002)
4. Anderson, D.P., Reed, K.: Celebrating diversity in volunteer computing. In: *Proceedings of the 42nd Hawaii International Conference on System Sciences, HICSS 2009*, pp. 1–8. IEEE Computer Society, Washington (2009)
5. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. *Science* **286**(5439), 509–512 (1999)
6. Beltrán, M., Guzmán, A.: How to balance the load on heterogeneous clusters. *Int. J. High Perform. Comput. Appl.* **23**, 99–118 (2009)

7. Cole, N.: Evolutionary algorithms on volunteer computing platforms: the Milky-Way@Home project. In: de Vega, F.F., Cantú-Paz, E. (eds.) *Parallel and Distributed Computational Intelligence. Studies in Computational Intelligence*, vol. 269, pp. 63–90. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-10675-0\\_4](https://doi.org/10.1007/978-3-642-10675-0_4)
8. Cotta, C., et al.: Ephemeral computing and bioinspired optimization - challenges and opportunities. In: *7th International Joint Conference on Evolutionary Computation Theory and Applications*, pp. 319–324. SCITEPRESS, Lisboa, Portugal (2015)
9. Deb, K., Goldberg, D.: Analyzing deception in trap functions. In: Whitley, L. (ed.) *Second Workshop on Foundations of Genetic Algorithms*, pp. 93–108. Morgan Kaufmann Publishers, Vail (1993)
10. Dorronsoro, B., Alba, E.: *Cellular Genetic Algorithms Operations Research/Computer Science Interfaces*, vol. 42. Springer, Heidelberg (2008). <https://doi.org/10.1007/978-0-387-77610-1>
11. Goldberg, D., Deb, K., Horn, J.: Massive multimodality, deception and genetic algorithms. In: Männer, R., Manderick, B. (eds.) *Parallel Problem Solving from Nature - PPSN II*, pp. 37–48. Elsevier Science Inc., New York (1992)
12. Hidalgo, J., Lanchares, J., Fernández de Vega, F., Lombrana, D.: Is the island model fault tolerant? In: Thierens, D., et al. (eds.) *Genetic and Evolutionary Computation - GECCO 2007*, pp. 2737–2744. ACM Press, New York (2007)
13. Kumar, P., Sridhar, G., Sridhar, V.: Bandwidth and latency model for DHT based peer-to-peer networks under variable churn. In: *2005 Systems Communications (ICW 2005, ICHSN 2005, ICMCS 2005, SENET 2005)*, pp. 320–325. IEEE August 2005
14. Laredo, J., Castillo, P., Mora, A., Merelo, J.J.: Evolvable agents, a fine grained approach for distributed evolutionary computing: walking towards the peer-to-peer computing frontiers. *Soft Comput.* **12**(12), 1145–1156 (2008)
15. Laredo, J., Castillo, P., Mora, A., Merelo, J.J., Fernandes, C.: Resilience to churn of a peer-to-peer evolutionary algorithm. *Int. J. High Perform. Syst. Archit.* **1**(4), 260–268 (2008)
16. Lässig, J., Sudholt, D.: General scheme for analyzing running times of parallel evolutionary algorithms. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) *Parallel Problem Solving from Nature - PPSN XI*, pp. 234–243. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_24](https://doi.org/10.1007/978-3-642-15844-5_24)
17. Lastovetsky, A.: Heterogeneous parallel computing: from clusters of workstations to hierarchical hybrid platforms. *Supercomput. Front. Innovations* **1**(3), 70–87 (2014)
18. Lombrana González, D., Fernández de Vega, F., Casanova, H.: Characterizing fault tolerance in genetic programming. *Future Generation Computer Systems* **26**(6), 847–856 (2010)
19. Meri, K., Arenas, M., Mora, A., Merelo, J.J., Castillo, P., García-Sánchez, P., Laredo, J.: Cloud-based evolutionary algorithms: an algorithmic study. *Nat. Comput.* **12**(2), 135–147 (2013)
20. Nogueras, R., Cotta, C.: An analysis of migration strategies in island-based multimetric algorithms. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) *PPSN 2014. LNCS*, vol. 8672, pp. 731–740. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10762-2\\_72](https://doi.org/10.1007/978-3-319-10762-2_72)
21. Nogueras, R., Cotta, C.: Self-healing strategies for memetic algorithms in unstable and ephemeral computational environments. *Nat. Comput.* **16**(2), 189–200 (2017)

22. Nogueras, R., Cotta, C.: Analyzing self- $\star$  island-based memetic algorithms in heterogeneous unstable environments. *Int. J. High Perform. Comput., Appl* (2016). <https://doi.org/10.1177/1094342016678665>
23. Renard, H., Robert, Y., Vivien, F.: Data redistribution algorithms for heterogeneous processor rings. *Int. J. High Perform. Comput. Appl.* **20**, 31–43 (2006)
24. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: 6th ACM SIGCOMM Conference on Internet Measurement - IMC 2006, pp. 189–202. ACM Press, New York (2006)
25. Tomassini, M.: Spatially Structured Evolutionary Algorithms Natural Computing Series. Springer, Heidelberg (2005). <https://doi.org/10.1007/3-540-29938-6>
26. Vespignani, A.: Predicting the behavior of techno-social systems. *Science* **325**(5939), 425–428 (2009)
27. Watson, R.A., Hornby, G.S., Pollack, J.B.: Modeling building-block interdependency. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 97–106. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0056853>
28. Wickramasinghe, W., Steen, M.V., Eiben, A.E.: Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: Thierens, D. (ed.) Genetic and Evolutionary Computation - GECCO 2007, pp. 1460–1467. ACM Press, New York (2007)
29. Zhou, J., Tang, L., Li, K., Wang, H., Zhou, Z.: A low-latency peer-to-peer approach for massively multiplayer games. In: Despotovic, Z., Joseph, S., Sartori, C. (eds.) AP2PC 2005. LNCS (LNAI), vol. 4118, pp. 120–131. Springer, Heidelberg (2006). [https://doi.org/10.1007/11925941\\_10](https://doi.org/10.1007/11925941_10)



# Spark Clustering Computing Platform Based Parallel Particle Swarm Optimizers for Computationally Expensive Global Optimization

Qiqi Duan, Lijun Sun, and Yuhui Shi<sup>(✉)</sup>

Shenzhen Key Laboratory of Computational Intelligence,  
Department of Computer Science and Engineering,  
Southern University of Science and Technology, Shenzhen 518055, China  
shiyh@sustc.edu.cn

**Abstract.** The increasing demands on processing large-scale data from both industry and academia have boosted the emergence of data-intensive clustering computing platforms. Among them, Hadoop MapReduce has been widely adopted in the evolutionary computation community to implement a variety of parallel evolutionary algorithms, owing to its scalability and fault-tolerance. However, the recently proposed in-memory Spark clustering computing framework is more suitable for iterative computing than disk-based MapReduce and often boosts the speedup by several orders of magnitude. In this paper we will parallelize three mostly cited versions of particle swarm optimizers on Spark, in order to solve computationally expensive problems. First we will utilize the simple but powerful Amdahl's law to analyze the master-slave model, that is, we do quantitative analysis based on Amdahl's law to answer the question on which kinds of optimization problems the master-slave model could work well. Then we will design a publicly available Spark-based software framework which parallelizes three different particle swarm optimizers in a unified way. This new framework can not only simplify the development workflow of Spark-based parallel evolutionary algorithms, but also benefit from both functional programming and object-oriented programming. Numerical experiments on computationally expensive benchmark functions show that a super-linear speedup could be obtained via the master-slave model. All the source code are put at the complementary GitHub project for free access.

**Keywords:** Parallel particle swarm optimizer · Spark clustering computing  
Computationally expensive global optimization

## 1 Introduction

The increasing demands on processing large-scale data from both industry and academia have boosted the emergence of new data-intensive clustering computing platforms. Three of the most successful platforms are the disk-based MapReduce distributed computing paradigm [1–3], the general-purpose GPU heterogeneous computing environment, and the more recently in-memory Spark clustering computing

framework [4]. They have been successfully used in a variety of fields (e.g., database [8], machine learning [9], and business intelligence [10]). It is expected by many researchers from the evolutionary computation (EC) community (e.g., [1, 14, 15]) that parallelizing evolutionary algorithms (EAs) on these big data-driven clustering computing platforms could be beneficial to solve computationally expensive problems.

However, designing easy-to-use, scalable, portable, efficient parallel evolutionary algorithms (PEAs) is a non-trivial task. This is mainly due to the fact that we not only need knowledge of hardware architectures and software platforms, but also need to carefully make trade-offs among different performance metrics. For instance, given a fixed number of function evaluations, if a relatively large population size is chosen on each generation for more powerful parallelization, the slower convergence speed may be obtained. On the contrary, if a relatively small population size is chosen for faster convergence, more execution time may be spent. To alleviate such problems, a large number of PEAs based on these big data-driven computing platforms have been proposed (see [14, 15] for comprehensive surveys).

Among them, both Hadoop MapReduce (e.g., [25, 26]) and GPUs (e.g., [6, 7, 14]) have been widely adopted in the EC field to implement a variety of PEAs. To the best of our knowledge, however, there is few work attempting to use Spark, the state-of-the-art in-memory clustering computing platform, to accelerate PEAs. It has been recently found in [4, 9] that Spark is more suitable for iterative computing than MapReduce and often boost the speedup by more than one order of magnitude. Considering significant advantages of Spark over MapReduce, in this paper we will parallelize three highly cited versions of particle swarm optimizer (i.e., PSO [21], CLPSO [11], and ALCPSO [18]) based on Spark, in order to solve computationally expensive problems. More specifically, the main contributions of this paper are two-fold:

1. Inspired by [1, 12], we will utilize the simple but powerful Amdahl's law to theoretically analyze the master-slave model for PEAs, that is, we do quantitative analysis based on the Amdahl's law, in order to answer the question on which kinds of problems the master-slave model could work well (see Sect. 3.2 for more details).
2. We will design a Spark-based software framework parallelizing three highly cited PSOs. This new framework can not only simplify the development workflow of Spark-based PEAs, but also benefit from both functional programming and object-oriented programming (via Scala [22]). The framework is put at the complementary GitHub project<sup>1</sup> for free access. Numerical experiments on computationally expensive test functions show that a *super-linear* speedup could be obtained via the master-slave model.

The rest of the paper is organized as follows. Section 2 gives a brief review of the state-of-the-art works of PEAs. Section 3 describes Spark, the Amdahl's law for the master-slave model, and three Spark-based PSOs. Section 4 conducts numerical experiments. Section 5 gives conclusions and promising research directions.

---

<sup>1</sup> <https://github.com/QiqiDuan257/parallel-pso-spark>.

## 2 Review

In this section, we will review some state-of-the-art works of PEAs, owing to the limit of space. For more comprehensive surveys, please refer to [14, 15, 23].

The most representative work on MapReduce-based PEAs may be the work recently published by Ferrucci et al. [1]. This paper answered a critical research problem regarding when the MapReduce-based PEAs could execute faster than their sequential versions. In [1], a disadvantage of MapReduce-based PEAs (i.e., the overhead caused by communications with the distributed data storage system) was identified.

Wachowiak et al. [6] parallelized a PSO variant in a heterogeneous clustering computing environment, where many-core GPUs are used to run data-parallel operations (e.g., the matrix-matrix multiplication operation) and multi-core CPUs are used to execute other computationally complex tasks (e.g., complicated nested loops). To obtain a higher speedup, they used float-point precision rather than double-point precision in experiments, which may be not suited for real-world applications where high numerical errors are not allowed. Further, the performance of their algorithm depends heavily on the execution profiling to these test functions.

In the cloud computing environment, Zhan et al. [5] proposed a double-layered distributed differential evolution algorithm called Cloudde. The first layer is responsible for operating multiple independent populations with different parameter settings, while the second layer is in charge of computationally intensive function evaluations distributed on multiple virtual machines. The traditional MPI system was applied to realize Cloudde. Although Cloudde showed good performance on some benchmark functions, its scalability and fault-tolerance ability have not yet been tested and thus constitute an open question.

## 3 Spark-Based Parallel PSOs

This section first compares Spark with other parallel computing technologies. Then Amdahl's law is utilized to quantitatively analyze the master-slave model. Finally, a Spark-based software framework is developed to parallelize three PSOs.

### 3.1 Comparing Spark with Other Parallel Computing Technologies

Currently, spark is the most active open-source big data project [24]. When compared with MapReduce and MPI, two main advantages of Spark are presented below:

1. It provides a simple yet powerful abstract data structure called resilient distributed dataset (RDD) [20], which can utilize distributed RAM efficiently. Conceptually, RDD can be regarded as an immutable distributed shared memory with implicit data parallelism and fault tolerance. Spark hides the details of hardware architectures and communications among nodes, to some extent. With the help of RDD, developers can focus mainly on the algorithmic logic itself.

2. It supports over 100 high-level operators and the mix of functional programming and object-oriented programming, which simplify the development workflow. For instance, once the function evaluations are finished on different workers, the output can be reduced to the fitness value by invoking the *mapValues* method and then returned to the driver by invoking the *collect* method.

For iterative computation, Spark often reduces the execution time by several orders of magnitude when compared with MapReduce [4, 9].

### 3.2 Amdahl's Law for the Master-Slave Model

Owing to its simplicity, the master-slave model has been applied to design a variety of PEAs (e.g. Cloudde [5], PEPNet [13]) over two decades. Empirically, it can perform well when the fitness evaluation time dominates the total execution time of the algorithm. However, there is a lack of rigorous quantitative analysis on the theoretical upper bound of the speedup obtained by PEAs based on the master-slave model, except the early work conducted by Dubreuil et al. [12].

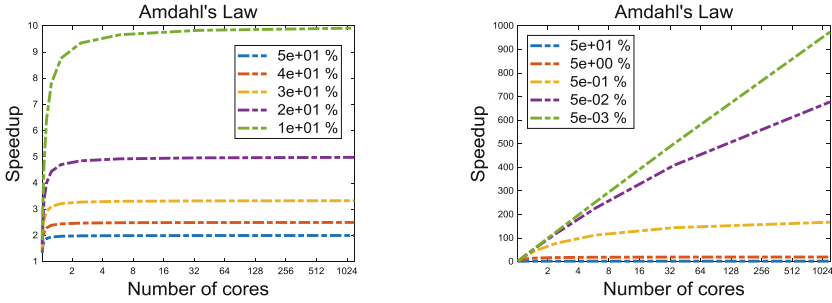
In [12], a complicated mathematical model was proposed to analyze the master-slave model, which takes some realistic factors (e.g., communication cost, network latency) into account. However, accurately estimating these parameter values involved is a non-trivial task in practice. Ferrucci et al. [1] hold that the ideal speedup is equal to the cluster size. Strictly speaking, the cluster size is a looser upper bound, when compared with Amdahl's law. Although they mentioned Amdahl's law in their paper, they did not attempt to use it to further analyze PEAs. Their works [1, 12] motivated us to utilize more general Amdahl's law to theoretically and empirically explain *when and why* the master-slave model could work well, especially under the Spark clustering computing framework. Inspired by Amdahl's law, we will show in Sect. 4 *when* a *super-linear* speedup could be obtained by Spark-based PEAs on computationally intensive continuous benchmark functions.

As stated in Amdahl's law [19, 27], the speedup obtained via parallelization can be calculated according to Eq. 1, as presented below.

$$speedup = \frac{1}{s + \frac{(1-s)}{p}} \quad (1)$$

For Eq. 1, the numerator is the unit time of the sequential program and the denominator is the time spent by the parallel program, where ( $s$ ) is the serial fraction and ( $p$ ) is the parallel level. Figure 1 gives a clear description of Amdahl's law, where different serial fractions are considered ranging from 50% to 0.005%.

Under the Spark clustering computing environment,  $p$  directly corresponds to the total number of *logical* cores used for function evaluations (rather than the total number of slaves). Therefore, we only need to estimate  $s$  for the sequential algorithm, which can be easily done in practice via adding timing. In [27], "*it therefore seems reasonable that there might be a rather even distribution of serial fraction from 0 to 1 over the entire space of computer applications*". We will validate it in the EC field via analyzing several commonly used test functions (see Sect. 4.2 for details).



**Fig. 1.** Amdahl's law [19]. (Different lines correspond to different serial fractions. Note that parallelization may be useful only for highly parallelizable programs [27].)

### 3.3 Spark-Based PEAs Framework for the Master-Slave Model

In this sub-section, we propose a Spark-based PEAs framework, which can in a unified way parallelize three highly cited PSO versions (i.e., PSO [21], CLPSO [11], and ALCP SO [18]) using the master-slave model. For details of these three PSOs, please refer to their corresponding original papers. For their concrete implementation details, please refer to the Scala source code on the complementary GitHub project.

This Spark-based PEAs framework is built on a unified interface with three basic configuration classes and an algorithm base class as parameters. Although sequential algorithms are also supported in this framework, we focus mainly on the parallelization of population-based evolutionary algorithms. Three configuration classes are *ConFuncParams*, *TestParams*, and *AlgoParams*, respectively. The *ConFuncParams* class includes the function name, function dimension, upper and lower search bounds during optimization, and initial upper and lower search bounds at the beginning stage of the search. The *TestParams* class includes the total number of independent tests, and random seeds to initialize the population. The *AlgoParams* class includes the population size, and the maximum of function evaluations, which can be inherited to customize the parameter settings. All algorithm sub-classes inherited from the algorithm base class have the same method called *optimize*, which takes as input the function, and as output the final optimization results. Taking as inputs the function rather than reference or value is one of very useful and flexible characteristics for functional programming. The unified interface takes as input two functions, one of which is the method of the optimization algorithm (i.e., *optimize*) and another of which is the function optimized at hand. Such functional programming-based design increases the scalability and flexibility of the proposed PEAs framework.

To parallelize function evaluations, a simple but resilient data structure built in Spark (i.e., *RDD*) is used. First we use the *parallelize* method of the built-in *SparkContext* object to transfer all individuals from the master to slave nodes. For simplicity, the parallel level is equal to the population size. Then function evaluations tasks can be started by invoking the built-in *mapValues* method. Finally, all the fitness values are returned from different slave nodes to the master by invoking the built-in



*collect* method. For more details, please refer to the public Scala source code. Overall, fulfilling the master-slave model for PEAs is simple and straight under the Spark clustering computing framework.

## 4 Numerical Experiments

In this section, we first describe a private Spark clustering computing platform used here. Then five of most commonly used continuous benchmark functions are empirically analyzed according the Amdahl's law. Finally, comparisons between sequential and parallel PSOs are conducted.

### 4.1 The Spark Clustering Computing Platform

All numerical experiments were conducted on a private Spark clustering computing platform with a total of 160 CPU cores, which consists of a master node (i.e., the driver) and three slave nodes (i.e., the workers). Except that the master node has four 480-GB SSD hard disks working in RAID 1+0 for high-availability, all the nodes have the same hardware and software configurations, as presented in Table 1. The recommendations from the Spark official website [16] are followed to configure the hardware. We also give a practical guidance on the online appendix<sup>2</sup> to illustrate how to rapidly and efficiently deploy a private Spark clustering computing platform. Both Matlab and Scala are also installed on these machines to run sequential algorithms. For Scala, the third-party numerical processing library (i.e., *breeze* [17]) is used.

**Table 1.** Hardware and software configurations for each node.

Hardware	Setting	Software	Version
Machine	Dell® PowerEdge R730 Server	OS	CentOS 7.3.1611
Architecture	64-bit	Spark	2.2.0
CPU	40 Intel® Xeon E5-2640 v4 @ 2.40 GHz	Scala	2.11.11
RAM	64 GB	Sbt	1.0.1
Hard disk	A 960 GB SSD hard disk <i>without</i> RAID	Matlab	R2016b (glnxa64)
Network	1Gbps LAN	Java	1.8.0_131

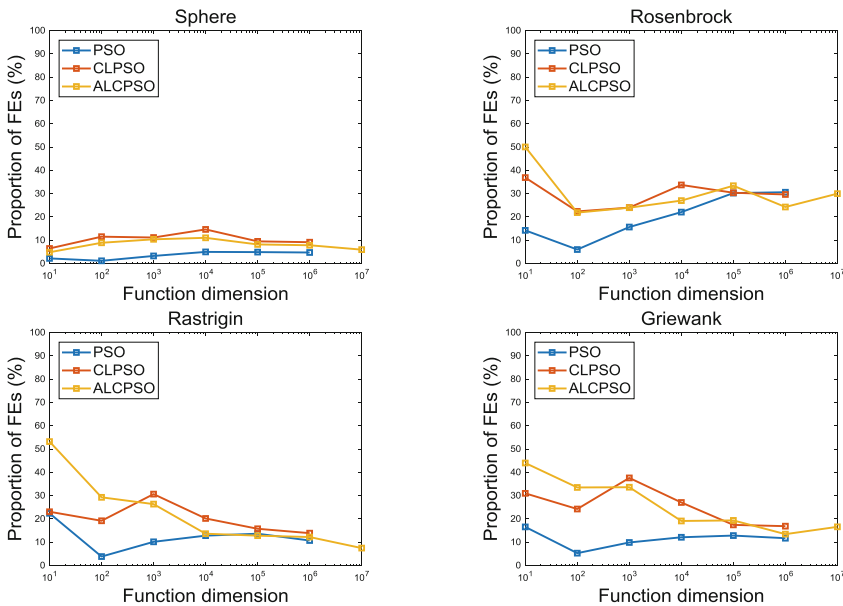
### 4.2 Analyses of Continuous Benchmark Functions

To compare the performance of different algorithms, five well-known continuous benchmark functions (i.e., Sphere, Rosenbrock, Rastrigin, Griewank, and Schwefel12) [18] are used. Because they have different landscape characteristics (e.g., unimodal vs. multimodal, and no-separable vs. separable) and different time complexities (e.g., linear vs. quadratic), we can compare their run time on different scenarios.

<sup>2</sup> <https://github.com/QiqiDuan257/parallel-pso-spark>.

To test the performance of PEAs on computationally expensive problems, a common practice is to use high-dimensional benchmark functions. However, we found that some high-dimensional benchmark functions may be not computationally expensive, assuming that for computationally expensive functions the function evaluations time should dominate the total execution time. According to the proportion of the function evaluations (i.e., FEs) time, these five high-dimensional benchmark functions can be classified empirically into two categories, as presented below:

1. Benchmark functions with a low proportion of the FE time include Sphere, Rosenbrock, Rastrigin, and Griewank. All of them have a *linear* time complexity with the dimension. As we can see from Fig. 2, for PSO, CLPSO, and ALCPSO, almost all of the proportions of FE are less than 50% even when the dimension reaches  $1e7$ . According to Amdahl's law, we can predict that the master-slave model could only obtain a limited speedup on these functions, which is less than 2 even in the ideal case. In the following parts, we will further validate our aforementioned prediction in Spark.
2. Benchmark functions with a high proportion of the FE time on high dimension include Schwefel12 with a quadratic time complexity. As shown in Fig. 3, when the dimension exceeds  $1e3$ , the proportion of the FE time reaches more than 95%. Based on Amdahl's law, it can be theoretically estimated that the master-slave model could show a significant speedup on this function. In the following parts, we will further prove that even a *super-linear* speedup can be achieved on this function in Spark.



**Fig. 2.** Four benchmark functions with a low proportion of the function evaluations time varying with function dimensions for PSO, CLPSO, and ALCPSO.

When using a PEA based on the master-slave model, we may first calculate the proportion of the FE time on its sequential version, and then estimate the theoretical speedup through Amdahl's law. In most cases this speedup may be *over-estimated* owing to a variety of overheads in practice (e.g., communication cost, synchronization barriers, and network latency). However, it is worth noting that we still achieve a *super-linear* speedup in some cases, often caused by *strong scaling* [27].

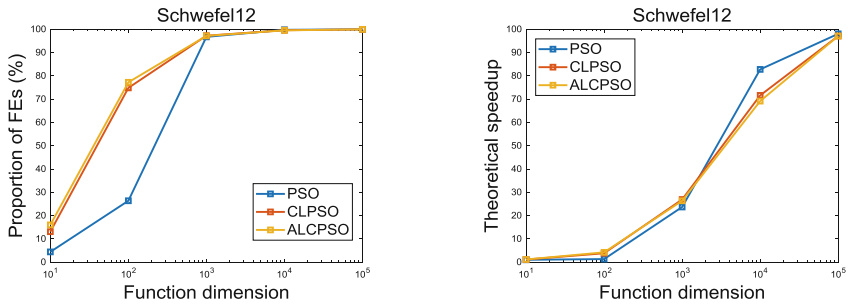


Fig. 3. A benchmark function with a high proportion of the function evaluations time.

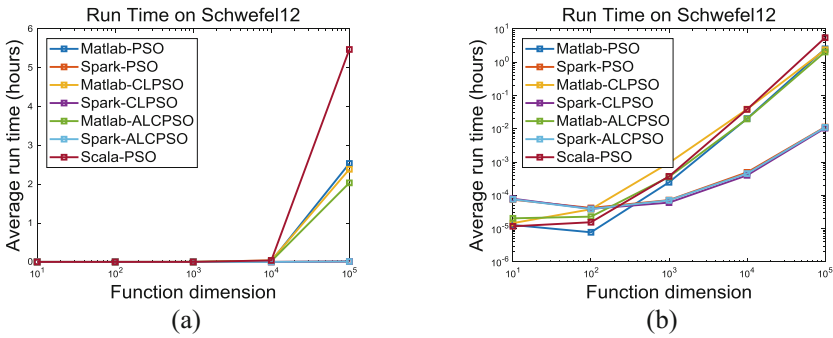
### 4.3 Comparisons on Computationally Expensive Functions

We first compare three Spark-based PSOs with their corresponding sequential versions on the computationally intensive Schwefel12 benchmark function varying function dimensions from  $1e1$  to  $1e5$ . To reduce statistical errors, all numerical experiments were run independently 30 times (except for inefficient sequential versions), and the average run time was recorded, as shown in Fig. 4. To make fair comparisons, for all algorithms, the population size and the maximum of function evaluations are set to 100 and 500, respectively. For high-dimensional problems, a relatively large population size (e.g., 100) is preferred to enhance exploration. Because the total run time of all the sequential algorithms on high dimensions is unacceptable for the large number of FE, a relatively small number of FE (i.e., 500) is used here. Other parameter settings of all algorithms follow the suggestions given in their corresponding original papers. Considering the repeatability of the experiment, all data and source code are freely available on the complementary GitHub project.

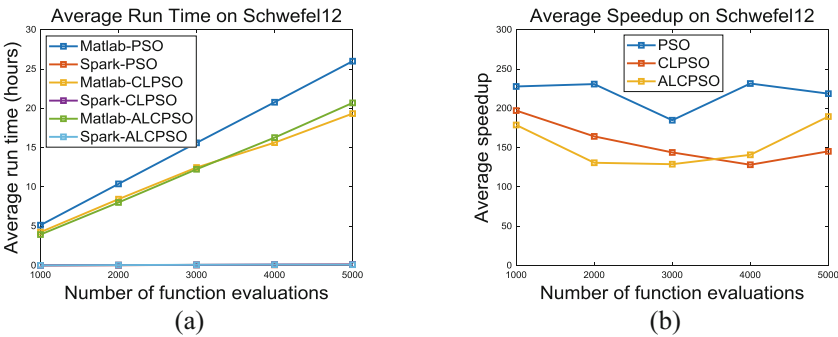
As we can see from Fig. 4, all three Spark-based PSOs can obtain the significant speedup on high dimensions, when compared with their corresponding Matlab-based sequential versions. More specifically, on  $1e3$ ,  $1e4$ , and  $1e5$  dimensions, Spark-based PSO, CLPSO and ALCPSO achieve the (3x, 41x, 224x), (6x, 50x, 194x), and (5x, 44x, 184x) speedup, respectively. However, on 10 and 100 dimensions, since the communications overheads between the master and all slaves cancel out the speedup obtained via parallelization, even worse results are obtained.

To test the scalability of the proposed algorithms on the function with the  $1e5$  dimension, we linearly increased the maximum of FE from 1000 to 5000 with step 1000. To reduce statistical errors, all numerical experiments were run independently 30 times for all three Spark-based parallel PSOs (except for inefficient sequential

contenders), and the average run time was summarized, as presented in Fig. 5. It can be observed from Fig. 5 that all three parallel PSOs can obtain the *super-linear* speedup on this high-dimensional, computationally expensive function. On the contrary, the time complexities of all three Matlab-based sequential versions linearly rise with the number of FE. For parallel PSOs, some stability issues raise with the increasing number of FE, which will be analyzed in Fig. 6.



**Fig. 4.** Comparisons of run time for Three Spark-based parallel PSOs *versus* sequential counterparts on Schwefel12 varying with function dimensions. (Since some lines are condensed into one single line in the left figure (a) owing to the large magnitude of the y-axis, we enlarge them in the right figure (b) via logarithmizing the y-axis.)



**Fig. 5.** Comparisons of speedup for Spark-based parallel PSOs *versus* Matlab-based sequential counterparts on 100000-dimensional Schwefel12 varying with number of function evaluations. (Note that some lines are condensed into one single line in the left figure (a) owing to the large magnitude of the y-axis.)

To further analyze the stability (i.e., fault-tolerance ability) of the proposed parallel algorithms, we plotted the boxplots of the execution time for all three Spark-based PSOs in Fig. 6. We can see that there are some outliers, which take approximately up to 3x times than typical runs. Although more time is spent, the program could

automatically be recovered from the struggling state which may be caused by the underlying network instability. In fact, the good fault-tolerance ability of Spark has been empirically proven in industry [4], which is one advantage over MPI in practice.

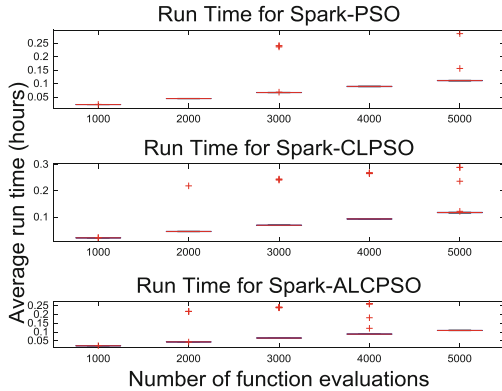


Fig. 6. Boxplot of the execution time obtained on 30 independently runs.

#### 4.4 Comparisons on Functions with Linear Time Complexity

We conducted experiments on four high-dimensional yet computationally-cheap benchmark functions. All experiments were run independently 30 times. For all four functions, the dimension and maximum of FE are set to  $1e5$  and 500, respectively. For all algorithms used here, the population size is set to 100.

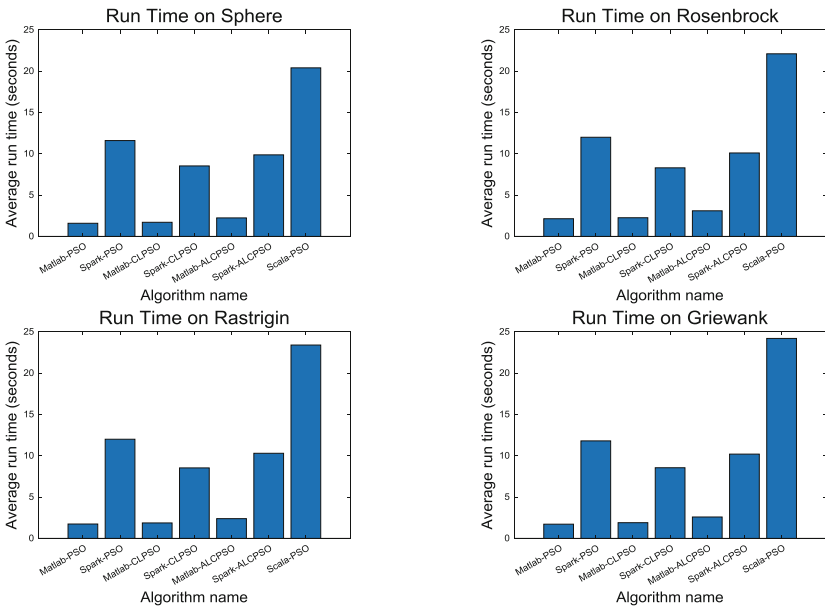


Fig. 7. Comparisons of run time on four computationally-cheap benchmark functions.

As we can see from Fig. 7, three Spark-based parallel PSOs do not obtain any speedup on computationally-cheap benchmark functions, when compared with their corresponding sequential counterparts. This is due to the fact that the communication and synchronization costs among the master and all slaves heavily exceed the parallelization benefit. The “*one-size-fits-all*” parallelization strategy may not exist.

## 5 Conclusions and Future Research Directions

In this paper we first analyzed the speedup of PEAs using the master-slave model. According to Amdahl’s law, we pointed out *when* the master-slave model could work well. Then we provided a Spark-based PEAs framework based on which three most cited PSOs have been parallelized using the master-slave model. The experimental results showed that a super-linear speedup could be obtained by the proposed parallel PSOs at least on computationally expensive test functions. However, there are some open questions which are our future research directions and are presented below:

1. The effectiveness and efficiency of the proposed PEAs need to be further tested on more realistic optimization problems (e.g. geostatic correction [6]).
2. For data-intensive function evaluations tasks, how do Spark-based PEAs read data from the distributed file storage system efficiently?

**Acknowledgements.** This work is partially supported by the Ministry of Science and Technology (MOST) of China under the Grant No. 2017YFC0804002, National Science Foundation of China under the Grant No. 61761136008, and Science and Technology Innovation Committee Foundation of Shenzhen under the Grant No. ZDSYS201703031748284. We acknowledge three anonymous reviewers for their valuable comments and Dr. Jun Huang, Hao Tong, Chang Shao, Liang Qu, and Jing Liu for their help.

## References

1. Ferrucci, F., Salza, P., Sarro, F.: Using Hadoop MapReduce for parallel genetic algorithms: a comparison of the global, grid and island models. *Evol. Comput.* **29**, 1–33 (2018). Early Access
2. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. *Commun. ACM* **51**(1), 107–113 (2008)
3. Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. *Commun. ACM* **53**(1), 72–77 (2010)
4. Zaharia, M., Xin, R.S., Wendell, P., et al.: Apache Spark: a unified engine for big data processing. *Commun. ACM* **59**(11), 56–65 (2016)
5. Zhan, Z.H., Liu, X.F., Zhang, H., et al.: Cloudde: a heterogeneous differential evolution algorithm and its distributed cloud version. *IEEE Trans. Parallel Distrib. Syst.* **28**(3), 704–716 (2017)
6. Wachowiak, M.P., Timson, M.C., DuVal, D.J.: Adaptive particle swarm optimization with heterogeneous multicore parallelism and GPU acceleration. *IEEE Trans. Parallel Distrib. Syst.* **28**(10), 2784–2793 (2017)

7. Kan, G., Lei, T., Liang, K., et al.: A multi-core CPU and many-core GPU based fast parallel shuffled complex evolution global optimization approach. *IEEE Trans. Parallel Distrib. Syst.* **28**(2), 332–344 (2017)
8. Thusoo, A., Sarma, J.S., Jain, N., et al.: Hive: a warehousing solution over a map-reduce framework. *Proc. VLDB Endow.* **2**(2), 1626–1629 (2009)
9. Meng, X., Bradley, J., Yavuz, B., et al.: MLlib: machine learning in Apache Spark. *J. Mach. Learn. Res.* **17**(1), 1235–1241 (2016)
10. Armbrust, M., Xin, R.S., Lian, C., et al.: Spark SQL: relational data processing in Spark. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1383–1394. ACM (2015)
11. Liang, J.J., Qin, A.K., Suganthan, P.N., et al.: Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **10**(3), 281–295 (2006)
12. Dubreuil, M., Gagné, C., Parizeau, M.: Analysis of a master-slave architecture for distributed evolutionary computations. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **36**(1), 229–235 (2006)
13. Riessen, G.A., Williams, G.J., Yao, X.: PEPNet: parallel evolutionary programming for constructing artificial neural networks. In: Angeline, P.J., Reynolds, R.G., McDonnell, J.R., Eberhart, R. (eds.) *EP 1997. LNCS*, vol. 1213, pp. 35–45. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0014799>
14. Tan, Y., Ding, K.: A survey on GPU-based implementation of swarm intelligence algorithms. *IEEE Trans. Cybern.* **46**(9), 2028–2041 (2016)
15. Gong, Y.J., Chen, W.N., Zhan, Z.H., et al.: Distributed evolutionary algorithms and their models: a survey of the state-of-the-art. *Appl. Soft Comput.* **34**, 286–300 (2015)
16. Spark Hardware Provisioning Homepage. <http://spark.apache.org/docs/latest/hardware-provisioning.html>. Accessed 02 Apr 2018
17. Scala Breeze Homepage. <https://github.com/scalanlp/breeze>. Accessed 02 Apr 2018
18. Chen, W.N., Zhang, J., Lin, Y., et al.: Particle swarm optimization with an aging leader and challengers. *IEEE Trans. Evol. Comput.* **17**(2), 241–258 (2013)
19. Kirkpatrick, K.: Parallel computational thinking. *Commun. ACM* **60**(12), 17–19 (2017)
20. Zaharia, M., Chowdhury, M., Das, T., et al.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, p. 2. USENIX Association (2012)
21. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: *IEEE World Congress on Computational Intelligence*, pp. 69–73 (1998)
22. Odersky, M., Spoon, L., Venners, B.: *Programming in Scala*. Artima Inc., Mountain View (2016)
23. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. *IEEE Trans. Evol. Comput.* **6**(5), 443–462 (2002)
24. Spark GitHub Homepage. <https://github.com/apache/spark>. Accessed 02 Apr 2018
25. Verma, A., Llorà, X., Goldberg, D.E., et al.: Scaling genetic algorithms using MapReduce. In: *Ninth International Conference on Intelligent Systems Design and Applications*, pp. 13–18. IEEE (2009)
26. Hajeer, M.H., Dasgupta, D.: Handling big data using a data-aware HDFS and evolutionary clustering technique. *IEEE Trans. Big Data* (2017). Early Access
27. Gustafson, J.L.: Amdahl’s law. In: Padua, D. (ed.) *Encyclopedia of Parallel Computing*, pp. 53–60. Springer, Boston (2011). <https://doi.org/10.1007/978-0-387-09766-4>



# Weaving of Metaheuristics with Cooperative Parallelism

Jheisson López<sup>1,2</sup>, Danny Múnera<sup>2</sup>, Daniel Diaz<sup>3</sup>, and Salvador Abreu<sup>4</sup>(✉)

<sup>1</sup> National University of General Sarmiento, Buenos Aires, Argentina  
jalopez@ungs.edu.ar

<sup>2</sup> University of Antioquia, Medellín, Colombia  
danny.munera@udea.edu.co

<sup>3</sup> University of Paris 1/CRI, Paris, France  
daniel.diaz@univ-paris1.fr

<sup>4</sup> University of Évora/LISP, Évora, Portugal  
spa@uevora.pt

**Abstract.** We propose PHYSH (Parallel HYbridization for Simple Heuristics), a framework to ease the design and implementation of hybrid metaheuristics via cooperative parallelism. With this framework, the user only needs encode each of the desired metaheuristics and may rely on PHYSH for parallelization, cooperation and hybridization. PHYSH supports the combination of population-based and single-solution metaheuristics and enables the user to control the tradeoff between intensification and diversification. We also provide an open-source implementation of this framework which we use to model the Quadratic Assignment Problem (QAP) with a hybrid solver, combining three metaheuristics. We present experimental evidence that PHYSH brings significant improvements over competing approaches, as witness the performance on representative hard instances of QAP.

## 1 Introduction

Metaheuristics are often the most efficient approach to address the hardest Combinatorial Optimization Problems (COP). Metaheuristics are high-level procedures using choices (i.e., heuristics) to limit the part of the search space which actually gets visited, in order to make problems tractable. Metaheuristics can be classified in two main categories: *single-solution* and *population-based* methods. Single-solution metaheuristics (S-MH) maintain, modify and stepwise improve on a single candidate solution, hence the term *trajectory-based* metaheuristics. On the other hand, population-based metaheuristics (P-MH), modify and improve a *population*, i.e. a set of individuals corresponding to candidate solutions.

Metaheuristics generally implement two main search strategies: *intensification* and *diversification*, also called exploitation and exploration [1]. Intensification guides the solver to deeply explore a promising part of the search space. In

---

This work was partly funded by FCT under grant UID/CEC/4668/2016 (LISP).



contrast, diversification aims at extending the search onto different parts of the search space [8]. In order to obtain the best performance, a metaheuristic should provide a useful balance between intensification and diversification. By design, some heuristics are good at one but not at the other.

More generally, each metaheuristic can perform differently according to the problem or even instance being solved. A single metaheuristic will also vary depending on its chosen tuning parameters. The current trend is thus to design *hybrid* metaheuristics, by combining different methods in order to benefit from the individual advantages of each one [9]. An effective approach consists in combining an evolutionary algorithm with a single-solution method (very often a local search procedure). These hybrid methods are called *memetic algorithms* [10]. Hybrid metaheuristics tend to be complex procedures, tricky to design, implement and tune, therefore, most of them only combine two methods.

Despite the good results obtained with the use of hybrid metaheuristics, it is still necessary to reduce the processing times needed for harder instances [18]. One possible answer entails resorting to parallel execution [5]. For instance, several instances of a given metaheuristic can be executed in parallel in order to develop concurrent explorations of the search space, either *independently* or *cooperatively* by means of communication between concurrent processes. The first is easiest to implement on parallel computers, as the metaheuristics run oblivious to each other and execution stops as soon as any of them finds a solution [16, 22]. For some problems this provides very good results [3] but in many cases the speedup tends to taper off when increasing the number of processors [13]. A cooperative approach entails adding a communication mechanism in order to share or exchange information among solver instances during the search process [20]. However, designing an efficient cooperative method is a dauntingly complex task [4], and many issues must be solved: *What information is exchanged? Between which processes is it exchanged? When is it exchanged? How is it exchanged? How is the imported data used?* [21]. Moreover, most cooperative choices are problem-dependent (and sometimes even instance-dependent). Bad choices result in poor performance, possibly much worse than what could be obtained with independent parallelism. However, a well-tuned cooperative scheme may significantly outperform the independent approach.

In 2014, we proposed the Cooperative Parallel Local Search framework (CPLS) for the cooperative parallel execution of local search metaheuristics [13, 14]. The user only has to encode the LS procedure and can rely on CPLS to obtain a parallel application able to run concurrently and cooperatively several instances of this LS procedure. At runtime, the outcome is a parallel exploration of the search space with candidate solution interchange. All low-level parallel mechanisms (task creation/destruction, mapping to physical resources, synchronization, communication, termination, ...) are transparently handled by CPLS. CPLS has been successfully used to tackle stable matching problems [15] and very difficult instances of the Quadratic Assignment Problem (QAP) [12]. We later extended CPLS to allow the user to run *different* metaheuristics in parallel. CPLS has enabled a simpler way to hybridize metaheuristics, by exploiting

its solution-sharing cooperative parallelism mechanism. At runtime, the parallel instances of each different metaheuristic communicate their best solutions, and one of them may forgo its current computation to *adopt* a better solution from the others, hoping to converge faster. The expected outcome is that a solution which may be stagnating for one solver, has a chance to be improved on by another metaheuristic. CPLS has been successfully used to develop a very efficient hybrid solver for QAP [11]. However, CPLS was designed for local search metaheuristics: its cooperation mechanisms can only handle single-solution metaheuristics. When pursuing hybridization this limitation becomes too severe.

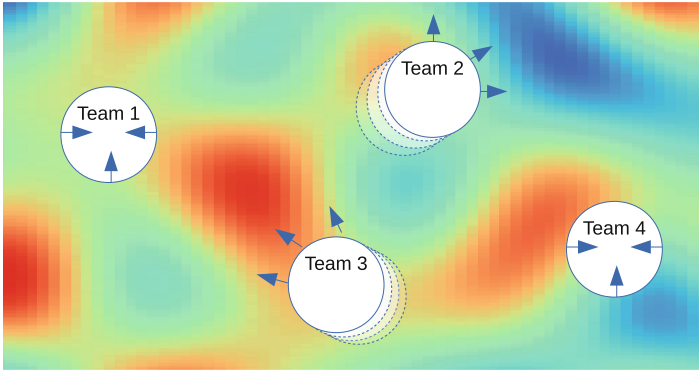
In this paper we propose a framework for the Parallel HYbridization of Simple Heuristics (PHYSH), which eases the implementation of hybrid metaheuristics using cooperative parallelism. As in CPLS, the user only needs to code each of the desired metaheuristics, independently, and may rely on PHYSH to provide both parallelism and cooperation to get “the best of both worlds”. PHYSH is highly parametric and the user has control over the trade-off between intensification and diversification. Single-solutions methods are in charge of intensifying the search while population-based methods can be used to provide diversification through the evolution of a population. We also sketch a prototype implementation, available as an open source library written in the IBM X10 concurrent programming language. Needs only code the desired metaheuristic, PHYSH API. We used this implementation to develop a parallel solver for QAP by hybridizing 3 metaheuristics: a Genetic Algorithm, an Extremal Optimization procedure and a Tabu Search method. The resulting solver performs extremely well on the hardest instances of QAP.

The rest of this paper is organized as follows: in Sect. 2 we describe the framework, while in Sect. 3 we discuss implementation issues. In Sect. 4 we carry out an experimental evaluation on hard QAP instances. Finally, we summarize our results and draw plans for future developments in Sect. 5.

## 2 The PHYSH Framework

The aim of PHYSH is to offer the user an environment for the development of hybrid and parallel metaheuristics. By transparently managing all of the technical details of parallel programming as well as mechanisms for hybridization, PHYSH allows the user to focus on metaheuristic codings and problem modeling. The resulting parallel hybrid search process starts from different points in the search space, attempting to ensure convergence on proper solutions while escaping local extrema. We achieve this with multiple concurrent worker *teams*, each one tasked with visiting a different region of the search space. Figure 1 depicts a search space where red regions contain high-quality solutions which is explored by 4 teams in parallel: 2 teams are intensifying the search in a promising region while the 2 others are diversifying the search in order to reach other rich region.

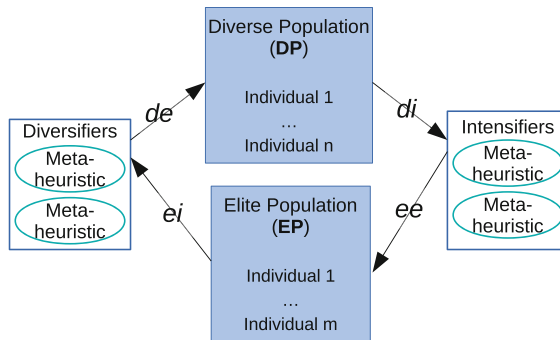
Teams are composed of the following components: a set of search units, a diverse and an elite populations. The main active element of the framework is the *search unit* (SU) which encapsulates a single metaheuristic that can be



**Fig. 1.** PHYSH search process (Color figure online)

either a S-MH or a P-MH. If the SU contains a S-MH, it takes the role of an *intensifier* otherwise (implementing P-MH) it takes the role of a *diversifier*. The elite population (EP) retains the best individuals found by the intensifiers, while the diverse population (DP) holds individuals sent by diversifiers. The interaction patterns between the different components that make up a team establish a parametric four-way migratory flow process (see Fig. 2). In each case a parameter controls the migration frequency.<sup>1</sup>

- *Elite Emigration* (**ee**): from the intensifier worker to the EP.
- *Diverse Emigration* (**de**): from the diversifier worker to the DP.
- *Elite Immigration* (**ei**): from the EP to the diversifier worker.
- *Diverse Immigration* (**di**): from the DP to the intensifier worker.



**Fig. 2.** PHYSH team structure

<sup>1</sup> Terms “immigration” and “emigration” are from the metaheuristics point-of-view.

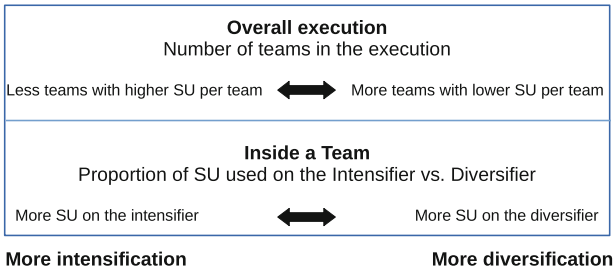
The intensifiers (resp. diversifiers) must apply a *selection policy* to determine which individuals emigrate to the EP (resp. DP). EP and DP population implement an *acceptance policy* for deciding whether the incoming individual is accepted or rejected (discarded). For immigration flows, intensifiers and diversifiers request individuals respectively from DP and EP. Once again a *selection policy* is implemented on the populations to define how to chose an individual and send it to the corresponding entity.

Our framework follows the design principle of separating policy form mechanism. As a result, this process constitutes a flexible interaction model between intensifiers and diversifiers which eases the hybridization of simple metaheuristics, effectively promoting cross-fertilization among different types.

Different mechanisms can be implemented for the same policy e.g., an *elitist* or *non-elitist* mechanism. In the first case we favor elite individuals, while in the second we may, for instance, select the most diverse individual or even adopt a stochastic stance. We may assign several mechanisms for the same policy to a component, in that case the mechanisms are applied in a round-robin fashion until they succeed in the (selection/acceptance) pipeline.

An intuitive configuration could assign elitist mechanism to the intensifiers, non-elitist mechanism to the diversifiers, and both types of mechanism to the populations. We decided to make this a configurable option, as it provides rich choices of search strategy.

In PHYSH, the programmer may easily control the balance between intensification and diversification (see Fig. 3). Take the proportion of SUs used for the intensifiers vs. diversifiers: it may be tuned to achieve a specific balance. For instance, if more intensification is needed for a given instance, one may increase the number of SUs in the role of intensifier. The intensification/diversification level may also be tweaked by varying the number of teams in the execution: given a fixed number of processing units, using more teams with a lower SU count will increase the diversification on the search.



**Fig. 3.** PHYSH intensification-diversification control

The PHYSH framework is designed to adapt to different parallel architectures: shared-memory multiprocessors as well as distributed systems with network-connected MP nodes. SUs are meant to be mapped to physical processors, while teams may be configured very flexibly.

### 3 PHYSH×10: A Prototype Implementation

We implemented our prototype in the X10 programming language which is a high level object-oriented programming language, focused on concurrency and distribution. X10 supports a wide range of parallel platforms and it has been in active development by IBM research since 2004. X10 is based on the Asynchronous Partitioned Global Address Space model (APGAS). Using this model, computation and data are partitioned into *places* which are abstractions for mutable, shared-memory regions that can contain global references to locations in other places, as well as worker threads operating on this memory.

In adoption of common practice for metaheuristics tools, PHYSH×10 presents a clear separation between available metaheuristics and the problems that can be solved. We have implemented a genetic algorithm (GA), a robust tabu search (RoTS) and an extremal optimization (EO) procedure. Consequently, the diversifiers are built from SUs that contain a GA, while the other two metaheuristics are available for the SUs in the intensifiers. Figure 4 displays the main classes of PHYSH×10, a few application-specific ones and their relationships.

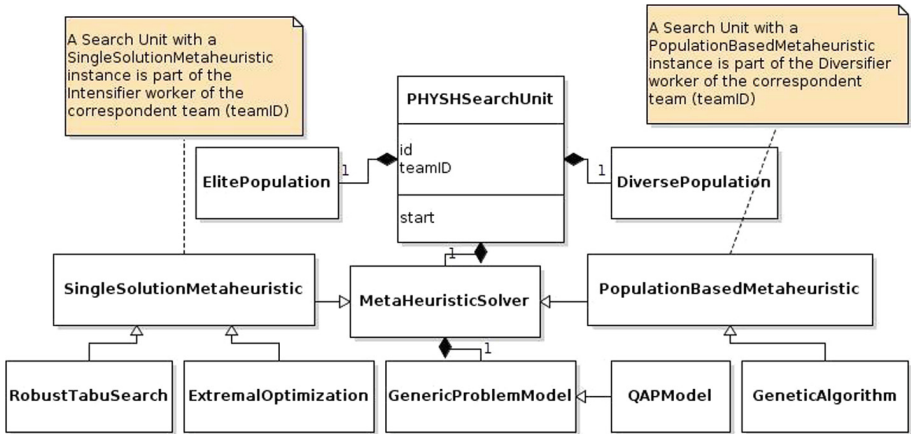


Fig. 4. PHYSH×10 UML diagram of the main classes

PHYSH×10 uses the features offered by X10-APGAS model to assign available physical processing resources. Accordingly, each SU is allocated to an X10 *place*, so that intensifiers and diversifiers operate as a distributed system. As explain above, SUs are grouped to form teams. Each team is composed of `tz` SUs. The number of teams is thus `#cores/tz`. *EP* and *DP* populations are bound to a single SU within each team. These populations have a parametric size i.e., `epz` individuals for EP and `dpz` individuals for DP. Each component implements the most convenient mechanism for the acceptance and selection criteria.

At present, PHYSH×10 provides the following *selection mechanisms*:

- *Best*: best individual found in the search process.
- *Current*: all eligible individuals are selected (for S-MH the current configuration is the unique eligible individual.)
- *Random*: an individual is randomly selected from the eligible set.

The following *acceptance mechanism* are also provided:

- *Elitist*: The individual is accepted if it is better than the worst in the target population (if it is not present yet.)
- *Probabilistic*: The individual is accepted, regardless of its cost, with a given probability (if it is not present yet.)
- *Maximizer*: The individual is accepted if its average distance to the other individuals is greater than a defined threshold.

Intensifiers implement the *current* mechanism for the selection policy i.e., SU sends its current configuration to perform the emigration to the EP. Parameter elite emigration period (**eep**) controls the periodicity of this communication. Intensifiers also request an immigrant individual from DP each diverse immigration period (**dip**). To accept or deny this individual intensifiers implement an *elitist* mechanism for the acceptance policy (for S-MH the target “population” is current solution of the metaheuristic).

Diversifiers implements a *random* mechanism for the selection policy. This mechanism requires a parameter to define the percentage of the population eligible for emigration (**ppfe**). The individual to emigrate is randomly chosen among the top **ppfe**% of the SU’s population (the best individuals). Parameter diverse emigration period (**dep**) controls the periodicity of this emigration process. Diversifiers also request an immigrant from EP each elite immigration period (**eip**). Individual diversifiers implement an *elitist* acceptance mechanism.

To simplify the assignment of these parameters we define two general values: emigration period **ep** and immigration period **ip**. Considering teams of size **tz** (a team embeds **tz** SUs) and a problem of size of **n**, the default values are computed as follows: **eip** = **ep**/**tz**, **dep** = **ip**/**n**, **eep** = **ep**/**n** and **dip** = **ip**.

## 4 Experimental Evaluation

To evaluate the performance of our framework, we developed PHYSH-QAP<sup>2</sup>: a parallel hybrid solver for QAP which combines three metaheuristics: a Genetic Algorithm (GA) [7], a Robust Tabu Search (RoTS) [19] and an Extremal Optimization procedure (EO) [12]. PHYSH-QAP is built on top of PHYSH×10. We consider three sets of very hard benchmarks: the 33 hardest instances of QAPLIB and two sets of even harder instances: Drezners **dreXX** and Palubeckis **InstXX** instances. All experiments have been carried out on a cluster of 16 machines, each with 4 × 16-core AMD Opteron 6376 CPUs running at 2.3 GHz and 128 GB of RAM. The nodes are interconnected with InfiniBand FDR 4 × (i.e., 56 GBPS). We had access to 4 nodes and used up to 32 cores per node.

<sup>2</sup> The source code is available from [https://github.com/jlopezrf/COPSolver-V\\_2.0](https://github.com/jlopezrf/COPSolver-V_2.0).

#### 4.1 Evaluation of PHYSH-QAP on QAPLIB

QAPLIB is a collection of 134 QAP problems of different sizes [2]. The instances are generally named as `nameXX` where `name` corresponds to the first letters of the author and `XX` is the size of the problem. For each instance, QAPLIB also includes the Best Known Solution (BKS), which is sometimes the optimum. Many QAPLIB instances are easy for a parallel solver, we therefore only considered the 33 hardest instances, as reported in [12]. Each problem instance is executed 30 times, stopping as soon as the BKS is reached or when a time limit of 5 min is hit, using 64 cores. PHYSH-QAP was configured with four teams, each of size `tz = 16` embedding 1 diversifier running GA, 8 intensifiers running RoTS and 7 intensifiers running EO. The size for the elite population and the diverse population was set to 4 (`epz = dpz = 4`). The `ppfe` parameter is instance-dependent (we only experimented with values 0, 50 and 100).

Table 1 has all the results. For each instance we have the BKS, the `ppfe` parameter used, the number of times the BKS is reached (across the 30 executions), the Average Percentage Deviation (ADP) which is the average of the 30 relative deviation percentages computed as follows:  $100 \times \frac{Sol - BKS}{BKS}$ , the Best Percentage Deviation (BPD) which corresponds to the relative deviation percentage of the best solution found among the 30 executions, the Worst Percentage Deviation (WPD) which corresponds to the worst solution, the average execution time given in seconds which corresponds to the elapsed (wall) time, and includes the time to install all solver instances, solve the problem communications and the time to detect and propagate the termination and, finally, the average number of times the winning SU adopted an individual from the diverse/elite populations.

On this set of 33 hardest instances, even with a limit of time of 5 min PHYSH-QAP is able to find the BKS at least once for 29 instances. Moreover, it is even able to reach the BKS systematically at each replication for 21 instances. For the 4 remaining instances (`tai80a`, `tai100a`, `tai150b` and `tai256c`), the quality of solutions returned by PHYSH-QAP is very good, around 0.2% of the BKS. The summary row has interesting numbers. The average ADP is only 0.051%, the average BPD is 0.024% and the average WPD is 0.079%. These numbers confirm that all runs provide high quality solutions; even the worst runs provide good results. For instance, in the worst case (`tai80a`), the worst solution among 30 runs is within just 0.547% of the BKS. Performance-wise, PHYSH-QAP averages just 96 s to find a solution. If we do not take into account the 4 unsolved instances (whose time is bounded by the time limit), the average run time is 70 s. The number of adopted configurations on the winning SU is 4.2, on average, showing that the hybridization is effectively taking place.

**Comparison with Another Parallel Hybrid Solver for QAP:** ParEOTS is a hybrid solver for QAP built on the top of the CPLS framework. ParEOTS combines RoTS and EO and has shown to perform very well. Indeed, on the hardest instances of QAPLIB, it outperforms most of state-of-the-art methods [11].

For this comparison we selected the 15 hardest instances from Table 1. We then ran ParEOTS using the parameters reported in [11] in the same execution

**Table 1.** PHYSH-QAP on hard QAPLIB instances (64 cores, timeout = 5 min)

	BKS	ppfe	#BKS	APD	BPD	WPD	Time	#adopt
els19	17212548	50	30	0	0	0	0.0	0.1
kra30a	88900	100	30	0	0	0	0.0	0
sko56	34458	50	30	0	0	0	1.8	0.5
sko64	48498	50	30	0	0	0	2.0	0.3
sko72	66256	50	30	0	0	0	9.8	1.2
sko81	90998	50	30	0	0	0	22.4	1.6
sko90	115534	100	30	0	0	0	104.4	6.3
sko100a	152002	100	27	0.001	0	0.016	129.3	3.4
sko100b	153890	0	30	0	0	0	52.4	1.0
sko100c	147862	0	30	0	0	0	77.5	1.3
sko100d	149576	0	30	0	0	0	64.9	1.2
sko100e	149150	0	30	0	0	0	49.4	0.9
sko100f	149036	100	29	0.000	0	0.005	103.7	2.4
tai40a	3139370	50	20	0.025	0	0.074	173.9	4.7
tai50a	4938796	100	8	0.133	0	0.336	262.0	10.3
tai60a	7205962	0	1	0.242	0	0.368	292.7	9.5
tai80a	13499184	50	0	0.460	0.335	0.547	300.0	8.6
tai100a	21052466	0	0	0.352	0.167	0.463	300.0	22.6
tai20b	122455319	100	30	0	0	0	0.0	0.0
tai25b	344355646	50	30	0	0	0	0.0	0.1
tai30b	637117113	50	30	0	0	0	0.1	1.3
tai35b	283315445	0	30	0	0	0	0.3	1.8
tai40b	637250948	0	30	0	0	0	0.4	2.5
tai50b	458821517	0	30	0	0	0	6.7	0
tai60b	608215054	0	30	0	0	0	10.9	0
tai80b	818415043	0	30	0	0	0	42.0	1.3
tai100b	1185996137	0	29	0.001	0	0.024	143.4	4.9
tai150b	498896643	50	0	0.190	0.085	0.410	300.0	10.1
tai64c	1855928	0	30	0	0	0	0.2	0.1
tai256c	44759294	50	0	0.264	0.211	0.312	300.0	4.4
tho40	240516	0	30	0	0	0	1.1	0.1
tho150	8133398	0	1	0.021	0	0.043	298.8	29.7
wil100	273038	100	26	0.000	0	0.002	144.7	5.2
Summary			<b>771</b>	<b>0.051</b>	<b>0.024</b>	<b>0.079</b>	<b>96.8</b>	<b>4.2</b>



environment as for PHYSH-QAP: same machine, using 64 cores with a time limit of 5 min and 30 repetitions per instance.

**Table 2.** PHYSH-QAP vs ParEOTS (64 cores, timeout = 5 min)

	PHYSH-QAP			ParEOTS		
	#BKS	APD	Time	#BKS	APD	Time
sko81	<b>30</b>	0	22.4	25	0.002	70.6
sko90	<b>30</b>	0	104.4	29	0.000	116.5
sko100a	<b>27</b>	0.001	129.3	25	0.003	128.9
sko100c	<b>30</b>	0	77.5	29	0.000	127.3
tai40a	20	0.025	<b>173.9</b>	20	0.025	184.2
tai50a	<b>8</b>	0.133	262.0	3	0.144	289.8
tai60a	<b>1</b>	0.242	292.7	0	0.270	300.0
tai80a	0	0.460	300	<b>0</b>	<b>0.460</b>	<b>300.0</b>
tai100a	0	<b>0.352</b>	300	0	0.358	300.0
tai100b	<b>29</b>	0.001	143.4	22	0.015	181.4
tai150b	0	0.190	300.0	<b>0</b>	<b>0.130</b>	<b>300.0</b>
tai64c	<b>30</b>	0	0.2	28	0.004	20.0
tai256c	0	<b>0.264</b>	300.0	0	0.272	300.0
tho150	<b>1</b>	0.021	298.8	0	0.019	300.0
wil100	<b>26</b>	0	144.7	14	0.001	213.9
Summary	<b>232</b>	<b>0.113</b>	<b>190.0</b>	195	0.114	208.8

Table 2 presents the results. To compare the two solvers, compare the number of BKS found, then (in case of a tie), the APDs and finally the execution times. For each benchmark, the best-performing solver row is highlighted and the discriminant field is enhanced in bold font. PHYSH-QAP outperforms ParEOTS on 13 out of 15 of the hardest QAPLIB instances while the reverse only occurs for one instance (*tai150b*). Our implementation systematically solves 4 instances which are not fully solved on ParEOTS (*sko81*, *sko90*, *sko100c* and *tai64c*). The summary row shows that PHYSH-QAP obtains a total *#BKS* higher than ParEOTS (232 vs. 195). It is worth noticing that this quality of solutions is obtained in a shorter execution time (190 s vs. 208 s).

## 4.2 Evaluation of PHYSH-QAP on Harder Instances

We evaluated our hybrid solver on two sets of instances, artificially crafted to be very difficult for metaheuristics: the *dreXX* instances proposed by Drezner et al. [6] and the *InstXX* instances by Palubeckis [17]. These instances are generated with a known optimum. For this test we used the same machine, with 128 cores and a time limit of 10 min with 30 repetitions. We used the same framework configuration as in Sect. 4.1 for QAPLIB. We could not yet experiment with different values for *ppfe* so we use *ppfe* = 100 for all instances.

**Table 3.** PHYSH-QAP on Drezner and Palubeckis (128 cores, timeout = 10 min)

	#BKS	APD	best	Time		#BKS	APD	best	Time
dre21	30	0	356	0.0	Inst20	30	0	81536	0.0
dre24	30	0	396	0.0	Inst30	30	0	271092	0.1
dre28	30	0	476	0.0	Inst40	30	0	837900	3.2
dre30	30	0	508	0.1	Inst50	30	0	1840356	7.7
dre42	30	0	764	0.9	Inst60	30	0	2967464	11.8
dre56	30	0	1086	11.5	Inst70	30	0	5815290	35.7
dre72	30	0	1452	90.9	Inst80	30	0	6597966	78.0
dre90	23	2.757	1838	281.2	Inst100	17	0.038	15008994	476.4
dre110	6	14.997	2264	549.4	Inst150	0	0.122	58411484	600.0
dre132	5	11.404	2744	558.2	Inst200	0	0.123	75495960	600.0
Summary	244	2.915		149.2	Summary	227	0.028		181.3

Table 3 presents the results obtained on both benchmarks. Regarding Drezner’s instances, PHYSH-QAP is able to optimally solve all instances. To best of our knowledge, no other dedicated solver for QAP has ever reported an optimal solution either for `dre110` or `dre132` (highlighted in green in the table). Moreover, all instances of size  $n \leq 72$  are systematically solved at each replication. Regarding Palubeckis’ instances, the optimum is found for instances with  $n \leq 100$  (and systematically found at each replication for  $n \leq 80$ ). For size  $n > 100$ , clearly a limit of 10 min is too short. Nevertheless the quality of obtained solutions within this time limit is very good with an APD around 0.12%. It is worth noting that for `Inst150` and `Inst200`, the solution computed by PHYSH-QAP improves on the best solutions ever published (corresponding best costs computed are highlighted in green in Table 3).

## 5 Conclusion and Future Directions

We have proposed PHYSH: a new framework for the efficient resolution of Combinatorial Optimization Problems combining single-solution metaheuristics, population-based metaheuristics, cooperative parallelism and hybridization. We have used our X10 implementation of this framework to construct a hybrid solver for the Quadratic Assignment Problem which combines up to three metaheuristics. This solver turns out to perform exceptionally well, particularly on very hard instances of QAP.

We plan to study the impact of each parameter in more detail; including experimentation with techniques for parameter auto-tuning, e.g. using F-Race. We also plan to add new metaheuristics to the prototype, particularly population-based methods. This enriched implementation we will enable us to address a wider range of problems. Finally, it will be interesting to experiment on different parallel architectures, for instance GPGPUs or Intel MIC, using the X10 language, which greatly abstracts on machine architectural specificities.

## References

1. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Comput. Surv.* **35**(3), 268–308 (2003)
2. Burkard, R.E., Karisch, S., Rendl, F.: QAPLIB - a quadratic assignment problem library. *Eur. J. Oper. Res.* **55**(1), 115–119 (1991)
3. Caniou, Y., Codognet, P., Richoux, F., Diaz, D., Abreu, S.: Large-scale parallelism for constraint-based local search: the costas array case study. *Constraints* **20**(1), 30–56 (2015)
4. Crainic, T., Gendreau, M., Hansen, P., Mladenovic, N.: Cooperative parallel variable neighborhood search for the p-median. *J. Heuristics* **10**(3), 293–314 (2004)
5. Crainic, T., Toulouse, M.: Parallel meta-heuristics. In: Gendreau, M., Potvin, J.Y. (eds.) *Handbook of Metaheuristics*. ISOR, vol. 146, pp. 497–541. Springer, Boston (2010). [https://doi.org/10.1007/978-1-4419-1665-5\\_17](https://doi.org/10.1007/978-1-4419-1665-5_17)
6. Drezner, Z.: The extended concentric tabu for the quadratic assignment problem. *Eur. J. Oper. Res.* **160**(2), 416–422 (2005)
7. Drezner, Z.: Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. *Comput. Oper. Res.* **35**(3), 717–736 (2008)
8. Hoos, H., Stützle, T.: *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann/Elsevier, Burlington (2004)
9. Misevicius, A.: A tabu search algorithm for the quadratic assignment problem. *Comput. Optim. Appl.* **30**(1), 95–111 (2005)
10. Moscato, P., Cotta, C.: Memetic algorithms. In: *Handbook of Applied Optimization*, vol. 157, p. 168 (2002)
11. Munera, D., Diaz, D., Abreu, S.: Hybridization as cooperative parallelism for the quadratic assignment problem. In: Blesa, M.J., et al. (eds.) *HM 2016*. LNCS, vol. 9668, pp. 47–61. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-39636-1\\_4](https://doi.org/10.1007/978-3-319-39636-1_4)
12. Munera, D., Diaz, D., Abreu, S.: Solving the quadratic assignment problem with cooperative parallel extremal optimization. In: Chicano, F., Hu, B., García-Sánchez, P. (eds.) *EvoCOP 2016*. LNCS, vol. 9595, pp. 251–266. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-30698-8\\_17](https://doi.org/10.1007/978-3-319-30698-8_17)
13. Munera, D., Diaz, D., Abreu, S., Codognet, P.: A parametric framework for cooperative parallel local search. In: Blum, C., Ochoa, G. (eds.) *EvoCOP 2014*. LNCS, vol. 8600, pp. 13–24. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44320-0\\_2](https://doi.org/10.1007/978-3-662-44320-0_2)
14. Munera, D., Diaz, D., Abreu, S., Codognet, P.: Flexible cooperation in parallel local search. In: *Symposium on Applied Computing, SAC 2014*, pp. 1360–1361. ACM Press, Gyeongju (2014)
15. Munera, D., Diaz, D., Abreu, S., Rossi, F., Saraswat, V., Codognet, P.: Solving hard stable matching problems via local search and cooperative parallelization. In: *AAAI, Austin, TX, USA* (2015)
16. Novoa, C., Qasem, A., Chaparala, A.: A SIMD tabu search implementation for solving the quadratic assignment problem with GPU acceleration. In: *Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE 2015*, pp. 1–8 (2015)
17. Palubeckis, G.: An algorithm for construction of test cases for the quadratic assignment problem. *Inform. Lith. Acad. Sci.* **11**(3), 281–296 (2000)
18. Saifullah Hussin, M.: *Stochastic local search algorithms for single and bi-objective quadratic assignment problems*. Ph.D. thesis. Université de Bruxelles (2016)

19. Taillard, É.: Robust taboo search for the quadratic assignment problem. *Parallel Comput.* **17**(4–5), 443–455 (1991)
20. Talbi, E.G., Bachelet, V.: COSEARCH: a parallel cooperative metaheuristic. *J. Math. Model. Algorithms* **5**(1), 5–22 (2006)
21. Toulouse, M., Crainic, T., Gendreau, M.: Communication issues in designing cooperative multi-thread parallel searches. In: Osman, I., Kelly, J. (eds.) *Meta-Heuristics: Theory & Applications*, pp. 501–522. Kluwer Academic Publishers, Norwell (1995)
22. Tsutsui, S., Fujimoto, N.: An analytical study of parallel GA with independent runs on GPUs. In: Tsutsui, S., Collet, P. (eds.) *Massively Parallel Evolutionary Computation on GPGPUs*. NCS, vol. 8, pp. 105–120. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-37959-8\\_6](https://doi.org/10.1007/978-3-642-37959-8_6)

# **Applications**



# Conditional Preference Learning for Personalized and Context-Aware Journey Planning

Mohammad Haqqani<sup>(✉)</sup>, Homayoon Ashrafzadeh, Xiaodong Li,  
and Xinghuo Yu

School of Science, Computer Science and Software Engineering, RMIT University,  
Melbourne, Australia  
mohammad.haqqani@rmit.edu.au

**Abstract.** Conditional preference networks (CP-nets) have recently emerged as a popular language capable of representing ordinal preference relations in a compact and structured manner. In the literature, CP-nets have been developed for modeling and reasoning in mainly toy-sized combinatorial problems, but rarely tested in real-world applications. Learning preferences expressed by passengers is an important topic in sustainable transportation and can be used to improve existing journey planning systems by providing personalized information to the passengers. Motivated by such needs, this paper studies the effect of using CP-nets in the context of personalized and context-aware journey planning. We present a case study where we learn to predict the journey choices by the passengers based on their historical choices in a multi-modal urban transportation network. The experimental results indicate the benefit of the conditional preference in passengers' modeling in context-aware journey planning.

**Keywords:** User modeling · Preference learning  
Conditional preferences · CP-nets · Personalized journey planning

## 1 Introduction

Personalized journey planning provides tailored information to the passengers on sustainable transit options through usually web-based journey planner [3]. It seeks to overcome the habitual use of cars, enabling more journeys to be made on bike, foot, or public transport. This is achieved through the provision of personalized information, to increase the passengers' satisfaction using multimodal transit to support a voluntary shift towards more sustainable choices. The planner uses expressed passenger preferences to recommend journeys to the individuals based on his/her circumstances. The power of the individual-based journey planning is that it can often lead to more significant behavior change than a one-solution-fits-all-approach [3].

Currently, the majority of ‘*intelligent*’ commercial journey planners only have a small set of predefined preferences (e.g., preferred highways or public transit modes) made available for passengers to choose from and rank (Yahoo! trip planner, PTV journey planner, Google Maps) [2]. Although these planners are reliable and offer adequate assistance to passengers, they assume the values of passengers’ preferences are independent i.e., the value of one attribute does not influence the passenger’s preference on the value of other attributes [12]. This assumption, however, is not sound in real-world journeying. For example, the weather condition may affect the passengers’ preferences towards the transportation modes that they are willing to take. This issue could be alleviated by incorporating passengers’ preferences and context into the planning process. Here, we refer to the ‘context’ as the interrelated conditions in which the journey occurs such as departure-time, weather status, the purpose of the journey, companionship, etc. (see Sect. 3). By incorporating context and user preferences, more desirable journey plans can be recommended to the passengers which, by increasing their satisfaction, can motivate them to use multimodal transit.

As an example, suppose we are observing a user’s interactions with a particular web-based journey planning system. For instance, we observe that the passenger prefers a train over a bus arriving at *Flinders Street* for one query, and we also observe that for another query, the passenger prefers a train arriving at *Flinders Street* to a bus arriving at *Swanston Street* for a specific destination. An intuitively correct hypothesis that explains her behavior could be that she unconditionally prefers trains over buses, and *Flinders Street* over *Swanston Street*. Such a hypothesis is useful for further predictions. For example, using this hypothesis, we can predict that she will prefer a train to *Flinders Street* over anything else. However, such a hypothesis gives no further information about other preferences, for example, we cannot predict whether she prefers a bus arriving at *Flinders Street* over a train arriving at *Swanston Street* or not. Now assume that in the later observations, we observe that she also prefers a bus arriving at *Swanston Street* over a train arriving at *Swanston Street*. A new possible updated hypothesis could be that she prefers *Flinders Street* over *Swanston Street* when traveling by train and vice versa when traveling with buses. In other words, her preferences over the transportation modes are **conditioned** with her destined street.

In the above scenario, the passenger has used previous travel experiences to learn specific preferences about the journeys and a similar approach can be followed by a computer algorithm. The learning problem underlying this scenario is to extract a preference structure by observing the user’s behavior in situations involving a choice among several alternatives. Each alternative can be specified by many attributes, such as the transportation mode, the destination location, the arrival and departure time, etc. in the above example. As a result, the space of possible situations has a combinatorial structure. Furthermore, as we have shown in the example, the preferences induced by the passenger’s behavior are intrinsically related to conditional preferential independence, a fundamental notion in multi-attribute decision theory [20]. Indeed, the initial hypothesis is

unconditional in the sense that the preference over the values of each attribute is independent of the values of other attributes. By contrast, in the final hypothesis, the passenger's preferences among the transportation modes of the journeys are conditioned by the destined streets.

Conditional preference networks, also known as CP-nets, was proposed for handling problems where the preferences are conditioned to one another [4]. CP-nets have received a great deal of attention due to their compact and natural representation of conditional preferences [8, 12, 17]. Informally, a CP-net is a digraph where nodes represent attributes pointing to a (possibly empty) set of parents, and a set of conditional tables associated with each attribute, expressing the local preference on the values of the attribute given all possible combinations of values of its parents (Fig. 1) (see Sect. 2). The transitive closure of these local preferences is a partial order over the set of alternatives, which can be extended into several total orders. CP-nets and their generalizations are probably the most famous compact representation language for conditional preferences in multi-attribute domains [1]. While many facets of CP-nets have been studied in detail, such as learning of CP-nets, consistency and dominance checking, and optimization (constrained and unconstrained), to the best of our knowledge, there are no works on studying the effect of conditional preference modeling with CP-net in a real-world application. This paper aims to examine the effect of conditional preference modeling in the context-aware journey planning problem.

The objective of this paper is to investigate the effect of conditional preference modeling - using a GA-based CP-net learning methods (CPLGA) proposed in [8] - in personalized journey planning problem and compare it with various conventional preference learning techniques (four derived from the literature namely, RankNet citeburgess2005learning, AdaRank [18], OSVM [13] and SVOR [11], and one designed for the problem under investigation called learning preference weight (PWL) [9]) alongside with the performance comparison of three state-of-the-art passive CP-net learning methods presented in [8, 14, 15] for the personalized journey planning problem.

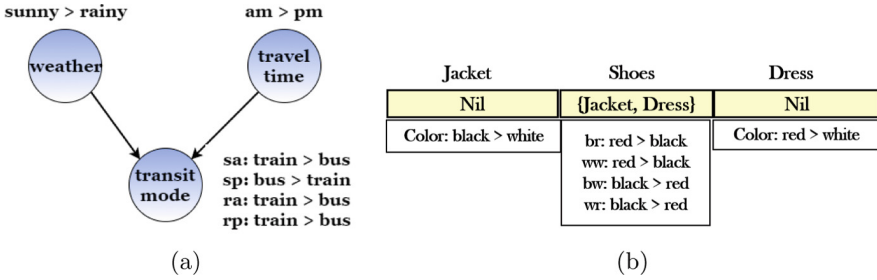
## 2 Background on CP-Net

Assume a finite list  $V = \{X_1, \dots, X_n\}$  of attributes, with their associated finite domains  $Dom = \{D_1, \dots, D_n\}$  where  $n$  is the number of domain elements. An attribute  $X_i$  is a binary attribute if  $D_i$  has two elements, which by convention we note  $x_i, \bar{x}_i$  [17]. By  $\Omega = \times_{X_i \in D} D_i$ , we denote the set of all complete alternatives, called *outcomes*.

A *preference relation* is a reflexive and transitive binary relation  $\succeq$  over  $\Omega$ . A *complete preference relation*  $\succeq$  is a preference relation that is connected, that is, for every  $x, y \in \Omega$  we have either  $x \succeq y$  or  $y \succeq x$ . A *strict preference relation*  $\succ$  is an irreflexive and transitive (thus asymmetric) binary relation over  $\Omega$ . A *linear preference relation* is a strict preference relation that is connected. From a preference relation we define a *strict preference relation* in the usual way:  $x \succ y$  iff  $x \succeq y$  and  $y \not\succeq x$ .



Preferences between outcomes that differ in the value of one attribute only, all other attributes being equal (or *ceteris paribus*) are often easy to assert and to understand. CP-nets [5] are a graphical language for representing such preferences. Informally, a CP-net is composed of a directed graph representing the preferential dependencies between attributes, and a set of conditional preference tables expressing, for each attribute, the local preference on the values of its domain given all possible combinations of values of its parents.



**Fig. 1.** (a) A simple CP-net  $N$ , modeling the passenger preferences. Journeys are defined by three attributes and for this particular passenger the preferences over transit mode is conditioned with the values of time of the journey and weather condition. (b) The equivalent chromosome of the sample CP-net

**Definition 1.** Preference: A strict preference relation  $\succ_u$  is a partial order on a set of outcomes  $O \in \Omega$  defined by a user  $u$ .  $o_i \succ_u o_j$  indicates that the user strictly prefers  $o_i$  over  $o_j$ .

**Definition 2.** Conditional Preference Rule (CP-rule): A CP-rule on an attribute  $X_i$  is an expression of the form  $t : p \succ \bar{p}$ , where  $p$  is a literal of  $X_i$  and  $t$  is a term such that  $t \in \{V \setminus X_i\}$ .

Such a rule means given that  $t$  holds, the value  $p$  is preferred to the value  $\bar{p}$  for the attribute  $X_i$ .

**Definition 3.** Conditional Preference Table (CPT):  $CPT(X_i)$  is a table associated with each attribute that consists of conditional preference rules (CP-rules)  $t : p \succ_i \bar{p}$  specifying a linear order on  $Dom(X_i)$  where  $t$  indicated to the parents of  $X_i$  in the dependency graph.

**Definition 4.** Conditional Preference Network (CP-net): A CP-net is a digraph on  $V = \{X_1, \dots, X_n\}$  in which each node is labeled with a CPT. An edge  $(X_i, X_j)$  indicates that the preferred value of  $X_j$  is conditioned by the value of its parent attribute  $X_i$ .

**Definition 5.** Dominance Testing: A dominance testing, defined by a triple  $(N, o_i, o_j)$ , is a decision of whether  $o_j$  is dominated by  $o_i$  given the CP-net  $N$  and  $o_i, o_j \in \Omega$ . The answer is in the affirmative if and only if  $N \models o_i \succ o_j$ .

Let us explain the properties of a CP-net with an example of the journey planning problem. Figure 1 represents a CP-net model for a particular passenger. Since the graph has three nodes we can infer that each journey is formulated by three attributes namely, weather condition, travel time and transit mode. Please note that one can describe journeys with a different set of attributes. As we can see in the Fig. 1, the CP-net contains three CPTs with six CP-rules (weather and travel time nodes has one rule each and four rules for transit mode node). Using this CP-net, as well as dominance testing, we can infer that the passenger prefers a train leaving in a morning on a sunny day to a bus leaving in the same condition. Formally speaking, a journey with a train dominates a journey with a bus for the traveler on a sunny morning. However, we still need to answer the question ‘*how can one model a passenger with a CP-net using her historical travel information?*’.

In GLPCA [8], we proposed a GA-based CP-nets learning solver in order to find a CP-net from historical and inconsistent preference examples. Each chromosome is representing a CP-net and the length of each chromosome is set to the number of attributes and is composed of two main parts:  $Parent_i$  and  $CPT_i$ .  $Parent_i$  denotes to the nodes  $j \in \{N \setminus i\}$  in the dependency graph which the preference over the value of node  $i$  is conditioned on them and  $CPT_i$  denotes the conditional preference table associated with node  $i$  (Fig. 1(b) represents the equivalent chromosome for the sample CP-net in Fig. 1(a)). Then, we used GA to find an individual that best describes the training preference dataset. The output of the algorithm is then considered as the user’s model and is used to predict her future ranking in order to provide personalized information. We refer readers to [8] for detailed information about the algorithm.

### 3 Multimodal Journey Planning Tool

In our study, we use the journey planner presented in [10] to find multimodal journey plans. This planner computes optimal multi-objective journey plans using a customized NSGAI-based algorithm [7]. Here we considered two criteria to optimize journey plans. The first criterion is the travel time and the second criterion is journey convenience which is a linear combination of the number of transfers, waiting and walking times. We refer the readers to [10] for detailed information about the algorithm.

#### 3.1 Journey Plan Attributes

To apply a CP-net, first, we need an attribute-based representation approach to describe each journey. Based on the knowledge of mobility experts, we divided the journey’s attributes into two categories: journey plan attributes and contextual attributes. Regarding journey plan attributes, we identify the following set of attributes to describe each journey:

**Travel Time:** which denotes to the total time spent to complete the journey.

**Modes of Transport:** which refers to the utilized transportation modes in the recommended journey.

**Personal Energy Expenditure (PEE):** which denotes to the PEE of the journeys that contain walking or cycling concerning the weight of the passenger as well as the average speed of the walking/cycling mode using the published energy consumption rates presented in [16].

**CO<sub>2</sub> Emission:** which denotes to the CO<sub>2</sub> emissions related to each journey. We utilized unit rates (per kilometer) for each vehicle to calculate the emission of a journey [ABS 2013].

**Number of Transfers:** which denotes to the number of transfers required to complete the journey.

**Monetary Cost:** which is the monetary cost associated with each journey [ABS 2013]. When a journey contains multiple public transport, the cost is calculated once in every 2-h time window.

Finally, CP-nets are typically designed to function with categorical data; therefore, we first have to discretize the numeric attributes described above. To do this, we employed a fuzzy-set method [12] that assigns each possible value to one or two predefined categories. In particular, we divide each numerical attribute into five equal intervals: very low, low, normal, high and very high. This method allows for a more accurate discretization by assigning a weight to the categories that are close to the boundaries separating two intervals.

### 3.2 Contextual Attributes

Based on the knowledge of mobility experts we identified seven contextual factors as relevant in this domain: 2 *user-specific* factors: companionship and reason of the journey, and five *environmental-based* factors namely: time of day, time of the week, weather, temperature, and crowdedness.

**Companionship:** which is a binary attribute indicating that the passenger is alone or not.

**Reason of the Journey:** which specifies the purpose of the journey including, *going to work*, *going back home* and *site seeing*.

**Time of Day:** which can be either *early morning*, *morning*, *afternoon*, *evening* and *night*.

**Time of the Week:** which is a binary value distinguishing between *weekends* and *week-days*.

**Weather:** which indicate the expected weather of a particular journey including, *sunny*, *rainy* and *windy*.

**Temperature:** which is a multivalued attribute consisting of *very cold*, *cold*, *normal*, *hot* and *very hot*.

**Crowdedness:** which denotes to the expected crowdedness of a particular public transit mode and can range from *quiet*, *natural* and *crowded*.

## 4 Algorithms' Evaluation

### 4.1 Experimental Setup

We have conducted experiments on real data collected from the transportation network of the City of Melbourne, to evaluate the effectiveness of the conditional preference modeling in the context-aware journey planning domain. For the road, bike and foot transportation network the OpenStreetMap<sup>1</sup> data has been used. Regarding public transit network, we used the GTFS<sup>2</sup> data, consisting of several information such as stop locations, routes, and timetable. A total of 34617 locations considered including 31259 bus stops, 1763 tram stations, 218 train stations, and 44 rental bike stations were included in the network. For the multi-modal network, all pairs of nodes, within 0.25 km radius, are connected by walking legs. Cycling legs are only available between two bike stations within the distance of two hours. The speed of walking and cycling legs is 5 km/h and 12 km/h respectively.

To carry out the experiment, we first had to collect a data-set of user ratings for a variety of journey plans. For each user, a set of 200 random queries, including random origin, destination and departure time, are created. By default, a set of contextual conditions was randomly picked for each query. In response to each query, the journey planner generated five to seven alternative journey plans combining different modes of transportation. Each plan was followed by a detailed explanation of characteristics of the journey plan and Users were asked to analyze and rank them from 'best' to 'worst' taking into consideration the 'active' contextual situation. This experiment lasted four weeks, and we collected a total of 5,218 orders given by 45 users to 31,350 journey plans in 8,710 queries. The participants comprised of 55% women and 45% men living in Melbourne (Australia) at the time of the experiment. Each user, on average, provided 115 rankings.

Besides, a common problem that arises when dealing with human subjects is the possibility of noise or inconsistent information [8]. Therefore, to test the robustness of the results, we also evaluated the behavior of preference learning methods under noisy conditions. To add order noise into the data set, we swapped the rankings of two randomly selected pairs of adjacent journeys in the original sample orders. The noise level could be controlled by changing the number of times that the swapping happens. Finally, We generated three data-set with 0.1%, 1% and 10% of noise, respectively.

Various types of distance metrics have been proposed in the literature to compute the distance between two orders,  $O_1$  and  $O_2$ , composed of the same sets of solutions, i.e.,  $X(O_1) = X(O_2)$ . In this paper, we use the widely-used Spearman's rank correlation coefficient ( $\rho$ ) [17], which is a non-parametric measure of correlation between two variables and is defined as:

<sup>1</sup> <http://www.openstreetmap.com>.

<sup>2</sup> The General Transit Feed Specification (GTFS) data which defines a common format for public transportation schedules and associated geographic information. For more information, please visit <http://www.transitwiki.org>.

$$\rho = 1 - \frac{6d_s(O_1, O_2)}{L^3 - L}, \quad (1)$$

where  $L$  is the length of orders and  $d_s(O_1, O_2)$  is the sum of the squared differences between ranks  $O_1$  and  $O_2$ .

Finally, in all the experiments, we used the CPLGA with the configuration setup described in Table 1. The parameters of CPLGA are set following our experience in practice. We have chosen a non-parametric test, Wilcoxon Signed Rank Test [6] as the statistical significant testing. The test is performed at the 5% significance level.

**Table 1.** CPLGA setup used in experiments

Selection mechanism	Ranked bias
	Bias = 1.2
Nr. of parents	Nr. of attributes
Cross-over rate	0.8
Mutation rate	0.4
Pool size	200
Maximum number of evaluation	20000
Results average over	30

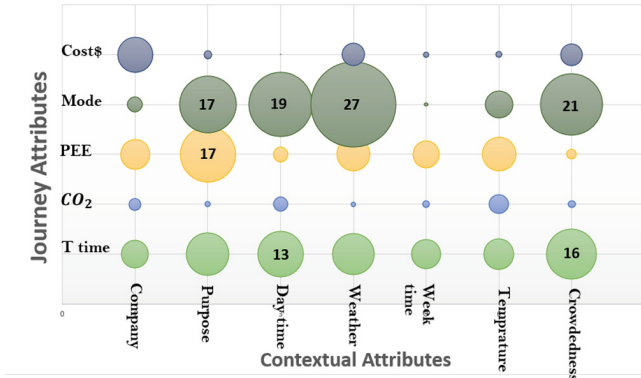
## 4.2 Result Analysis

Table 2 shows the means of  $\rho$  for the CP-net based preference learning algorithm with the learning-to-rank methods, namely RankNet [5], AdaRank [18], OSVM [13], SVOR [11] and PWL [9], different sample size and noise levels. These methods are the most popular methods for learning-to-rank in recent years and can perform reasonably well under noisy training samples. The experiment shows that CP-net based ranking significantly outperformed all the learning-to-rank methods at different noise levels and different training sizes. This is due to the fact that learning-to-rank methods do not take into account the conditional dependency of the attributes. However, our further experiments reveal that there exists a dependency between passengers' preferences that the conventional learning-to-rank methods tend to overlook.

As discussed earlier, the purpose of CP-net is to provide a conditional model to represent the user preferences. Therefore, during the experiments, we modeled each user with a CP-net based on his/her rating data-set, i.e., a total number of 45 CP-nets were obtained. Figure 2 illustrates the dependencies between journey attributes and contextual attributes among all 45 learned CP-nets. The number in a circle represents the number of CP-nets that the two attributes were conditioned to each other. For example, we observed that for 27 passengers, the value of transportation mode was dependent on the expected weather condition of the journey. In other words, for 27 passengers, the learned model indicates

**Table 2.** Comparing the conditional preference learning with conventional learning to rank methods for different training sizes.

$ S $	200				500				1000			
Noise level	0	0.01	0.05	0.1	0	0.01	0.05	0.1	0	0.01	0.05	0.1
Method	$\rho$											
AdaRank	0.7453	0.7458	0.7352	0.7140	0.7642	0.7799	0.7614	0.7397	0.7895	0.7917	0.7743	0.7527
RankNet	0.6663	0.6642	0.6438	0.6242	0.7031	0.6833	0.6768	0.6493	0.7376	0.7165	0.7046	0.6765
OSVM	0.7305	0.7123	0.6653	0.6276	0.7866	0.7866	0.7622	0.7201	0.6383	0.8257	0.7881	0.6476
SVOR	0.7360	0.6965	0.6569	0.6363	0.7718	0.7704	0.6754	0.6271	0.8063	0.7704	0.6883	0.6435
PWL	0.7260	0.7246	0.7149	0.7002	0.7864	0.7751	0.7592	0.7520	0.8119	0.8031	0.7808	0.7717
CPLGA	<b>0.8435</b>	<b>0.8432</b>	<b>0.8215</b>	<b>0.8090</b>	<b>0.8817</b>	<b>0.8769</b>	<b>0.8530</b>	<b>0.8433</b>	<b>0.9285</b>	<b>0.9019</b>	<b>0.8946</b>	<b>0.8775</b>

**Fig. 2.** The conditional dependency between contextual and journey attributes. The numbers in the circles denote the number of CP-nets that the values of two attributes are dependent on each other.

that their preferences among the transportation modes used in the journey are conditioned on the weather status. We also observed that almost half of the participants have a conditioned preference over the transportation modes based on the expected crowdedness of the transportation network. Latent information such as this, which is ignored by the majority of popular learning-to-rank methods, can be precious when one wants to predict the passengers' behavior. We believe that this information was the main reason of why CP-net based preference learning method outperformed all conventional ones. However, we first need to prove that the learned CP-net are concordant with the actual passengers' behavior. To achieve this, we conduct another experiment to reveal that whether the actual behavior of passengers matched with our learned CP-nets. For the sake of brevity, in this paper, we only present the two highest conditioned attributes, namely (weather and mode) and (crowdedness and mode).

Figure 3 presents the average percentage of transportation mode against four different attributes namely, crowdedness, day-time, purpose and weather condition. In Fig. 3(a) we show the average percentage of transportation modes for



(a) Weather (27 passengers from Fig. 2). (b) Crowdedness (21 passengers from Fig. 2)



(c) Day-time (19 passengers from Fig. 2) (d) Purpose (17 passengers from Fig. 2)

**Fig. 3.** The conditional dependency between transit mode and four highly conditioned variables extracted from the true ratings of the passengers.

the first ranked journey for the 27 passengers presented in Fig. 2 based on the learned CP-nets for these passengers, we expected that the transportation mode was conditioned to the weather status. Figure 3(a) shows the actual behavior of these passengers when they rated the actual recommended journeys. In here we assumed that, for each query, they would choose their highest ranked journey. As shown in Fig. 3(a), there is a clear correlation between the used transportation mode and the weather status which demonstrates that the learned CP-net is concordant with the actual behavior of the passengers. For example, we observed that when raining, the usage of trains was increased as these passengers preferred trains more over other means of transportation. We also observed that the usage of buses dropped dramatically in raining condition. It could be since, for buses and trams, the possibility of delays increases in raining weather and passengers – who gained this knowledge through experience – try to avoid it by leaning towards trains which are more robust against variations in weather conditions. Although, this information may seem trivial, but note that these explicit dependencies are being ignored by the conventional learning-to-rank methods. Needless to say, such information is beneficial when the system wants to predict passengers’ preferences to recommend personalized journeys to them.

Figure 3(b) demonstrates the same results for the relation between expected crowdedness of the transportation network and the passengers’ preferences among different modes of transportation. As we stated before, in 21 out of 47 learned CP-net, the value of transportation mode is conditioned with the value of expected crowdedness. Again, we observed that there is a clear correlation between the two attributes in actual passengers’ behaviors. For example, we observed an increase in train usage in crowded situations. One explanation could be that the passengers prefer to avoid traffic jams, in case of buses, or limited space, in case of trams. We also observed an increase of bicycle usage when the transportation network is crowded. This could be because, in the City

of Melbourne passengers are only allowed to bring their bikes onto the trains, but prohibited for other means of transportation, so some passengers are willing to take some part of the journey with bikes during rush hours.

To have a fair comparison, we also compare CPLGA [8] against two CP-net learning algorithms proposed in [14, 15]. Similar to CPLGA, these methods learn CP-nets passively from inconsistent examples. We observed that CPLGA algorithm significantly performs better than the other two. Regarding [15] it is because this algorithm starts with a hypothesis and then performs a local search to optimize that hypothesis, making the algorithm prone to getting stuck in local optima for larger problems. Another issue is the sample size. Note that for larger problems (i.e., more than ten attributes) these algorithms need a large training set to prove their hypothesis. We also tested the robustness of these methods in noisy condition by adding 1% to 20% of noise to the data-set. We observed that all methods which have handled the noisy data and could find similar preference graphs as the noise-free setting; however, we again observed a significant gap between CPLGA model and the other algorithms concerning their performance (Table 3).

**Table 3.** Comparison between the three state-of-the-art passive CP-net learning methods on real data with different noise level.

S	$\rho$	0	0.01	0.05	0.1	0.2
	Method	Sample agreement				
500	[15]	0.5533	0.5564	0.4756	0.4213	0.2712
	[14]	0.5117	0.5107	0.4819	0.4665	0.2301
	CPLGA	<b>0.9212</b>	<b>0.9230</b>	<b>0.9195</b>	<b>0.8400</b>	<b>0.7512</b>
1000	[15]	0.5812	0.5601	0.5139	0.4201	0.3320
	[14]	0.5210	0.5109	0.4939	0.4339	0.3134
	CPLGA	<b>0.9309</b>	<b>0.9101</b>	<b>0.9152</b>	<b>0.8754</b>	<b>0.7713</b>

## 5 Conclusions

In this paper, we discussed the effect of conditional preference learning in the domain of context-aware journey planning problem. To this aim, we have proposed a context-aware journey recommendation test-bed and we have implemented and evaluated the CP-net based preference learning algorithm and compared it with five state-of-the-art PL strategies and two similar CP-net learning approaches. Our experiment results have concluded that there exists the latent conditional information in the user preferences and this information can be very useful when one wants to predict the passengers' behavior in the urban transportation network.

Our future work is to further improve the performance of the conditional preference learning methods. We also want to investigate the effectiveness of the



conditional preference learning strategies when applied during the construction of the journey plans. We believe that in this way the preference model can have a major impact on quality of the recommended journeys and also help to speed up the plan generation process by reduction of the search space.

**Acknowledgment.** This research was supported under Australian Research Council's Linkage Projects funding scheme (project number LP120200305).

## References

1. Allen, T.E.: CP-nets: from theory to practice. In: Walsh, T. (ed.) ADT 2015. LNCS, vol. 9346, pp. 555–560. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-23114-3\\_33](https://doi.org/10.1007/978-3-319-23114-3_33)
2. Bell, P., Knowles, N., Everson, P.: Measuring the quality of public transport journey planning. In: IET and ITS Conference on Road Transport Information and Control, RTIC 2012, pp. 1–4. IET (2012)
3. Bonsall, P.: Do we know whether personal travel planning really works? *Transp. Policy* **16**(6), 306–314 (2009)
4. Boutilier, C., Brafman, R.I., Hoos, H.H., Poole, D.: Reasoning with conditional ceteris paribus preference statements. In: UAI, pp. 71–80 (1999)
5. Burges, C., et al.: Learning to rank using gradient descent. In: ICML, pp. 89–96 (2005)
6. Corder, G.W., Foreman, D.I.: *Nonparametric Statistics: A Step-by-Step Approach*. Wiley, Hoboken (2014)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.A.M.T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
8. Haqqani, M., Li, X.: An evolutionary approach for learning conditional preference networks from inconsistent examples. In: Cong, G., Peng, W.-C., Zhang, W.E., Li, C., Sun, A. (eds.) ADMA 2017. LNCS, vol. 10604, pp. 502–515. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-69179-4\\_35](https://doi.org/10.1007/978-3-319-69179-4_35)
9. Haqqani, M., Li, X., Yu, X.: Estimating passenger preferences using implicit relevance feedback for personalized journey planning. In: Wagner, M., Li, X., Hendtlass, T. (eds.) ACALCI 2017. LNCS, vol. 10142, pp. 157–168. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-51691-2\\_14](https://doi.org/10.1007/978-3-319-51691-2_14)
10. Haqqani, M., Li, X., Yu, X.: An evolutionary multi-criteria journey planning algorithm for multimodal transportation networks. In: Wagner, M., Li, X., Hendtlass, T. (eds.) ACALCI 2017. LNCS, vol. 10142, pp. 144–156. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-51691-2\\_13](https://doi.org/10.1007/978-3-319-51691-2_13)
11. Herbrich, R., Graepel, T., Obermayer, K.: Support vector learning for ordinal regression. In: ICANN, vol. 1, pp. 97–102 (1999)
12. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst. (TOIS)* **22**(1), 5–53 (2004)
13. Kazawa, H., Hirao, T., Maeda, E.: Order SVM: a kernel method for order learning based on generalized order statistics. *Syst. Comput. Jpn.* **36**(1), 35–43 (2005)
14. Liu, J., Xiong, Y., Caihua, W., Yao, Z., Liu, W.: Learning conditional preference networks from inconsistent examples. *IEEE TKDE* **26**(2), 376–390 (2014)
15. Liu, J., Yao, Z., Xiong, Y., Liu, W., Caihua, W.: Learning conditional preference network from noisy samples using hypothesis testing. *Knowl.-Based Syst.* **40**, 7–16 (2013)

16. Owen, N., Humpel, N., Leslie, E., Bauman, A., Sallis, J.F.: Understanding environmental influences on walking. *Am. J. Prev. Med.* **27**(1), 67–76 (2004)
17. Spearman, C.: The proof and measurement of association between two things. *Am. J. Psychol.* **15**(1), 72–101 (1904)
18. Xu, J., Li, H.: AdaRank: a boosting algorithm for information retrieval. In: *ACM SIGIR*, pp. 391–398 (2007)



# Critical Fractile Optimization Method Using Truncated Halton Sequence with Application to SAW Filter Design

Kiyoharu Tagawa<sup>(✉)</sup>

Kindai University, Higashi-Osaka 577-8502, Japan  
tagawa@info.kindai.ac.jp

**Abstract.** This paper proposes an efficient optimization method to solve the Chance Constrained Problem (CCP) described as the critical fractile formula. To approximate the Cumulative Distribution Function (CDF) in CCP with an improved empirical CDF, the truncated Halton sequence is proposed. A sample saving technique is also contrived to solve CCP by using Differential Evolution efficiently. The proposed method is applied to a practical engineering problem, namely the design of SAW filter.

**Keywords:** Chance Constrained Problem · Empirical distribution

## 1 Introduction

In real-world optimization problems, various uncertainties have to be taken into account. Traditionally, there are two kinds of problem formulations for handling uncertainties in the optimization [11], namely the deterministic one and the stochastic one. Chance Constrained Problem (CCP) [13] is one of the possible formulation of the stochastic optimization problem. Since the balance between optimality and reliability can be taken with a probability in CCP, a number of real-world optimization problems have been formulated as CCPs [7, 9].

CCP has been studied in the field of stochastic programming [13]. If the chance constraint is linear, CCP can be transformed to a deterministic optimization problem. Otherwise, CCP is so hard to solve because the time-consuming Monte Carlo simulation is needed to calculate the empirical probability that the chance constraint is satisfied. For solving CCP with the optimization methods of nonlinear programming, the stochastic programming assumes that the chance constraint is differentiable and convex. Even though Evolutionary Algorithms (EAs) are also reported to solve CCP [8, 12], they use Monte Carlo simulations to evaluate the feasibility of every solution in the process of optimization.

In our previous paper [16], an optimization method based on Differential Evolution (DE) [14] was given to solve CCP without the Monte Carlo simulation. Specifically, CCP is described by using the Cumulative Distribution Function (CDF) of uncertain function value. In order to approximate CDF from samples,

an extended version of Empirical CDF (ECDF) [10], which is called Weighted ECDF (W\_ECDF) [15], was employed. Thereby, for solving the CCP formulated with CDF, an Adaptive DE (ADE) combined with W\_ECDF was used.

This paper focuses on a specific CCP known as the critical fractile formula [4] and improves the previous method [16] by introducing two new techniques. Firstly, the truncated Halton sequence is proposed to approximate CDF with W\_ECDF more efficiently. Secondly, a new ADE equipped with a sample saving technique is proposed. The improved method is applied to the structural design of Surface Acoustic Wave (SAW) filters [2], which are widely used in the radio frequency circuits of mobile communication systems such as cellular phones.

## 2 Background and Problem Formulation

As stated above, there are two problem formulations for handling uncertainties. Robust optimization problem is a deterministic problem formulation [3]. Let  $\mathbf{x} = (x_1, \dots, x_D) \in \mathbf{X} \subseteq \mathbb{R}^D$ ,  $\mathbf{X} = [\underline{x}_j, \bar{x}_j]^D$ ,  $j = 1, \dots, D$  be a vector of decision variables, or a solution. The uncertainty is given by a vector of random variables  $\boldsymbol{\xi} = (\xi_1, \dots, \xi_K) \in \boldsymbol{\Xi}$  with a support  $\boldsymbol{\Xi} \subseteq \mathbb{R}^K$ . Robust optimization problem is defined with a measurable function  $g : \mathbf{X} \times \boldsymbol{\Xi} \rightarrow \mathbb{R}$  as

$$\min_{\mathbf{x} \in \mathbf{X}} \gamma \quad \text{s.t.} \quad \forall \boldsymbol{\xi} \in \boldsymbol{\Xi} : g(\mathbf{x}, \boldsymbol{\xi}) \leq \gamma. \tag{1}$$

The feasible solution  $\mathbf{x} \in \mathbf{X}$  of the robust optimization problem in (1) has to satisfy the constraint  $g(\mathbf{x}, \boldsymbol{\xi}) \leq \gamma$  absolutely with 100% probability. Therefore, it seems to be too conservative from an engineering perspective.

CCP is a stochastic problem formulation [13]. By introducing any required sufficiency level  $\alpha \in (0, 1)$  into an infinite number of constraints in (1), CCP reduces the conservatism of the robust optimization problem as

$$\min_{\mathbf{x} \in \mathbf{X}} \gamma \quad \text{s.t.} \quad \Pr(g(\mathbf{x}, \boldsymbol{\xi}) \leq \gamma) \geq \alpha \tag{2}$$

where  $\Pr(A)$  denotes the probability that an event  $A$  will occur.

Actually, CCP may have more than one constraint. Besides, there are two types of CCPs, namely separate CCP and joint CCP [13]. In this paper, separate CCP having only one chance constraint is considered as shown in (2).

The presence of the uncertainty in CCP leads to different results for repeated evaluations of the same solution  $\mathbf{x} \in \mathbf{X}$ . Since  $\boldsymbol{\xi} \in \boldsymbol{\Xi}$  is a vector of random variables, the function value  $g(\mathbf{x}, \boldsymbol{\xi}) \in \mathbb{R}$  in (2) becomes a random variable too. The CDF of  $g(\mathbf{x}, \boldsymbol{\xi})$  depending on the solution  $\mathbf{x} \in \mathbf{X}$  is defined as

$$F(\mathbf{x}, \gamma) = \Pr(g(\mathbf{x}, \boldsymbol{\xi}) \leq \gamma). \tag{3}$$

By using the inverse CDF of  $g(\mathbf{x}, \boldsymbol{\xi})$ , an alternative formulation of the CCP in (2), which is known as the critical fractile formula [4], is written as

$$\min_{\mathbf{x} \in \mathbf{X}} \gamma(\mathbf{x}) = F^{-1}(\mathbf{x}, \alpha) \tag{4}$$

where  $\gamma(\mathbf{x})$  denotes the critical fractile  $\gamma = \gamma(\mathbf{x})$  achieved by  $\mathbf{x} \in \mathbf{X}$ .

The probability distribution of  $\boldsymbol{\xi} \in \boldsymbol{\Xi}$  in CCP is usually known [13]. If the probability distribution of  $g(\mathbf{x}, \boldsymbol{\xi}) \in \mathfrak{R}$  is also known or the inverse CDF of  $g(\mathbf{x}, \boldsymbol{\xi})$  can be derived analytically, the CCP in (4) can be transformed into a deterministic optimization problem [4, 13]. Otherwise, for solving the original CCP in (2), the probability  $\Pr(g(\mathbf{x}, \boldsymbol{\xi}) \leq \gamma)$  in (2) has to be evaluated repeatedly with the Monte Carlo simulation by changing the value of  $\gamma \in \mathfrak{R}$ .

### 3 Approximation of CDF

#### 3.1 Empirical CDF (ECDF)

In real-world optimization problems,  $g(\mathbf{x}, \boldsymbol{\xi})$  in (3) is too complex to derive its CDF analytically. Therefore, an approximation of the CDF is composed from samples. Let  $g(\mathbf{x}, \boldsymbol{\xi}^n) \in \mathfrak{R}$ ,  $\boldsymbol{\xi}^n \in \boldsymbol{\Xi}$ ,  $n = 1, \dots, N$  be a set of random samples of the function value  $g(\mathbf{x}, \boldsymbol{\xi})$  in (3). The indicator function is defined as

$$\mathbb{1}(g(\mathbf{x}, \boldsymbol{\xi}^n) \leq \gamma) = \begin{cases} 1 & \text{if } g(\mathbf{x}, \boldsymbol{\xi}^n) \leq \gamma \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

From the samples  $g(\mathbf{x}, \boldsymbol{\xi}^n)$ ,  $n = 1, \dots, N$ , ECDF [10] is composed as

$$\mathbb{F}(\mathbf{x}, \gamma) = \frac{1}{N} \sum_{n=1}^N \mathbb{1}(g(\mathbf{x}, \boldsymbol{\xi}^n) \leq \gamma). \tag{6}$$

Let  $\tilde{\mathbb{F}}(\mathbf{x}, \gamma)$  be a smoothed ECDF. The CDF of  $g(\mathbf{x}, \boldsymbol{\xi})$  is approximated by  $\tilde{\mathbb{F}}(\mathbf{x}, \gamma)$ . Since  $\tilde{\mathbb{F}}(\mathbf{x}, \gamma)$  is a monotone increasing function, we can get the inverse CDF value, or the critical fractile in (4), numerically as  $\gamma = \tilde{\mathbb{F}}^{-1}(\mathbf{x}, \alpha)$ .

As a drawback of ECDF, many samples are required to approximate CDF accurately because the samples  $g(\mathbf{x}, \boldsymbol{\xi}^n)$ ,  $\boldsymbol{\xi}^n \in \boldsymbol{\Xi}$  taken from the tail part of the probability distribution on  $\boldsymbol{\Xi} \subseteq \mathfrak{R}^K$  are relatively few in number.

#### 3.2 Weighted Empirical CDF (W\_ECDF)

W\_ECDF [15] is an improved ECDF to approximate CDF in (3). In order to take samples  $\boldsymbol{\xi}^n \in \boldsymbol{\Xi}$  from  $\boldsymbol{\Xi} \subseteq \mathfrak{R}^K$  uniformly,  $K$ -dimensional Halton Sequence (HS) is used instead of the random sampling. HS is a low-discrepancy sequence [5]. Let  $\boldsymbol{\theta}^n \in \boldsymbol{\Theta} \subseteq \mathfrak{R}^K$ ,  $n = 1, \dots, N$  be a set of points generated as HS. Considering the support  $\boldsymbol{\Xi} \subseteq \mathfrak{R}^K$ , the region  $\boldsymbol{\Theta} \subseteq \mathfrak{R}^K$  of HS is chosen as  $\boldsymbol{\Theta} \supseteq \boldsymbol{\Xi}$ .

Let  $f : \boldsymbol{\Xi} \rightarrow [0, \infty)$  be the Probability Density Function (PDF) of  $\boldsymbol{\xi} \in \boldsymbol{\Xi}$ . Each of the points  $\boldsymbol{\theta}^n \in \boldsymbol{\Theta}$  of HS is weighted by the PDF of  $\boldsymbol{\xi} \in \boldsymbol{\Xi}$  as  $f(\boldsymbol{\theta}^n)$ . Thereby, W\_ECDF is composed from  $g(\mathbf{x}, \boldsymbol{\theta}^n)$ ,  $\boldsymbol{\theta}^n \in \boldsymbol{\Theta}$ ,  $n = 1, \dots, N$  as

$$\mathbb{F}(\mathbf{x}, \gamma) = \frac{1}{W} \sum_{n=1}^N f(\boldsymbol{\theta}^n) \mathbb{1}(g(\mathbf{x}, \boldsymbol{\theta}^n) \leq \gamma) \tag{7}$$

where  $W = f(\boldsymbol{\theta}^1) + \dots + f(\boldsymbol{\theta}^n) + \dots + f(\boldsymbol{\theta}^N)$ .

By using a smoothed W\_ECDF  $\tilde{\mathbb{F}}(\mathbf{x}, \gamma)$ , we can obtain  $\gamma = \tilde{\mathbb{F}}^{-1}(\mathbf{x}, \gamma)$ .

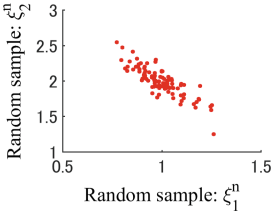


Fig. 1. RS:  $\xi^n \in \Xi$

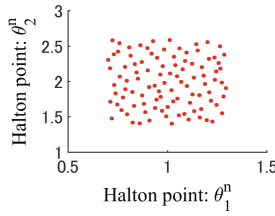


Fig. 2. HS:  $\theta^n \in \Theta$

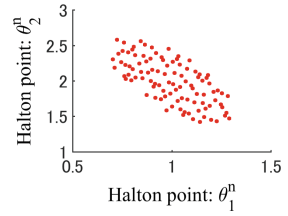


Fig. 3. THS:  $\theta^n \in \mathcal{S}$

### 3.3 Truncated Halton Sequence (THS)

In our previous paper [16], we supposed that all of the random variables  $\xi_j \in \mathfrak{R}$ ,  $j = 1, \dots, K$  are mutually independent. Besides, for composing W\_ECDF in (7) from  $\theta^n \in \Theta$ , the region  $\Theta \subseteq \mathfrak{R}^K$  of HS was given by a hyper-cube.

In this paper, Truncated HS (THS) is proposed to compose W\_ECDF more efficiently. The region  $\mathcal{S} \subseteq \Theta$  of THS is defined with  $\Theta \subseteq \mathfrak{R}^K$  as

$$\mathcal{S} = \{\theta^n \in \Theta \mid f(\theta^n) \geq f_{\min}\} \tag{8}$$

where the minimum PDF value  $f_{\min}$  is a parameter given in advance.

By using the points  $\theta^n \in \mathcal{S}$ ,  $n = 1, \dots, N$  of THS for composing W\_ECDF, we can eliminate futile points  $\theta^n \in \Theta$  such as  $f(\theta^n) \approx 0$ . The correlation between two random variables  $\xi_i$  and  $\xi_j$ ,  $i \neq j$  is also reflected in  $\theta^n \in \mathcal{S}$  naturally.

**Example of W\_ECDF with THS.** Let's consider a stochastic function:

$$g(\mathbf{x}, \boldsymbol{\xi}) = \mathbf{x} \boldsymbol{\xi}^T = x_1 \xi_1 + x_2 \xi_2 \tag{9}$$

where  $\boldsymbol{\xi} \in \Xi \subseteq \mathfrak{R}^2$  is following a 2-dimensional normal distribution such as

$$\boldsymbol{\xi} = (\xi_1, \xi_2) \sim \mathcal{N}_2(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho) = \mathcal{N}_2(1, 2, 0.1^2, 0.2^2, -0.8) \tag{10}$$

where  $\rho$  denotes the correlation coefficient between  $\xi_1$  and  $\xi_2$ .

Figure 1 shows  $\xi^n \in \Xi$ ,  $N = 100$  generated by the Random Sampling (RS) of  $\boldsymbol{\xi} \in \Xi$  in (10). Figure 2 shows  $\theta^n \in \Theta$ ,  $N = 100$ . Figure 3 shows  $\theta^n \in \mathcal{S}$ ,  $N = 100$ . Since HS [5] is deterministic, the randomized HS [19] is used in this paper.

From the theory of probability [1], the PDF of  $\boldsymbol{\xi} \in \Xi$  in (10) is

$$f(\boldsymbol{\xi}) = \frac{1}{2\pi \sqrt{|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{\xi} - \boldsymbol{\mu}) \boldsymbol{\Sigma}^{-1}(\boldsymbol{\xi} - \boldsymbol{\mu})^T\right) \tag{11}$$

where  $\boldsymbol{\mu} = (\mu_1, \mu_2)$  and the covariance matrix  $\boldsymbol{\Sigma}$  is given as

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & \sigma_1 \sigma_2 \rho \\ \sigma_1 \sigma_2 \rho & \sigma_2^2 \end{pmatrix}. \tag{12}$$

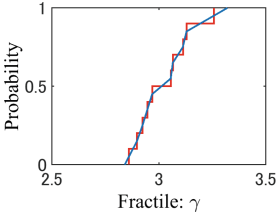


Fig. 4. ECDF in (6)

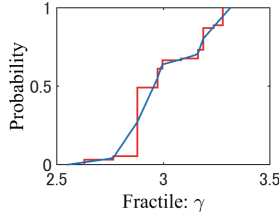


Fig. 5. W\_ECDF in (7)

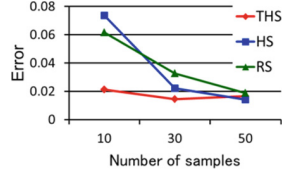


Fig. 6. Estimation error

From the linearity of the normal distribution, the value of  $g(\mathbf{x}, \boldsymbol{\xi})$  in (9) also follows a normal distribution with mean  $\mu_g(\mathbf{x})$  and variance  $\sigma_g^2(\mathbf{x})$  as

$$g(\mathbf{x}, \boldsymbol{\xi}) \sim \mathcal{N}(\mu_g(\mathbf{x}), \sigma_g^2(\mathbf{x})) = \mathcal{N}(\mathbf{x} \boldsymbol{\mu}^T, \mathbf{x} \boldsymbol{\Sigma} \mathbf{x}^T). \tag{13}$$

From (13), the CDF of  $g(\mathbf{x}, \boldsymbol{\xi})$  in (9) can be derived exactly as

$$F(\mathbf{x}, \gamma) = \Pr \left( \frac{g(\mathbf{x}, \boldsymbol{\xi}) - \mu_g(\mathbf{x})}{\sigma_g(\mathbf{x})} \leq \frac{\gamma - \mu_g(\mathbf{x})}{\sigma_g(\mathbf{x})} \right) = \Phi \left( \frac{\gamma - \mu_g(\mathbf{x})}{\sigma_g(\mathbf{x})} \right) \tag{14}$$

where  $\Phi$  denotes the CDF of the standard normal distribution [1].

ECDF and W\_ECDF are used to approximate  $F(\hat{\mathbf{x}}, \gamma)$  in (14) for a solution  $\hat{\mathbf{x}} = (1, 1)$ . Figure 4 shows an example of the step function of ECDF and its smoothed one. ECDF is composed from  $N = 10$  samples  $g(\hat{\mathbf{x}}, \boldsymbol{\xi}^n)$ ,  $\boldsymbol{\xi}^n \in \boldsymbol{\Xi}$ . Similarly, Fig. 5 shows W\_ECDF and its smoothed one composed from  $N = 10$  samples  $g(\hat{\mathbf{x}}, \boldsymbol{\theta}^n)$ ,  $\boldsymbol{\theta}^n \in \boldsymbol{S}$ . From Figs. 4 and 5, the samples  $g(\hat{\mathbf{x}}, \boldsymbol{\theta}^n)$  for W\_ECDF are distributed wider than the samples  $g(\hat{\mathbf{x}}, \boldsymbol{\xi}^n)$  for ECDF.

From (14), the critical fractile  $\hat{\gamma} = F^{-1}(\hat{\mathbf{x}}, \alpha) \approx 3.17$  is obtained exactly for  $\alpha = 0.9$ . Figure 6 compares between THS, HS, and RS in the estimation error  $|\tilde{\mathbb{F}}^{-1}(\hat{\mathbf{x}}, \alpha) - \hat{\gamma}|$  averaged over 10 runs. For generating  $\boldsymbol{\theta}^n \in \boldsymbol{S}$  from  $\boldsymbol{\theta}^n \in \boldsymbol{\Theta}$ ,  $f_{\min} = 0.01$  is used in (8) and about 40% of  $\boldsymbol{\theta}^n \in \boldsymbol{\Theta}$  are dumped. From Fig. 6, the estimation error with THS is small even if the sample size  $N$  is small.

## 4 Critical Fractile Optimization Method

### 4.1 Differential Evolution with Sample Saving Technique

By using the smoothed W\_ECDF composed of  $N$  samples and a correction level  $\beta \geq \alpha$ , the CCP in (4), namely the critical fractile formula, is written as

$$\min_{\mathbf{x} \in \mathbf{X}} \gamma(\mathbf{x}) = \tilde{\mathbb{F}}^{-1}(\mathbf{x}, \beta) \tag{15}$$

where the correction level is initialized as  $\beta := \alpha$  and regulated in the procedure of the proposed optimization method as noted below if it is necessary.

The original versions of many EAs including DE have been developed to solve unconstrained optimization problems. Therefore, they can be applied directly to

the CCP in (15). In this paper, one of the most successful ADE, namely JADE without archive [20], is used. As well as DE, JADE has a set of solutions  $\mathbf{x}_i \in \mathbf{P}_t$ ,  $i = 1, \dots, N_P$  called population. An initial population  $\mathbf{P}_0 \subseteq \mathbf{X}$  is generated randomly. Then every solution  $\mathbf{x}_i \in \mathbf{P}_0$  is evaluated  $N$  times and the objective function  $\gamma(\mathbf{x}_i)$  in (15) is estimated from  $g(\mathbf{x}_i, \boldsymbol{\theta}^n)$ ,  $\boldsymbol{\theta}^n \in \mathbf{S}$ ,  $n = 1, \dots, N$  as stated above. At each generation  $t$ ,  $\mathbf{x}_i \in \mathbf{P}_t$ ,  $i = 1, \dots, N_P$  is assigned to a parent in turn. By using the strategy named “DE/current-to- $p$ best/1/bin” [20], a child  $\mathbf{u}_i \in \mathbf{X}$  is generated from the parent  $\mathbf{x}_i \in \mathbf{P}_t$  and evaluated  $N$  times. If  $\gamma(\mathbf{u}_i) \leq \gamma(\mathbf{x}_i)$  holds, the parent  $\mathbf{x}_i \in \mathbf{P}_t$  is replaced by the child  $\mathbf{u}_i \in \mathbf{X}$ .

JADE applied to a real-world optimization problem spends most of time to evaluate children. The proposed sample saving technique called “pretest” can find and eliminate fruitless children with a few samples. When a newborn child  $\mathbf{u}_i \in \mathbf{X}$  is compared with its parent  $\mathbf{x}_i \in \mathbf{P}_t$ , the pretest takes its samples  $g(\mathbf{u}_i, \boldsymbol{\theta}^n)$  one by one. Let  $m \leq N$  be the number of samples obtained so far. From these samples, the empirical probability is calculated with weights as

$$\widehat{\Pr}(\gamma(\mathbf{u}_i) > \gamma(\mathbf{x}_i)) = \frac{1}{W} \sum_{n=1}^m f(\boldsymbol{\theta}^n) \mathbb{1}(g(\mathbf{u}_i, \boldsymbol{\theta}^n) > \gamma(\mathbf{x}_i)) \tag{16}$$

where  $\widehat{\Pr}(A)$  denotes the predicted value of  $\Pr(A)$  through observations.

If  $\widehat{\Pr}(\gamma(\mathbf{u}_i) > \gamma(\mathbf{x}_i)) > 2(1 - \beta)$  holds on the way,  $\mathbf{u}_i \in \mathbf{X}$  is regarded as worse than  $\mathbf{x}_i \in \mathbf{P}_t$  and discarded without evaluating  $\gamma(\mathbf{u}_i) = \tilde{\mathbb{F}}^{-1}(\mathbf{u}_i, \beta)$ .

JADE combined with Pretest is named JADEP. In the global optimization process of JADEP, the pretest is used locally in the competition between parent and child. Therefore, the pretest doesn’t degrade the performance of JADE.

### 4.2 Verification of Solution Using Monte Carlo Simulation

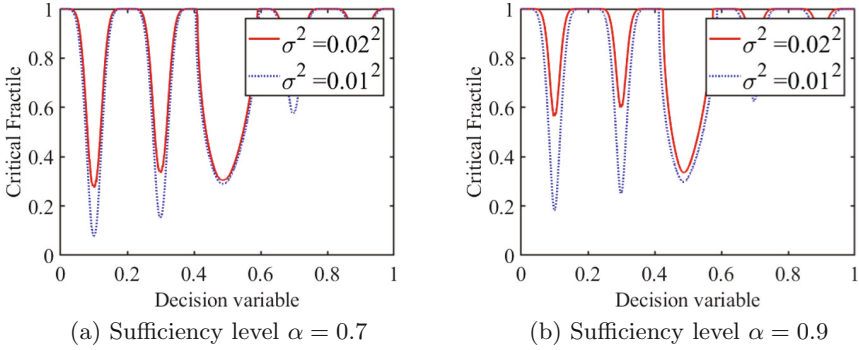
We verify the feasibility of the solution  $\mathbf{x}_b \in \mathbf{X}$  obtained by JADEP for the CCP in (15). Specifically, by using a huge number of random samples  $g(\mathbf{x}_b, \boldsymbol{\xi}^n)$ ,  $\boldsymbol{\xi}^n \in \boldsymbol{\Xi}$ ,  $n = 1, \dots, \hat{N}$ , we calculate the empirical probability that the chance constraint of the CCP in (2) is satisfied with the solution  $\mathbf{x}_b \in \mathbf{X}$  as

$$\widehat{\Pr}(g(\mathbf{x}_b, \boldsymbol{\xi}) \leq \gamma(\mathbf{x}_b)) = \frac{1}{\hat{N}} \sum_{n=1}^{\hat{N}} \mathbb{1}(g(\mathbf{x}_b, \boldsymbol{\xi}^n) \leq \gamma(\mathbf{x}_b)). \tag{17}$$

If  $\widehat{\Pr}(g(\mathbf{x}_b, \boldsymbol{\xi}) \leq \gamma(\mathbf{x}_b)) \geq \alpha$  holds, we regard that  $\mathbf{x}_b \in \mathbf{X}$  is a feasible solution of the CCP in (2). Otherwise, we increase the value of the correction level  $\beta$  just a little and apply JADEP to the CCP in (15) again.

The sample size  $\hat{N}$  in (17) is determined as follows. Let  $\mathbf{x}^* \in \mathbf{X}$  be the optimum solution of the CCP in (2) and  $y_n = \mathbb{1}(g(\mathbf{x}^*, \boldsymbol{\xi}^n) \leq \gamma)$ . Therefore,  $\Pr(y_n = 1) = \alpha$  and  $\Pr(y_n = 0) = (1 - \alpha)$  hold. Let  $\hat{y}$  be the sample mean of  $y_n$ ,  $n = 1, \dots, \hat{N}$ . From the central limit theorem [1], the confidence interval of the sample mean  $\hat{y}$  is obtained for a confidence level  $q \in (0, 1)$  as





**Fig. 7.** Landscapes of the critical fractile  $\gamma(x)$  of  $h(x, \xi)$  in (21)

$$\Pr(|\hat{y} - \alpha| \leq \varepsilon) = \Pr\left(|\hat{y} - \alpha| \leq z_{q/2} \sqrt{\frac{\alpha(1-\alpha)}{\hat{N}}}\right) \geq 1 - q \quad (18)$$

where  $\varepsilon$  is a margin of error and  $z_{q/2}$  is the  $z$ -score for  $q/2 \in (0, 0.5]$ .

From desired  $\varepsilon$  and  $q$  in (18), the sample size  $\hat{N}$  is determined as

$$\hat{N} = \left(\frac{z_{q/2}}{\varepsilon}\right)^2 \alpha(1-\alpha). \quad (19)$$

In this paper,  $\varepsilon = 10^{-3}$  and  $q = 0.01$  are chosen in (18). Therefore, if  $\alpha = 0.9$  is given by the CCP in (2), we have  $\hat{N} = 597,128$  from (19).

## 5 Numerical Experiment on Test Problem

### 5.1 Test Problem of CCP

The following function  $h(x)$ ,  $x \in [0, 1]$  has five unequal valleys [18].

$$h(x) = \begin{cases} 1 - e(x) |\sin(5\pi x)|^{0.5} & \text{if } 0.4 < x \leq 0.6 \\ 1 - e(x) \sin(5\pi x)^6 & \text{otherwise} \end{cases} \quad (20)$$

where  $e(x) = \exp(-2 \log_2((x - 0.1)/0.8)^2)$ .

A random variable  $\xi \in \mathfrak{R}$  is added to the function  $h(x)$  in (20) as

$$h(x, \xi) = h(x + \xi), \quad \xi \sim \mathcal{N}(0, \sigma^2). \quad (21)$$

Figure 7 illustrates the landscapes of the critical fractiles  $\gamma(x) = F^{-1}(x, \alpha)$  evaluated from the CDF of  $h(x, \xi)$  in (21). From Fig. 7, the value of  $\gamma(x)$  depends not only on the sufficiency level  $\alpha$  but also on the variance  $\sigma^2$  in (21).

As an instance of the CCP in (2),  $g(\mathbf{x}, \boldsymbol{\xi})$  is defined as

$$g(\mathbf{x}, \boldsymbol{\xi}) = \sqrt{h(x_1, \xi_1) h(x_2, \xi_2)} \quad (22)$$

where  $h(x_j, \xi_j)$  is given by (21).  $\xi_1$  and  $\xi_2$  are mutually independent.

**Table 1.** Comparison of JADE and JADEP on the CCP defined by (22)

$\alpha$	$\sigma^2$	JADE			JADEP			Rate
		$\gamma(\mathbf{x}_b)$	$\widehat{\Pr}(A)$	$\beta$	$\gamma(\mathbf{x}_b)$	$\widehat{\Pr}(A)$	$\beta$	
0.7	$0.02^2$	0.305	0.718	0.805	0.305	0.718	0.809	0.139
		(0.003)	(0.002)	(0.013)	(0.003)	(0.001)	(0.003)	(0.034)
0.7	$0.01^2$	0.083	0.719	0.810	0.082	0.716	0.807	0.192
		(0.000)	(0.002)	(0.000)	(0.001)	(0.005)	(0.004)	(0.029)
0.9	$0.02^2$	0.340	0.908	0.950	0.340	0.908	0.949	0.385
		(0.001)	(0.002)	(0.000)	(0.001)	(0.003)	(0.002)	(0.048)
0.9	$0.01^2$	0.213	0.909	0.949	0.196	0.909	0.943	0.407
		(0.035)	(0.002)	(0.002)	(0.008)	(0.006)	(0.004)	(0.049)

## 5.2 Comparison Between JADEP and JADE

JADEP is compared with JADE on the CCP defined by  $g(\mathbf{x}, \boldsymbol{\xi})$  in (22). They are coded by MATLAB. The population size  $N_P = 20$  is used. The maximum number of generations is fixed to  $G_{\max} = 100$ . The sample size  $N = 30$  is used to compose W\_ECDF. JADEP and JADE are run 20 times in each case.

Table 1 shows the result of experiment averaged over 20 runs. In Table 1,  $\gamma(\mathbf{x}_b)$  is the critical fractile attained with the best solution  $\mathbf{x}_b$ . The feasibility of  $\mathbf{x}_b$  is ensured by the empirical probability  $\widehat{\Pr}(A)$  as stated above. The rate denotes the percentage of children eliminated by the pretest of JADEP.

From the rate in Table 1, the pruning effect of the pretest depends on the case, but it works in all cases. From the result of Wilcoxon test about the value of  $\gamma(\mathbf{x}_b)$ , it is confirmed that there is no difference between JADE and JADEP in all cases. Consequently, the proposed pretest can reduce the number of the children examined  $N$  times without spoiling the quality of obtained solution.

## 6 Application to SAW Filter Design

### 6.1 Structure and Mechanism of SAW Filter

A SAW filter consists of some electrodes and reflectors, namely Inter Digital Transducers (IDTs) and Shorted Metal Strip Arrays (SMSAs), fabricated on a piezoelectric substrate. Figure 8 shows the symmetric structure of a resonator type SAW filter. The input-port of SAW filter is connected to two transmitter IDTs (IDT-Ts). The output-port is connected to a receiver IDT (IDT-R).

IDT-T converts electric input signals into acoustic signals. The acoustic signal of a specific frequency resonates between two SMSAs. The resonant frequency depends on the geometrical structure of SAW filter. Then IDT-R reconverts the enhanced acoustic signal to electric output signal. As a result, the resonator type SAW filter in Fig. 8 works as an electro-mechanical band-pass filter.

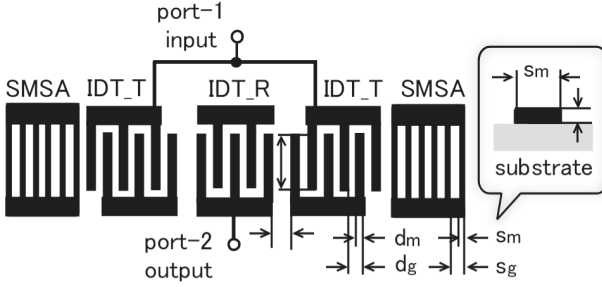


Fig. 8. Symmetric structure of resonator type SAW filter

Table 2. Design parameters of SAW filter

$x_j$	$e_j$	$[x_j, \bar{x}_j]$	Description
$x_1$	—	[0.25, 0.35]	Thickness of electrode
$x_2$	—	[0.45, 0.55]	Metallization ratio of IDT: $d_m/d_g$
$x_3$	—	[0.45, 0.55]	Metallization ratio of SMSA: $s_m/s_g$
$x_4$	—	[1.0, 1.1]	Pitch ratio of SMSA: $d_g/s_g$
$x_5$	—	[1.0, 1.1]	Gap between IDT_R and IDT_T
$x_6$	—	[250.0, 350.0]	Overlap between electrodes
$x_7$	5.0	[50, 200]	Number of strips of SMSA
$x_8$	1.0	[10.5, 30.5]	Number of finger-pairs of IDT_R
$x_9$	0.5	[10, 30]	Number of finger-pairs of IDT_T

Table 3. JADEP

Parameter	Value
$N_P$	100
$G_{\max}$	200
$N$	100

### 6.2 Design of SAW Filter Under Uncertainty

In order to describe the structure of SAW filter in Fig. 8, design parameters, or decision variables  $\mathbf{x} = (x_1, \dots, x_9)$ , are chosen as shown in Table 2. Each design parameter takes either a continuous value  $x_j \in \mathfrak{R}$  or a discrete value at  $e_j \in \mathfrak{R}$  interval. In the procedure of JADEP, a decision variable  $x_j \in \mathfrak{R}$  is rounded to the nearest discrete value if it has to take a discrete value. Figure 8 also illustrates graphically the design parameters of SAW filter listed in Table 2.

We consider processing errors  $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3) \in \mathfrak{R}^3$  for the thickness of electrode  $x_1$  and the metallization ratios of IDT and SMSA  $x_j, j = 2, 3$  as

$$x_1(1 + \xi_1), \xi_1 \sim \mathcal{E}_{\mathcal{X}\mathcal{P}}(\lambda) = \mathcal{E}_{\mathcal{X}\mathcal{P}}(100) \tag{23}$$

where  $\mathcal{E}_{\mathcal{X}\mathcal{P}}(\lambda)$  denotes the exponential distribution with mean  $1/\lambda$  and

$$x_j + \xi_j, j = 2, 3 \tag{24}$$

where  $(\xi_2, \xi_3) \sim \mathcal{N}_2(\mu_2, \mu_3, \sigma_2^2, \sigma_3^2, \rho) = \mathcal{N}_2(0, 0, 0.01^2, 0.01^2, 0.5)$ .

Each of IDT and SMSA can be modeled by an elemental circuit. Therefore, the equivalent circuit model of SAW filter is built up from the elemental circuits of IDT and SMSA [6, 17], and then transformed to a network model as

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{25}$$

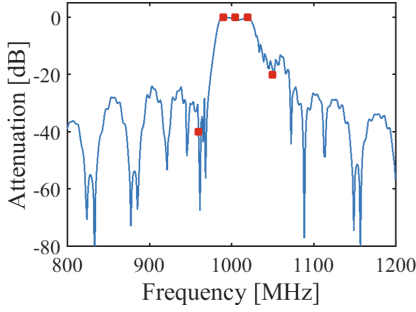


Fig. 9. Reference points  $R(\omega_k)$  in (27)

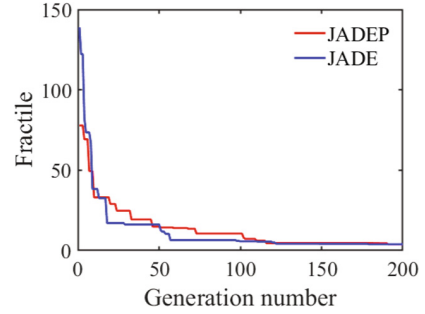


Fig. 10. Convergence plots

where  $a_p, p = 1, 2$  denotes the input signal at port- $p$ , while  $b_p$  denotes the output signal at port- $p$ . Scattering parameter  $s_{pq}$  gives the transition characteristic from port- $q$  to port- $p$ , while  $s_{pp}$  gives the reflection characteristic at port- $p$ .

From (25), the attenuation of SAW filter is defined as

$$L(\mathbf{x}, \boldsymbol{\xi}, \omega) = 20 \log_{10}(|s_{21}(\mathbf{x}, \boldsymbol{\xi}, \omega)|) \quad (26)$$

where  $s_{21}$  depends on  $\mathbf{x} \in \mathbf{X}$ ,  $\boldsymbol{\xi} \in \boldsymbol{\Xi}$ , and frequency  $\omega \in \mathfrak{R}$ .

For the attenuation in (26), some reference points  $R(\omega_k)$  and weights  $c_k$  are specified at frequencies  $\omega_k, k = 1, \dots, M$ . Thereby, the design of SAW filter is formulated as the CCP in (2) by using the following function:

$$g(\mathbf{x}, \boldsymbol{\xi}) = \sum_{k=1}^M c_k (L(\mathbf{x}, \boldsymbol{\xi}, \omega_k) - R(\omega_k))^2 \quad (27)$$

where  $M = 5$  reference points are specified as shown in Fig. 9. Three points are given in the pass-bound and two points are given in the stop-bound.

### 6.3 Result of Experiment and Discussion

JADEP is compared with JADE on the above design problem of SAW filter. Table 3 shows the values of the parameters of JADEP. The same parameter values are used for JADE. Thereby, JADE and JADEP are run on a personal desktop computer (CPU: Intel Core i7@3.40GHz, OS: Windows 7).

Figure 10 shows a typical example of the convergence plots of JADEP and JADE which start from the same initial population. Figure 11 compares JADEP with JADE in the critical fractiles  $\gamma(\mathbf{x}_b)$  of the obtained solutions  $\mathbf{x}_b \in \mathbf{X}$  for some sufficiency levels  $\alpha$ . From Fig. 11, we can confirm the trade-off between the values of  $\gamma(\mathbf{x}_b)$  and  $\alpha$ . From Figs. 10 and 11, there is no significant difference between JADEP and JADE in  $\gamma(\mathbf{x}_b)$ , namely the quality of solution.

Since each sample  $g(\mathbf{u}_i, \boldsymbol{\theta}^n)$  has to be evaluated through the simulation of SAW filter, the efficiency of JADEP is much higher than JADE. In order to obtain a solution of the CCP in (15), JADEP spent 642 [sec] on average except

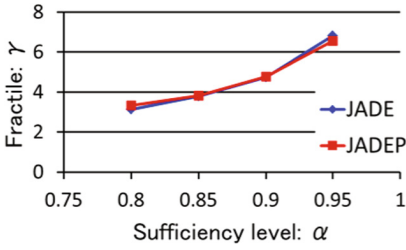


Fig. 11. Trade-off between  $\gamma$  and  $\alpha$

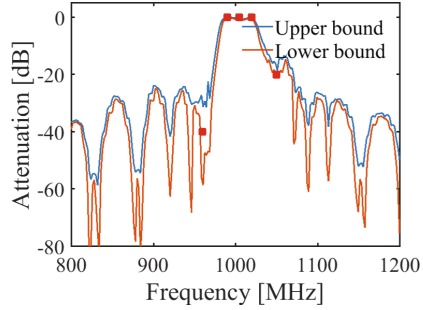


Fig. 12. Prediction interval in (28)

the verification of solution using the Monte Carlo simulation, while JADE spent 1,464 [sec]. The pretest of JADEP discarded more than 70% of the children.

The prediction interval of the attenuation in (26) is defined as

$$\Pr(\underline{L}(\mathbf{x}, \omega) \leq L(\mathbf{x}, \boldsymbol{\xi}, \omega) \leq \bar{L}(\mathbf{x}, \omega)) = (1 - q). \tag{28}$$

From the inverse CDF of  $L(\mathbf{x}, \boldsymbol{\xi}, \omega)$ , the upper and lower bounds are

$$\begin{cases} \bar{L}(\mathbf{x}, \omega) = F^{-1}(\mathbf{x}, \omega, (1 - q/2)) \\ \underline{L}(\mathbf{x}, \omega) = F^{-1}(\mathbf{x}, \omega, q/2). \end{cases} \tag{29}$$

By approximating the CDF in (29) with W\_ECDF, the prediction interval in (28) can be estimated for a solution  $\mathbf{x}_b \in \mathbf{X}$  found by JADEP. Figure 12 shows an example the prediction interval of  $L(\mathbf{x}_b, \boldsymbol{\xi}, \omega)$  estimated for  $q = 0.1$ . The reference points in Fig. 9 exist within the prediction interval in Fig. 12.

By using Figs. 11 and 12, which are provided by the proposed method, we can guarantee the performance of SAW filter under uncertainties.

## 7 Conclusion

For solving CCP efficiently, two new techniques were contrived to improve the optimization method based on JADEP and W\_ECDF. Firstly, THS was used to compose W\_ECDF from fewer samples. Secondly, the sample saving technique called Pretest was introduced into JADE. Finally, the contribution of this paper was demonstrated on the design of SAW filter formulated as CCP.

In this paper, an appropriate value of  $f_{\min}$  in (8) was decided empirically considering the range of PDF and the number of points  $\boldsymbol{\theta}^n \in \mathbf{S}$ . Future work includes how to decide the value of  $f_{\min}$  theoretically for generating THS.

**Acknowledgment.** This work was supported by JSPS (17K06508).

## References

1. Ash, R.B.: Basic Probability Theory. Dover, Downers Grove (2008)
2. Bauer, T., Eggs, C., Wagner, K., Hagn, P.: A bright outlook for acoustic filtering. *IEEE Microwave Mag.* **16**(7), 73–81 (2015)
3. Ben-Tal, A., Ghaoui, L.E., Nemirovski, A.: Robust Optimization. Princeton University Press, Princeton (2009)
4. Geoffrion, A.M.: Stochastic programming with aspiration or fractile criteria. *Manag. Sci.* **13**(9), 672–679 (1967)
5. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* **2**(1), 84–90 (1960)
6. Hashimoto, K.: Surface Acoustic Wave Devices in Telecommunications - Modeling and Simulation. Springer, Heidelberg (2000). <https://doi.org/10.1007/978-3-662-04223-6>
7. Jiekang, W., Jianquan, Z., Guotong, C., Hongliang, Z.: A hybrid method for optimal scheduling of short-term electric power generation of cascaded hydroelectric plants based on particle swarm optimization and chance-constrained programming. *IEEE Trans. Power Syst.* **23**(4), 1570–1579 (2008)
8. Liu, B., Zhang, Q., Fernández, F.V., Gielen, G.G.E.: An efficient evolutionary algorithm for chance-constrained bi-objective stochastic optimization. *IEEE Trans. Evol. Comput.* **17**(6), 786–796 (2013)
9. Lubin, M., Dvorkin, Y., Backhaus, S.: A robust approach to chance constrained optimal power flow with renewable generation. *IEEE Trans. Power Syst.* **31**(5), 3840–3849 (2016)
10. Martinez, A.R., Martinez, W.L.: Computational Statistics Handbook with MATLAB ®, 2nd edn. Chapman & Hall/CRC, Boca Raton (2008)
11. Parkinson, A., Sorensen, C., Pourhassan, N.: A general approach for robust optimal design. *J. Mech. Des.* **115**(1), 74–80 (1993)
12. Poojari, C.A., Varghese, B.: Genetic algorithm based technique for solving chance constrained problems. *Eur. J. Oper. Res.* **185**, 1128–1154 (2008)
13. Prékopa, A.: Stochastic Programming. Kluwer Academic Publishers, Alphen aan den Rijn (1995)
14. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Heidelberg (2005). <https://doi.org/10.1007/3-540-31306-0>
15. Tagawa, K.: A statistical sensitivity analysis method using weighted empirical distribution function. In: Proceedings of the 4th IIAE International Conference on Intelligent Systems and Image Processing, pp. 79–84 (2016)
16. Tagawa, K., Miyanaga, S.: Weighted empirical distribution based approach to chance constrained optimization problems using differential evolution. In: Proceedings of IEEE CEC2017, pp. 97–104 (2017)
17. Tagawa, K., Sasaki, Y., Nakamura, H.: Optimum design of balanced SAW filters using multi-objective differential evolution. In: Dep, K., et al. (eds.) SEAL 2010. LNCS, vol. 6457, pp. 466–475. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17298-4\\_50](https://doi.org/10.1007/978-3-642-17298-4_50)
18. Tsutsui, S.: A comparative study on the effects of adding perturbations to phenotypic parameters in genetic algorithms with a robust solution searching scheme. In: Proceedings of IEEE SMC, pp. 12–15 (1999)
19. Wang, X.: Randomized Halton sequences. *Math. Comput. Model.* **32**, 887–899 (2000)
20. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **13**(5), 945–958 (2009)



# Directed Locomotion for Modular Robots with Evolvable Morphologies

Gongjin Lan<sup>(✉)</sup>, Milan Jelisavcic, Diederik M. Roijers, Evert Haasdijk, and A. E. Eiben

Department of Computer Science, VU University Amsterdam,  
Amsterdam, The Netherlands

{g.lan,m.j.jelisavcic,d.m.roijers,a.e.eiben}@vu.nl

**Abstract.** Morphologically evolving robot systems need to include a learning period right after ‘birth’ to acquire a controller that fits the newly created body. In this paper, we investigate learning one skill in particular: walking in a given direction. To this end, we apply the HyperNEAT algorithm guided by a fitness function that balances the distance travelled in a direction and the deviation between the desired and the actually travelled directions. We validate this method on a variety of modular robots with different shapes and sizes and observe that the best controllers produce trajectories that accurately follow the correct direction and reach a considerable distance in the given test interval.

**Keywords:** Evolutionary robotics · Evolvable morphologies  
Modular robots · Gait learning · Directed locomotion

## 1 Introduction

While it can already be hard to design robots for known environments, it is considerably harder for (partially) unknown environments, like the deep sea or Venus. In unknown environments, robots should be able to respond to the circumstances they encounter. The problem with this however, is that there is no way to predict what the robots will encounter. Therefore, in such environments, it would be highly useful to have robots that evolve over time, changing their controllers and their morphologies to better adapt to the environment.

The field that is concerned with such evolving robots is Evolutionary Robotics [6, 10]. To date, the research community has mainly been focussing on evolving only the controllers in fixed robot bodies. The evolution of morphologies has received much less attention even though it has been observed that adequate robot behaviour depends on both the body and the brain [3, 4, 30]. To unlock the full potential of the evolutionary approach one should apply it to both bodies and brains. At present, we can only do this in simulation, as there are still key obstacles to overcome for evolving robots in real hardware [12, 13, 21].

One of the challenges inherent to evolving robot bodies – be it simulated or real – is rooted in the fact that ‘robot children’ are random combinations

of the bodies and brains of their parents. In general it cannot be assumed that simply recombining the parents' controllers results in a controller that fits the recombined body. Hence, a 'robot child' must learn how to control its body, not unlike a little calf that spends the first hour of its life learning to walk. It is vital that the learning method is general enough to work for a large variety of morphologies and fast enough to work within practical time intervals.

A generic architecture of robot systems, where both morphologies and controllers undergo evolution has been introduced recently [11, 14]. The underlying model, called the Triangle of Life (ToL), describes a life cycle that runs from conception (being conceived) to conception (conceiving offspring) through three principal stages: Birth, Infancy, and Mature Life. Within this scheme, the learn-to-control-your-own-body problem can be positioned in the Infancy phase, where a newborn robot acquires the basic sensory-motor skills. Formerly, we have investigated the most elementary case: gait learning [21–23, 35]. However, although gait learning is a popular problem in evolutionary robotics, in practice we are not really interested in a robot that just walks without purpose. For most cases, a robot has to move in a given direction, e.g., to move towards a destination. Here we focus on the task of directed locomotion, where the robot must follow a given direction, e.g. "go left". Our specific research goals are the following:

1. Develop a dedicated evaluation function that balances the distance travelled in a direction and the deviation between the desired and the actually travelled directions.
2. Provide a method to learn a controller for directed locomotion in different modular robots.
3. Evaluate the method on a test suite consisting of robots with different shapes and sizes.

## 2 Related Work

The design of locomotion for modular robots is a difficult task. Several approaches based on various types of controllers and algorithms for locomotion of robots have proposed in [1, 32]. An early approach is based on gait control tables that in essence are a simple cyclic finite state machines [5]. A second major approach is based on neural networks, for instance, HyperNEAT. In previous work we have implemented evolutionary controllers for locomotion in modular robots [16, 35] using HyperNEAT. Other studies also have shown that HyperNEAT can evolve the good controllers for the efficient gaits of a robot [8, 36]. Other successful approaches that have been extensively investigated for robot locomotion are based on Central Pattern Generators (CPGs) [19]. CPGs are neural networks that can produce rhythmic patterned outputs without rhythmic sensory or central input [17]. The use CPG-based controllers reduces the dimensionality of the locomotion control problem while remaining flexible enough to continuously adjust velocity, direction, and type of gait depending on the environmental context [20]. This technique has been shown to produce well-performing and stable gaits for modular robots [24, 25, 27]. Last, an alternative



approach based on machine learning for adaptive locomotion was proposed by Cully et al., to account for changes in body properties [9].

Although there are extensive existing studies on the locomotion of robots, most of them focus on the controllers in fixed robot bodies for gait learning, and only the research described in [24, 32] tested on multiple shapes. Our own previous work [21–23, 35] focussed on gait learning for modular robots with evolvable morphologies. For directed locomotion, most related studies with robots concern the control of vertebrates with fixed shapes, such as a bipeds. The different neural control systems involved in directed vertebrates locomotion are reviewed in [15]. A CPG approach based on phase oscillators towards directed biped locomotion is presented in [28]. A special snake-like robot with screw-drive units is presented in [7] for directed locomotion using a reinforcement learning approach. There are few studies on the directed locomotion of the modular robots, and they focus on fixed morphologies or the special structures.

### 3 Experimental Set-Up

In this study, the controllers for all modular robots are learned in an infinite plane environment [18], using our Gazebo-based<sup>1</sup> custom simulator *Revolve*.

#### 3.1 Robots

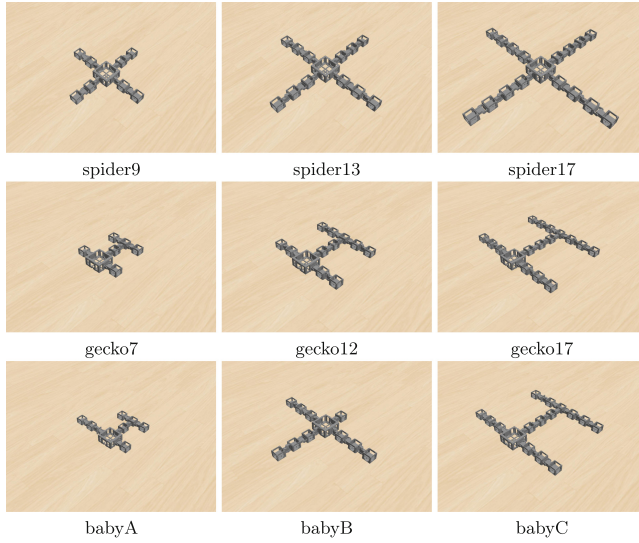
Our robot design is based on RoboGen [2]. We use a subset of those 3D-printable components: *fixed bricks*, a *core component*, and *active hinges*. The fixed bricks are cubic components with slots that can attach other components. The core component holds a controller board. It also has slots on its four lateral faces to attach other components. The active hinge is a joint moved by a servo motor. It can attach to other components by inserting its lateral faces into the slots of these other components. Each robot’s genotype describes its layout and consists of a tree structure with the root node representing a core module from which further components branch out. These models are used in simulation, but also could be used for 3D printing and the construction of the real robots.

As a test suite we chose nine robots in three different shapes and sizes, to examine the generality and scalability of our method, see Fig. 1. We refer to these three shapes as *spider*, *gecko*, and *baby*. The ‘baby’ robots were created through recombination of the ‘spider’s’ and ‘gecko’s’ [22] morphological genotypes.

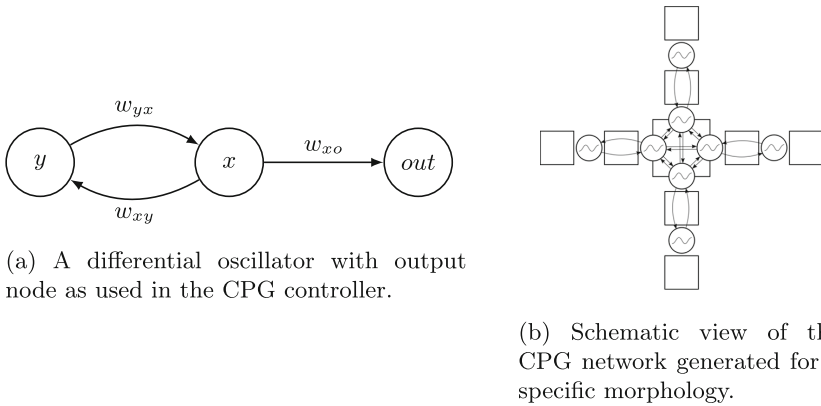
#### 3.2 Controllers

Controllers based on Central Pattern Generators (CPGs) have been proven to perform well for modular robots. In this work, we use CPGs whose main components are differential oscillators. Each oscillator is defined by two neurons that

<sup>1</sup> <http://gazebosim.org/>.



**Fig. 1.** Images of the used robots. Note that the top leg of gecko17 and babyC are different; babyC has one more active hinge where gecko17 has a brick.



(a) A differential oscillator with output node as used in the CPG controller.

(b) Schematic view of the CPG network generated for a specific morphology.

**Fig. 2.** Controller concept used in the robots. In (b) the rectangular shapes indicate passive body parts, the circles show active hinges, each with their own differential oscillator, and the arrows indicate the connections between the oscillators for the body shown in the top-left panel of Fig. 1.

are recursively connected as shown in Fig. 2a. These generate oscillatory patterns by calculating their activation levels  $x$  and  $y$  according to the following differential equation:

$$\dot{x} = w_{yx}y + bias_x$$

$$\dot{y} = w_{xy}x + bias_y$$

with  $w_{xy}$  and  $w_{yx}$  denoting the weights of the connections between the neurons;  $bias_x$  and  $bias_y$  are parameters of the neurons. If  $w_{yx}$  and  $w_{xy}$  have different signs the activation of the neurons  $x$  and  $y$  is periodic and bounded.

We used Compositional Pattern-Producing Networks (CPPNs) to generate the weights of the CPG controller. CPPNs are a variation of artificial neural networks (ANNs) that have an architecture whose evolution is guided by HyperNEAT algorithm [33], so that the substrate network’s performance is optimised [34]. The CPG nodes are positioned in a three-dimensional space. Such *modular differentiation* allows specialisation of the active hinge’s movements depending on its relative position in the robot. The hinge coordinates are obtained from a top-down view of the robot body. Thus, two coordinates of a node in the CPG controller correspond to the relative position of the active hinge it is associated with. The third coordinate depends on the role of the node in the CPG network: output nodes have a value of 0 and differential nodes have values of 1 for  $x$  and  $-1$  for  $y$  nodes. Therefore the CPPNs have six inputs denoting the coordinates of a source and a target node when querying connection weights or just the position of one node when obtaining node parameters with the other three inputs being initialised as zero. The CPPNs have three outputs: the weight of the connection from source to target as well as the bias and gain values when calculating parameters for a node.

The CPPNs return the connection weights for the CPG network that in turn constitutes the controller that induces the behaviour for directed locomotion. The behaviour is evaluated by a fitness function (Sect. 4) and the fitness value is fed to HyperNEAT which in turn generates new CPPNs. The CPPNs evolve until a termination condition is triggered; in our experiments this is reaching a maximum number of generations.

### 3.3 Experimental Parameters

An initial population of 20 CPPNs are randomly generated in the first generation. Each CPPN generates the weights of a CPG network whose topology is based on a robot’s morphology. The fitness of the CPG is evaluated in Revolve for a given evaluation time. We set this evaluation time to be 60 s to balance computing time and accurately evaluating a complex task as directed locomotion. We found this 60 s to be a suitable value empirically. Each EA run is terminated after 300 generations, that is,  $300 * 20 = 6000$  fitness evaluations – this amounts to 100 h of (simulated) time.

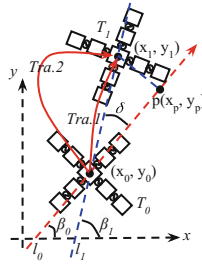
The robots used in the experiments include three small robots (spider9, gecko7, babyA), three medium size robots (spider13, gecko12, babyB) and three large robots (spider17, gecko17, babyC). For each robot we tested the EA on five target directions ( $-40^\circ$ ,  $-20^\circ$ ,  $0^\circ$ ,  $20^\circ$ , and  $40^\circ$  relative to the robot) to simulate the robot’s limited field of view in the real-world. This resulted in 45 test cases. For each test case the EA runs were repeated five times. All together, we performed 225 HyperNEAT runs per 100 h of simulated time each (Table 1).

**Table 1.** Experimental parameters

Parameter	Value	Description
Population size	20	Number of individuals per generation
Generations	300	Termination condition for each run
Tournament size	4	Number of individuals used in tournament selection
Mutation	0.8	Probability of mutation for individuals
Evaluation time	60	Duration of the test period per fitness evaluation in seconds

## 4 Fitness Function

In this section, we propose a fitness function for directed locomotion and illustrate how the performance of a controller is evaluated. We provide a step-by-step derivation that leads to our final fitness function shown in Eq. 5.



**Fig. 3.** Illustration of the fitness calculation for each evaluation.  $T_0$  is the starting position of the robot, with coordinate  $(x_0, y_0)$ .  $T_1$  is the end position of the robot, with coordinate  $(x_1, y_1)$ .  $l_0$  is a given target direction. The point  $p$  is the projected point on the target direction  $l_0$ . The red lines  $Tra.1$  and  $Tra.2$  show two different trajectories of the robot. (Color figure online)

The scenario for an evaluation in our experiments is illustrated in Fig. 3. We can collect the following measurements from the Revolve simulator:

1.  $c_0 = (x_0, y_0)$  is the coordinate of the core component of the robot at the start of the simulation, i.e., time  $T_0$ .
2.  $c_1 = (x_1, y_1)$  is the coordinate of the core component of the robot at the end of the simulation,  $T_1$ .
3. The orientation of the robot in  $T_0$  and  $T_1$ .
4. The length of the trajectory that the robot travelled from  $c_0$  to  $c_1$

The target direction,  $\beta_0$ , is an angle with respect to the initial orientation of the robot at  $T_0$ . In Fig. 3 we drew lines in the target direction,  $l_0$ , and the line through  $c_0$  and  $c_1$ ,  $l_1$ . The angle between  $l_1$  and  $x$ -axis,  $\beta_1 = \text{atan2}((y_1 - y_0), (x_1 - x_0))$ , is the actual direction of the robot displacement between  $T_0$  and  $T_1$ .

The absolute intersection angle between  $l_0$  and  $l_1$ ,  $\delta$ , is the deviation between the actual direction of the robot locomotion and the target direction. It can be calculated as:

$$\delta = \begin{cases} 2 * \pi - |\beta_1 - \beta_0| & (|\beta_1 - \beta_0| > \pi) \\ |\beta_1 - \beta_0| & (|\beta_1 - \beta_0| \leq \pi) \end{cases} \quad (1)$$

Note that we pick the smallest angle between the two lines. To perform well on a directed locomotion task,  $\delta$  should be as small as possible. However, just minimizing  $\delta$  is not enough to for successful directed locomotion.

In addition to moving in the right direction, i.e., minimizing  $\delta$ , the robot should move as far as possible in the target direction. Therefore, we calculate distance travelled by the robot in the target direction by projecting the final position at  $T_1$ ,  $(x_1, y_1)$ , onto  $l_0$ ; we denote this point as  $p = (x_p, y_p)$ . The distance travelled is then

$$distProjection = sign |p - c_0|, \quad (2)$$

where  $|p - c_0|$  is the Euclidean distance between  $p$  and  $c_0$ , and  $sign = 1$  if  $\delta < \frac{\pi}{2}$  (noting that  $\delta$  is an absolute value) and  $sign = -1$  otherwise. The  $distProjection$  is thus negative when the robot moves in the opposite direction.

To further penalize deviating from the target direction we calculate the distance between  $(x_1, y_1)$  and  $(x_p, y_p)$ :

$$penalty = f_p * |c_1 - p|, \quad (3)$$

where  $|c_1 - p|$  is the Euclidean distance between  $c_1$  and its projection on the target direction line  $l_0$ ,  $p$ .  $f_p$  is a constant scalar penalty factor, determining the relative importance of the deviation. In our experiments we use  $f_p = 0.01$ .

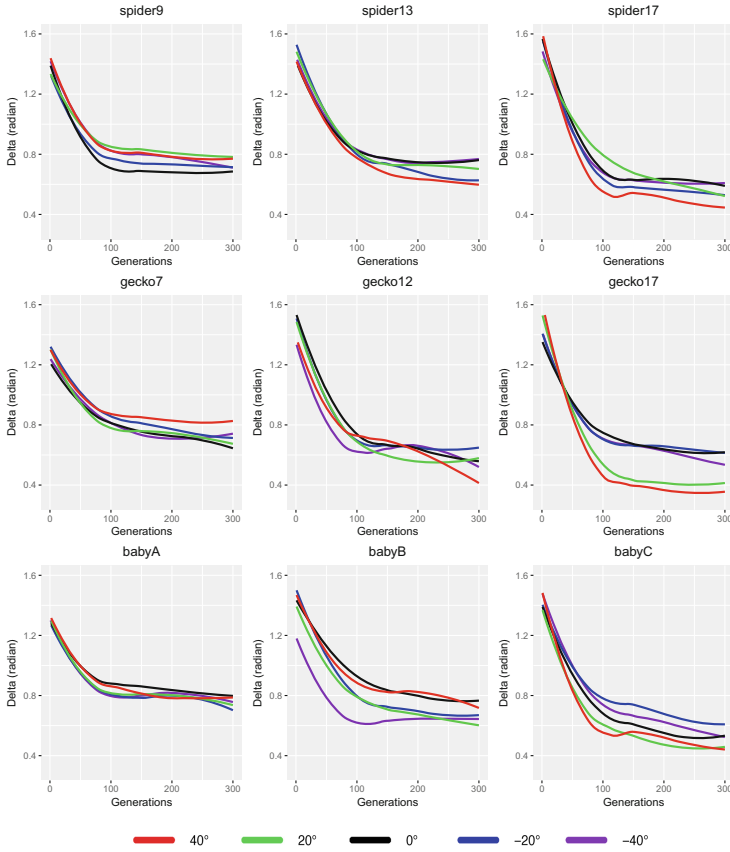
A naive version of the fitness would be:

$$fitnessPro = \frac{distProjection}{\delta + 1} - penalty, \quad (4)$$

where  $(\delta + 1)$  aims to guarantee that the denominator does not equal zero.

While  $fitnessPro$  is proportional to  $distProjection$ , and inversely proportional to  $\delta$  and  $penalty$ , this does not yet entirely express all desirable features of a good trajectory for the robot. Specifically, we not only care about the final position of the robot, but also about how the robot moves to the end point. To illustrate this please compare the trajectories marked  $Tra.1$  and  $Tra.2$  in Fig. 3. Although the robot has the same starting and end position for both trajectories,  $Tra.1$  is a more efficient way of moving between the two points. Therefore, we would want the controller of  $Tra.1$  to have a higher fitness than that of  $Tra.2$ . In general, we aim to evolve a controller to move from start to finish as efficiently as possible, i.e., in a straight line. Therefore, we make the fitness function inversely proportional to the length of the trajectory (noted as  $lengthTra$ ) that the robot performs. We thus propose the following fitness function to measure the performance of controllers for directed locomotion:

$$fitness = \frac{|distProjection|}{lengthTra + \varepsilon} * \left( \frac{distProjection}{\delta + 1} - penalty \right) \quad (5)$$



**Fig. 4.** Deviation ( $\delta$ ) from the target direction during the learning process (Color figure online).

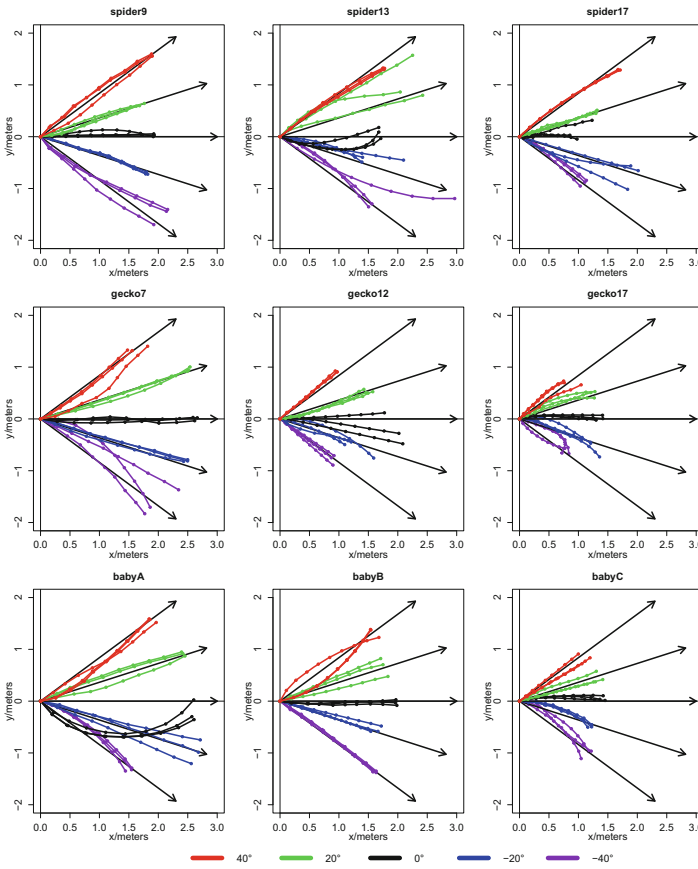
where  $\varepsilon$  is an infinitesimal constant. The fitness function is proportional to *distProjection*, but inversely proportional to *lengthTra* and  $\delta$ . That is, the fitness function rewards higher speeds in the target direction (as measured through *distProjection*), and punishes the length of trajectories, *lengthTra*, and deviations from the target directions.

## 5 Experimental Results

Inspecting the usual fitness vs. time curves (omitted here because of space limitations) we observe that the controllers of small size robots have the highest average fitness. The controllers of medium and large size robots reach significantly lower values. This is in line with our previous work [22] suggesting that the parameter settings for the larger robots are more difficult to learn, irrespective of the algorithm, such as HyperNEAT or RL PoWER.

An important metric for directed locomotion is the deviation from the target direction,  $\delta$ . The progression of the learning process is shown in Fig. 4 for each of the nine robots. Each sub-figure shows the average  $\delta$  for the 20 controllers in a population over five repetitions. The five target directions are represented by the colours. These curves show that in all cases  $\delta$  gradually decreases. Interestingly, the  $\delta$  of small size robots is higher than for the larger robots. This means that small size robots are easier to evolve for speed (as they have higher fitness), but do worse in terms of deviation. Similar results were shown in our previous work [22]. We hypothesize that this is because larger robots have more joints, they have more flexibility, and can control their direction more precisely.

To see the outcome of the learning process we select the best controllers from the 30000 controllers (6000 evaluations per run, 5 repetitions) for each robot in each target direction and inspect the trajectories these controllers induce. The best three trajectories for each robot and direction are shown in Fig. 5.



**Fig. 5.** The best three trajectories for each robot and each direction. The black arrows show the five target directions.

In general, the trajectories follow the target directions well. For example, the trajectories of spider9 are almost exactly on the target directions and they display faster speed than other robots. Because maximizing the distance in the target directions, *distProjection*, is rewarded in the fitness function, as well as minimizing the deviation from the target directions, evolution can lead to different trade-offs between these two preferences. For example, one of the trajectories (purple point-line) for  $-40^\circ$  of spider13 deviates quite far from the target direction but travels a long distance, while the other trajectories for this robot and direction get less far but stick more closely to the target direction. In addition, although the trajectories (black point-line) for  $0^\circ$  of babyA have high values for *lengthPath*, and thus receive a punishment in the fitness function for the deviation from the straight line in the target direction of  $0^\circ$ , they have top fitness because of the high speed (*distProjection*) and a good final  $\delta$ . The small size robots have the better trajectories, especially in terms of speed. The medium size robots have the second-best trajectories. The large size robots also have good trajectories but not as good as the small and medium size robots, especially in terms of speed. In summary, we conclude that using our method, successful controllers can be evolved for directed locomotion for modular robots with evolvable morphologies. Furthermore, the small-sized robots have the better performance for directed locomotion, especially in terms of speed in the target direction.

## 6 Concluding Remarks

We addressed the problem of learning sensory-motor skills in morphologically evolvable robot systems where the body of newborn robots can be a random combination of the bodies of the parents. In particular, we presented a method to learn good robot controllers for directed locomotion based on HyperNEAT and a new fitness function that balances the distance travelled in a desired direction and the angle between the desired direction and the direction actually travelled. We tested this method on nine modular robots for five different target directions and found that the robots acquired good controllers in all cases. From the resulting trajectories it is apparent that our fitness function adequately balances the speed and direction of the robots.

These experiments were, while well-performing, not too efficient, as the learning speed of HyperNEAT is not very high. Currently we are comparing HyperNEAT to other methods for training the controllers, such as reinforcement learning [26] and Bayesian optimisation [29]. Furthermore, we aim to investigate which other trade-offs between deviation from the target direction and the speed exist by using a vector-valued, i.e., multi-objective, rather than a scalar fitness function [31]. Finally, we aim to validate our results by replicating the experiments in real hardware and consider more scenarios and other skills.



## References

1. Aoi, S., Manoonpong, P., Ambe, Y., Matsuno, F., Wörgötter, F.: Adaptive control strategies for interlimb coordination in legged robots: a review. *Front. Neurobotics* **11**, 39 (2017)
2. Auerbach, J., et al.: RoboGen: robot generation through artificial evolution. In: Sayama, H., Rieffel, J., Risi, S., Doursat, R., Lipson, H. (eds.) *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, pp. 136–137. The MIT Press, New York, July 2014
3. Auerbach, J.E., Bongard, J.C.: On the relationship between environmental and morphological complexity in evolved robots. In: *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, pp. 521–528. GECCO 2012. ACM, New York (2012)
4. Beer, R.D.: *The Dynamics of Brain–Body–Environment Systems: A Status Report* (2008)
5. Bongard, J., Zykov, V., Lipson, H.: Resilient machines through continuous self-modeling. *Science* **314**(5802), 1118–1121 (2006)
6. Bongard, J.C.: Evolutionary robotics. *Commun. ACM* **56**(8), 74–83 (2013)
7. Chatterjee, S., et al.: Reinforcement learning approach to generate goal-directed locomotion of a snake-like robot with screw-drive units. In: *2014 23rd International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, pp. 1–7, September 2014
8. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the hyperneat generative encoding. In: *2009 IEEE Congress on Evolutionary Computation*, pp. 2764–2771, May 2009
9. Cully, A., Clune, J., Tarapore, D., Mouret, J.B.: Robots that can adapt like animals. *Nature* **521**, 503 (2015)
10. Doncieux, S., Bredeche, N., Mouret, J.B., Eiben, A.: Evolutionary robotics: what, why, and where to. *Front. Robot. AI* **2**(4) (2015)
11. Eiben, A., et al.: 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)
12. Eiben, A., Kernbach, S., Haasdijk, E.: Embodied artificial evolution. *Evol. Intell.* **5**(4), 261–272 (2012)
13. Eiben, A., Smith, J.: From evolutionary computation to the evolution of things. *Nature* **521**(7553), 476–482 (2015)
14. Eiben, A.E.: In vivo veritas: towards the evolution of things. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) *PPSN 2014. LNCS*, vol. 8672, pp. 24–39. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10762-2\\_3](https://doi.org/10.1007/978-3-319-10762-2_3)
15. Grillner, S., Wallén, P., Saitoh, K., Kozlov, A., Robertson, B.: Neural bases of goal-directed locomotion in vertebrates-an overview. *Brain Res. Rev.* **57**(1), 2–12 (2008)
16. Haasdijk, E., Rusu, A.A., Eiben, A.E.: HyperNEAT for locomotion control in modular robots. In: Tempesti, G., Tyrrell, A.M., Miller, J.F. (eds.) *ICES 2010. LNCS*, vol. 6274, pp. 169–180. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15323-5\\_15](https://doi.org/10.1007/978-3-642-15323-5_15)
17. Hooper, S.L.: Central pattern generators. In: *Encyclopedia of Life Sciences*, pp. 1–12, April 2001. <https://doi.org/10.1038/npg.els.0000032>

18. Hupkes, E., Jelisavcic, M., Eiben, A.E.: Revolve: a versatile simulator for online robot evolution. In: Sim, K., Kaufmann, P. (eds.) *EvoApplications 2018*. LNCS, vol. 10784, pp. 687–702. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77538-8\\_46](https://doi.org/10.1007/978-3-319-77538-8_46)
19. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. *Neural Netw.* **21**(4), 642–653 (2008). *Robotics and Neuroscience*
20. Ijspeert, A.J., Crespi, A., Ryczko, D., Cabelguen, J.M.: From swimming to walking with a salamander robot driven by a spinal cord model. *Science* **315**(5817), 1416–1420 (2007)
21. Jelisavcic, M., et al.: Real-world evolution of robot morphologies: a proof of concept. *Artif. Life* **23**(2), 206–235 (2017)
22. Jelisavcic, M., Carlo, M.D., Haasdijk, E., Eiben, A.E.: Improving RL power for on-line evolution of gaits in modular robots. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8, December 2016
23. Jelisavcic, M., Haasdijk, E., Eiben, A.: Acquiring moving skills in robots with evolvable morphologies: recent results and outlook. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2017* (2017)
24. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic locomotion design and experiments for a modular robotic system. *IEEE/ASME Trans. Mech.* **10**(3), 314–325 (2005)
25. Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Kokaji, S., Murata, S.: Distributed adaptive locomotion by a modular robotic system, M-TRAN II. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), vol. 3, pp. 2370–2377, September 2004
26. Kohl, N., Stone, P.: Policy gradient reinforcement learning for fast quadrupedal locomotion. In: *IEEE International Conference on 2004 Proceedings of Robotics and Automation, ICRA 2004*, vol. 3, pp. 2619–2624 (2004)
27. Marder, E., Bucher, D.: Central pattern generators and the control of rhythmic movements. *Curr. Biol.* **11**(23), R986–R996 (2001)
28. Matos, V., Santos, C.P.: Towards goal-directed biped locomotion: combining CPGs and motion primitives. *Robot. Auton. Syst.* **62**(12), 1669–1690 (2014)
29. Paul, S., Chatzilygeroudis, K., Ciosek, K., Mouret, J.B., Osborne, M.A., Whiteson, S.: Alternating optimisation and quadrature for robust control. In: *The Thirty-Second AAAI Conference on Artificial Intelligence, AAAI 2018* (2018)
30. Pfeifer, R., Bongard, J.C.: *How the Body Shapes the Way We Think: A New View of Intelligence* (Bradford Books). The MIT Press, Cambridge (2006)
31. Roijers, D.M., Whiteson, S.: Multi-objective decision making. *Synth. Lect. Artif. Intell. Mach. Learn.* **11**(1), 1–129 (2017)
32. Sproewitz, A., Moeckel, R., Maye, J., Ijspeert, A.J.: Learning to move in modular robots using central pattern generators and online optimization. *Int. J. Robot. Res.* **27**(3–4), 423–443 (2008)
33. Stanley, K.O.: Compositional pattern producing networks: a novel abstraction of development. *Genet. Program. Evolvable Mach.* **8**(2), 131–162 (2007)
34. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. *Evol. Comput.* **10**(2), 99–127 (2002)
35. Weel, B., D’Angelo, M., Haasdijk, E., Eiben, A.: Online gait learning for modular robots with arbitrary shapes and sizes. *Artif. life* **23**(1), 80–104 (2017)
36. Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J.C., Lipson, H.: Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization. In: *Proceedings of the 20th European Conference on Artificial Life*, pp. 890–897 (2011)



# Optimisation and Illumination of a Real-World Workforce Scheduling and Routing Application (WSRP) via Map-Elites

Neil Urquhart<sup>(✉)</sup> and Emma Hart

School of Computing, Edinburgh Napier University, Scotland, UK  
{n.urquhart,e.hart}@napier.ac.uk

**Abstract.** Workforce Scheduling and Routing Problems (WSRP) are very common in many practical domains, and usually have a number of objectives of interest to the end-user. Illumination algorithms such as Map-Elites (ME) have recently gained traction in application to *design* problems, in providing multiple diverse solutions as well as illuminating the solution space in terms of user-defined characteristics, but typically require significant computational effort to produce the solution archive. We investigate whether ME can provide an effective approach to solving WSRP, a *repetitive* problem in which solutions have to be produced quickly and often. The goals of the paper are two-fold. The first is to evaluate whether ME can provide solutions of competitive quality to an evolutionary algorithm in terms of a single objective function, and the second to examine its ability to provide a repertoire of solutions that maximise user choice. We find that very small computational budgets favour the EA in terms of quality, but ME outperforms the EA at larger budgets, provides a more diverse array of solutions, and lends insight to the end-user.

## 1 Introduction

Workforce scheduling and routing problems (WSRP) [3] are challenging problems for organisations with staff working in areas including health care [2] and engineering [5]. Finding solutions is the responsibility of a planner within the organisation who will have an interest in the wider organisational policy decisions surrounding the solution. Such wider issues could include the implications of solutions with a lower environmental impact, the effects of switching to public transport, or the impact of changing the size of the workforce.

Multi-objective optimisation approaches are commonly used to find solutions, to WSRP instances, as they can provide a front of solutions that trade-off objectives [13]. However, fronts may only comprise a small section of the total solution space, and are difficult to visualise if there are many dimensions. Thus, it can be difficult for a planner to understand the range of solutions, why solutions were produced, and in particular to know whether other compromise solutions might exist.

A class of algorithms known as *illumination* algorithms have recently been introduced by Mouret *et al.* [7], with a number of variants following, e.g. [8, 10]. Fundamentally different to a traditional search algorithm, the approach provides a holistic view of how high-performing solutions are distributed throughout a solution space [7]. The method creates a map of high-performing solutions at each point in a space defined by dimensions of variation that are chosen by a user, according to characteristics of a solution that are of interest. The resulting map (a Multi-dimensional Archive of Phenotypic Elites) enables the user to gain specific insight into how different combinations of characteristics of solutions correlate with performance, hence providing insight as well as multiple potential solutions. In addition, as the approach encourages diversity, it has often been shown to be more capable of fully exploring a search-space, outperforming state-of-the-art search algorithms given a single-objective, and can be particularly helpful in overcoming deception [9]. We therefore hypothesise that an illumination algorithm might provide particular benefit to real-world problems such as WSRP, which contain multiple, and sometimes conflicting, objectives. However, in contrast to the majority of previous applications of Map-Elites which fall mainly in the domain of design problems (e.g. designing robot morphology), WSRP is a repetitive problem, which requires solving new instances repeatedly and obtaining acceptable solutions in reasonable time. While investing effort into producing an archive of solutions can pay off in a design domain, it may prove prohibitive for repetitive problems. Therefore, in the context of a WSRP based on the city of London, using real geographical locations and real transport information. Previous approaches to solving the problem [14] has utilised a portfolio of multi-objective Evolutionary Algorithms to produce a non-dominated front, the principle contribution lies in the application of MAP-Elites to illuminate a combinatorial ESRP problem. To assess the success of MAP-Elites in this context we consider the following questions:

1. How does the relative performance of ME compare to a standard Evolutionary Algorithm (EA) in terms of satisfying a single objective-function over a range of evaluation budgets?
2. Does MAP-Elites provide useful insights into problem characteristics from a real-world perspective through providing a range of diverse but high-quality solutions?

Using 10 realistic problem instances, we demonstrate that for a small fixed evaluation budget, MAP-Elites does not outperform an EA in terms of the objective function, but as the budget increases, it outperforms the EA on the majority of instances tested. Furthermore, even when it is outperformed by an EA in terms of the single objective, it can discover solutions that have better values for the individual characteristics. From a user-perspective, it may therefore present an acceptable trade-off between overall quality and insight.

## 2 Previous Work

The Workforce Scheduling and Routing Problem (WSRP) was defined in [3] as a scenario that involves the mobilisation of personnel in order to perform work related activities at different locations. It has been tackled by a variety of methods including meta-heuristics [1] and hyper-heuristics [5]. It can involve consideration of many constraints and objectives, for example transport modality, time-windows, transport cost, travel cost etc. and hence is often treated as multi-objective problem, e.g. [13]. The reader is referred to [3] for a detailed survey on previous approaches.

The Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) was first introduced by Mouret *et al.* [7] and as discussed in the introduction, provides a mechanism for illuminating search spaces by creating an archive of high-performing solutions mapped onto solution characteristics defined by the user. To date, the majority of applications of illumination algorithms have been to *design* problems [7, 16]. Another tranche of work focuses on *behaviour* evolution in robotics, for example Cully *et al.* [4], who evolve a diverse set of behaviours for a single robot in a “pre-implementation” simulation phase: these are then used in future when the robot is in operation to guide intelligent choice of behaviour given changing environmental conditions.

To the best of our knowledge, an illumination algorithm has never been used to solve repetitive problems, i.e. problems faced in the real-world where acceptable solutions to problems have to be discovered in short time-frames, often many times a day. Typically these types of problems are combinatorial optimisation problems, e.g. scheduling, routing and packing, that often utilise indirect genotypic representations as a result of having to deal with multiple constraints. This contrasts to much of the existing work using MAP-Elites which uses a direct representation of design parameters (although the use of MAP-Elites with an indirect representation was discussed in [11]).

## 3 Methodology

We consider a WSRP characterised by time-windows, multiple transport modes and service times. Variations of this scenario include the scheduling of health and social care workers as well as those providing other services such as environmental health inspections.

We assume an organisation has to service a set of clients, who each require a single visit. Each of the visits  $v$  must be allocated to an employee, such that all clients are serviced, and an unlimited number of employees are available. Each visit  $v$  is located at  $g_v$ , where  $g$  represents a real UK post-code, has a service time  $d_v$  and a time-window in which it must commence described by  $\{e_v, l_v\}$ , i.e. the earliest and latest time at which can start and finish. Visits are grouped into *journeys*, where each journey contains a subset  $V_j$  of the  $V$  visits and is allocated to an employee. Each journey  $j$  starts and ends at the central office. Two modes of travel are available to employees: the first mode

uses private transport (car), the second makes use of available public-transport, encouraging more sustainable travel. The overall goal is to minimise the total distance travelled across all journeys completed and forms the objective function for the problem. However, in addition, discussions with end-users [14] highlights four characteristics of solutions that are of interest:

- The total **emissions** incurred by all employees over all visits
- The total **employee cost** the total cost (based on £/hour) of paying the workforce
- The total **travel cost** the cost of all of the travel activities undertaken by the workforce
- The % of employees using **car travel**

We develop an algorithm based on Map-Elites to minimise the distance objective through projecting solutions onto a 4-dimensional map, with each axis representing one of the above characteristics. Solution quality is compared to an Evolutionary Algorithm that uses exactly the same distance function as an objective, and an identical representation, crossover and mutation operators.

Both the Map-Elites algorithm and the EA use an identical representation of the problem, previously described in [14]. The genotype defines a *grand-tour* [6], i.e. a single permutation of all  $v$  required visits. This is subsequently divided into individual feasible journeys using a *decoder*. The genotype also includes  $v$  additional genes that denote the model of transport to be used for the visit, i.e. public or private.

The decoder converts the single grand tour into a set of journeys to be undertaken by an employee. It examines each visit in the grand tour in order. Initially, the first visit in the grand tour specified by the genotype is allocated to the first journey. The travel mode(car or public transport) associated with this visit in the genome is then allocated to the journey: this travel mode is then adopted for the entire journey (regardless of the information associated with a visit in the genome). The decoder then examines the next visit in the grand tour: this is added to the current journey if it is *feasible*. Feasibility requires that the employee arrives from the previous visit using the mode of transport allocated to the journey within the time window associated with the visit. Note that a travel mode cannot be switched during a journey. Subsequent visits are added using the journey mode until a hard constraint is violated, at which point the current journey is completed and a new journey initiated.

### 3.1 The MAP-Elites Algorithm

The implementation of MAP-Elites used in this paper is given in Algorithm 1 and is taken directly from [7].  $G$  random-solutions are initially generated and mapped to a discrete archive as follows. For each solution  $x'$  a *feature-descriptor*  $b$  is obtained by discretising the four features of interest associated with the solution (Sect. 3) into 20 bins; for 4 dimensions this gives a total of  $20^4 = 160,000$  cells. The upper and lower bounds required for discretisation are taken as the

maximum and minimum values observed by [14] for each dimension during an extensive experimental investigation. A solution is placed in the cell in the archive corresponding to  $b$  if its fitness ( $p$ , calculated as total distance travelled) is better than the current solution stored, or the cell is currently empty. Parents are selected at random from the archive. The *RandomVariation()* method applies either crossover followed by mutation, or just mutation, depending on the experiment. All operators utilised are borrowed from [14]. The *mutation* operator moves a randomly selected entry in the grand-tour to another randomly selected point in the tour. The *crossover* operator selects a random section of the tour from parent-1 and copies it to the new solution. The missing elements in the child are copied from parent-2 in the order that they appear in parent-2.

---

**Algorithm 1.** MAP-Elites Algorithm, taken directly from [7]

---

```

procedure MAP-ELITES ALGORITHM
  ( $\mathcal{P} \leftarrow \emptyset, \mathcal{X} \leftarrow \emptyset$ )
  for iter = 1  $\rightarrow$  I do
    if iter < G then
       $x' \leftarrow \text{randomSolution}()$ 
    else
       $x' \leftarrow \text{randomSelection}(\mathcal{X})$ 
       $x' \leftarrow \text{randomVariation}(\mathcal{X})$ 
    end if
     $b' \leftarrow \text{feature\_descriptor}(x')$ 
     $p' \leftarrow \text{performance}(x')$ 
    if  $\mathcal{P}(b') = \emptyset$  or  $\mathcal{P}(b') < p'$  then
       $\mathcal{P}(b') \leftarrow p'$ 
       $\mathcal{X}(b') \leftarrow x'$ 
    end if
  end for
  return feature-performance map( $\mathcal{P}$  and  $\mathcal{X}$  )
end procedure

```

---

### 3.2 The Evolutionary Algorithm

The EA uses exactly the same representation and operators as the Map-Elites algorithm. The EA uses a population size of 100, with 40 children being created each generation. Each child is created by cloning from one parent or crossover using two parent. Parents are selected using a tournament of size 2. A mutation-rate of 0.7 is applied to each child. The children are added back into the population, replacing the loser of a tournament, providing the child represents an improvement over the loser. The parameters for the EA were derived from the authors' previous experience with similar algorithms applied to the same problem instances.

### 3.3 Problem Instances

We use a set of problem instances based upon the city of London, divided into two problem sets, termed London (60 visits) and BigLondon (110 visits). These instances were first introduced in [14]. Each visit represents a real post-code within London. For each of the problem sets, 5 instances are produced in which the duration of each visit is fixed to 30 min. Visits are randomly allocated to one of  $n$  time-windows, where  $n \in \{1, 2, 4, 8\}$ . For  $n = 1$ , the time-window has a duration of 8 hours, for  $n = 2$ , the time-windows are “9am–1pm” and “1pm–5pm” etc. These instances are labelled using the scheme  $\langle set \rangle - numTimeWindows$ , i.e. *Lon-1* refers to an instance in the London with one time-window and *Blon-2* refers to an instance of the BigLondon problem with 2 time windows. The fifth instance represents a randomly chosen mixture of time windows based on 1,2,4 and 8 h.

If a journey is undertaken by car, paths between visits and distance is calculated according to the real road-network using the GraphHopper library<sup>1</sup>. This relies on Open StreetMap data<sup>2</sup>. Car emissions are calculated as 140 g/km based upon values presented in [12]. For journeys by public-transport, data is read from the Transport for London (TfL) API<sup>3</sup> which provides information including times, modes and routes of travel by bus and train. Public transport emissions factors are based upon those published by TfL [12].

### 3.4 Experimental Parameters

The function evaluation budget is fixed in all experiments. We tests two values: one million evaluations and five million. Each treatment is repeated 10 times on each instance. The best objective (distance) value is recorded for both treatments in each run. We apply Vargha and Delaney’s  $\hat{A}$  statistic [15] to assess difference between the algorithms. This is regarded as a robust test when assessing randomised algorithms. The test returns a statistic,  $\hat{A}$ , that takes values between 0 and 1; a value of 0.5 indicates that the two algorithms are stochastically equivalent, while values closer to 0 or 1 indicate an increasingly large stochastic difference between the algorithms. One of the most attractive properties of the Vargha-Delaney test is the simple interpretation of the  $\hat{A}$  statistic: for results from two algorithms, A and B, then is simply the expected probability that algorithm A produces a superior value to algorithm B. We follow the standard interpretation that a value in the range  $0.5 \pm 0.06$  indicates a small effect,  $0.5 \pm 0.14$  a medium effect and  $.5 \pm 0.21$  a large effect.

In addition we use two metrics to further analyse Map-Elites that are now de-facto in the literature:

- *Coverage* represents the area of the feature-space covered by a single run of the algorithm, i.e. the number of cells filled. For a single run  $x$  of algorithm

<sup>1</sup> <https://graphhopper.com/>.

<sup>2</sup> <https://openstreetmap.org/>.

<sup>3</sup> <https://api.tfl.gov.uk/>.



$y$ ,  $coverage = noOfCellsFilled/C_{Max}$  where  $C_{Max}$  is the total number of cells filled by combining all runs of any algorithm on the problem under consideration.

- *Precision* is also defined as *opt-in reliability*: if a cell is filled in a specific run, then the cell-precision is calculated as the inverse of the performance-value (distance) found in the that cell in that run, divided by the best-value ever obtained for cell in any run of any algorithm (as this is minimisation). Cell-precision is averaged over all *filled* cells in an archive to give a single precision value for a run.

From the perspective of a planner, this represents the choice of solutions available to them, while precision indicates whether a cell contains a solution that is likely to of potential use to the planner. The averaged precision for a run indicates the overall quality of the solutions produced.

## 4 Results

The first research question aims to compare the performance of MAP-Elites and EA algorithms under different evaluation budgets to determine whether MAP-Elites might be useful in producing a set of acceptable solutions quickly. Two values are tested : the first is relatively small with 1 million evaluations (as in [14]); the second increases this to 5 million.

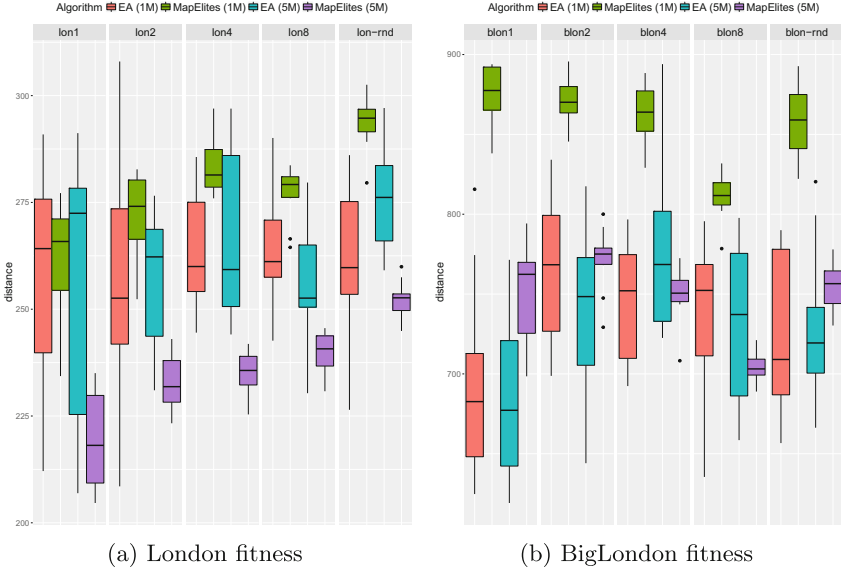
Figure 1(a,b) show the objective fitness values achieved by ME and the EA under both budgets on each of the problem instances. Table 1 shows effect size and direction according to the Vargha-Delaney metric.

**Table 1.** Comparison of Map-Elites (ME) to Evolutionary Algorithm (EA) at  $n$  million evaluations. Arrows show Vargha-Delaney A test effect size and direction

	London problems					Big London problems				
	Lon-1	Lon-2	Lon-4	Lon-8	Lon-rnd	Blon-1	Blon-2	Blon-4	Blon8	Blon-rnd
ME(1M) vs EA(1M)	↔	↓↓↓	↓↓↓	↓↓↓	↓↓↓	↓↓↓	↓↓↓	↓↓↓	↓↓↓	↓↓↓
ME(5M) vs EA(5M)	↑↑↑	↑↑↑	↑↑↑	↑↑↑	↑↑↑	↓↓↓	↓↓	↑	↑	↓↓↓

We note firstly that for 1M evaluations for both sets of problems, the EA outperforms Map-Elites: the median of the EA is lower than ME, and the effect size is large in each case. However, when the budget is increased to 5M, Map-Elites outperforms the EA on all of the smaller problems with a large effect size; it also outperforms the EA on two of the larger problems, although the effect size is small. In the remaining 3 cases, the EA still wins.

Note that the Fig. 1a and b only show performance in terms of distance and do not take into account the four characteristics which provide insight to the end-users. These values are given in Table 2. Firstly we note that for the smaller *lon*



**Fig. 1.** Performance of MAP elites and the EA with budgets of 1 million and 5 million evaluations.

problems, the best-value for each characteristic is obtained from the MAP-Elites algorithm in call cases. This includes *lon* – 8 in which the best objective value for a solution is obtained by the EA, but the solution has poorer values for each of the 4 characteristics than the best solution obtained by MAP-Elites. Examining the results for the larger BLon problem demonstrates that MAP-Elites, despite a sub-optimal performance (w.r.t the objective function), can still find solutions that out perform the EA in terms of the individual characteristics.

### 4.1 Coverage and Precision

The coverage metric evaluates the ability of an individual run of an algorithm to place individuals in each of the cells. Note that it is possible that some of the cells cannot be filled in because the characteristics of that instance do not allow a feasible solution in that area.

The coverage achieved is displayed in Fig. 2a and b. Observe that coverage of over 70% is common with MAP-Elites, but the EA gives very poor coverage as it converges to a single solution. In real-world terms, the EA leaves the user with little choice of solution and no insight into the problem.

Figure 2c and d show the precision achieved by MAP Elites and the EA. We note that the highest precision achieved by the EA outperforms MAP Elites. Recall that precision is calculated over only those cells that are filled. The EA allocates all of its evaluations to very few cells, and thus find good solutions for those cells. In contrast, MAP-Elites has to distribute the same budget of

**Table 2.** A detailed comparison of the best results found over 10 runs for performance (distance) and the 4 characteristics associated with the solutions, based on an evaluation budget of 5 million for each run. Values are shown for MAP elites on the left and the EA on the right.

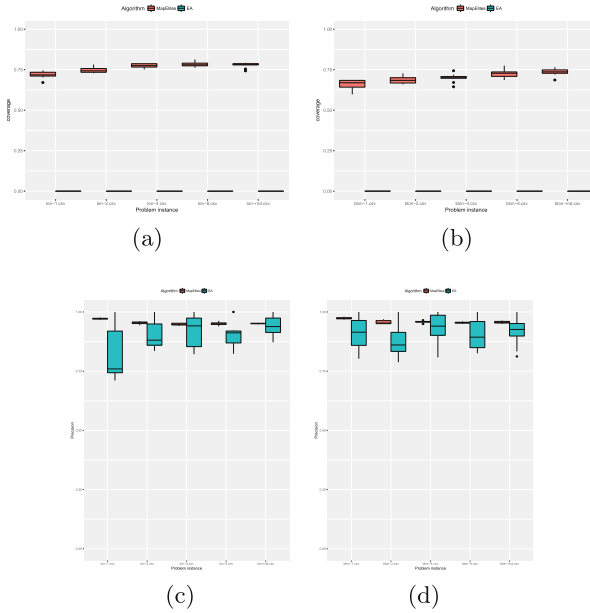
	Dist	StaffCost	TravelCost	CO2	CarUse
Lon-1	<b>204.64</b> : 206.93	<b>841</b> : 974.67	<b>82.54</b> : 85.79	<b>133.83</b> : 163.75	<b>0</b> : 0.25
Lon-2	<b>223.3</b> : 231.02	<b>870.67</b> : 1014.67	<b>89.71</b> : 103.04	<b>148.94</b> : 192.85	<b>0.06</b> : 0.33
Lon-4	<b>225.37</b> : 244.09	<b>904.33</b> : 1276	<b>94.63</b> : 116.74	<b>158.77</b> : 194.59	<b>0.04</b> : 0.33
Lon-8	230.8 : <b>230.34</b>	<b>967.33</b> : 1376.67	<b>103.5</b> : 140.1	<b>159.07</b> : 240.54	<b>0.04</b> : 0.35
Lon-Rnd	<b>244.91</b> : 259.11	<b>944</b> : 1140.33	<b>99.48</b> : 107.4	<b>155.17</b> : 216.53	<b>0.04</b> : 0.33
Blon-1	698.48 : <b>619.15</b>	<b>1987</b> : 2182.33	222.63 : <b>207.17</b>	527.27 : <b>506.02</b>	<b>0.04</b> : 0.25
Blon-2	729.21 : <b>644.07</b>	<b>2107.67</b> : 2385.67	244.54 : <b>243.55</b>	584.99 : <b>581.16</b>	<b>0.07</b> : 0.32
Blon-4	<b>708.25</b> : 722.53	<b>2183.33</b> : 2545.67	<b>267.85</b> : 272.34	<b>584.19</b> : 637.26	<b>0.08</b> : 0.33
Blon-8	688.94 : <b>658.52</b>	<b>2209</b> : 2772	<b>272.22</b> : 311.52	<b>586.81</b> : 637.5	<b>0.08</b> : 0.38
Blon-rnd	730.3 : <b>666.29</b>	<b>2256</b> : 2717.67	<b>251.31</b> : 263.1	<b>580.16</b> : 602.47	<b>0.09</b> : 0.36

evaluations across a much larger number cells, making it hard to always find a high-performing solution in each cell. In addition, many of the low-precision scores for MAP-Elites occur when one run does not find as high-performing a solution in a cell as another run of MAP-Elites. Running MAP-Elites for more evaluations would likely improve precision (without danger of convergence due to its propensity to enforce diversity).

## 4.2 Gaining Insight into the Problem Domain

Figure 3 plots the cells, and the elite solutions contained, for each 2-dimensional pairing of the 4 dimensions. Although the archive could be drawn in 4-dimensions, discussion with users suggested that presenting 2-dimensional maps provides more insight. Within each plot, each cell that is occupied is coloured to represent the distance objective value of the elite solution - lowest (best) values being green, highest being red. Note that most of the cells have a solution within them. Where there is an area with no solutions it tends to be at a corner of the plot. For instance, there are a lack of solutions with low  $CO_2$  and high travel costs (Fig. 3e) or high car use and low  $CO_2$  (Fig. 3a). From a planning perspective, Fig. 3 indicates (1) combinations of objectives that have no feasible solutions, and (2) quality of feasible solutions.

MAP-Elites tends to cover a larger part of the solution space. A common trend is that the solutions that are better in terms of one or two of the four characteristics are not always solutions that exhibit the lowest distance objective. The map also quantifies trade-offs in objective value: for example, the extent to which increased car use increases  $CO_2$  compared to options that utilise more public transport. Another insight to be gained is the effects of higher public transport use (i.e. low car use) and staff cost: staff costs rise as public transport



**Fig. 2.** Coverage and precision for map-elites and the EA on both problem sets

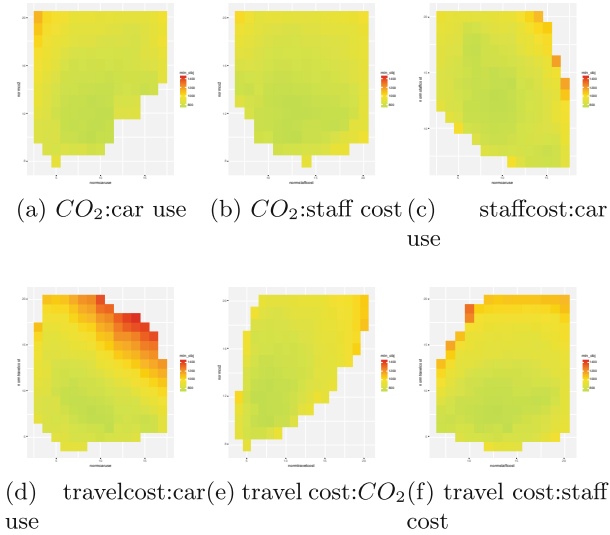
usage increases (Fig. 3a and c). This is due to the longer journey times experienced with public transport leading to increased working hours for staff.

A planner with responsibility for determining policies regarding staff scheduling may make use of the diagrams in Fig. 3c to understand what solutions are possible given a specific priority. For instance, if it is determined that reducing  $CO_2$  is a priority then they can determine what possible trade-offs exist for low  $CO_2$  solutions. Where a balance is required (i.e. lowering  $CO_2$  but also keeping financial costs in check) MAP-Elites allows the planner to find compromise solutions that are not optimal in any single dimension, but may prove useful when meeting multiple organisational targets or aspirations.

## 5 Conclusions

In this paper we have applied MAP-Elites to a real world combinatorial optimisation problem domain—a workforce scheduling and routing problem. Unlike previous applications of MAP-Elites that have tended to concentrate on design problems, WSRP is an example of a repetitive problem, requiring an optimisation algorithm to find acceptable solutions in a short period of time. In addition to an acceptable solution however, a user also requires choice, in being able to select potential solutions based on additional criteria of relevance to a particular company.

With reference to the research questions in Sect. 1, we note that MAP-Elites tends to require a larger evaluation budget to produce results that are com-



**Fig. 3.** Maps produced from a single run of the blon-1 problem: rather than display the single 4-dimensional map produced from map-elites, we display the data as all possible pairings of the 4 characteristics (Color figure online)

parable with a straightforward EA for the problems tested. However, for small problems, affording a larger evaluation budget to Map-Elites enables it to discover improved solutions, compared to the EA. For larger problems, although our results show that MAP-Elites cannot outperform the EA in terms of objective performance, it does find solutions that outperform the EA in terms of the individual characteristics. It is likely that running MAP-Elites for longer would continue to improve its performance, without risking convergence. The increased cpu-time required for such a budget may be easily obtained through the use of multi-core desktop computers or cloud based resources in a practical setting. We also note that the illumination aspect of MAP-Elites may aid the ability of planners to understand the factors that lead to good solutions and subsequently influence policy planning/determine choices based on organisational values, and that this aspect is of considerable benefit. Illumination of the solution-space also provides additional insight to planners, who can gain understanding into the influence of different factors on the overall cost of a solution.

Future work will focus on further exploration of the relationship between objective quality and function evaluations, to gain insight into the anytime performance of Map-Elites, for use in a real-world setting. The granularity of the archive clearly influences performance and should be investigated by depth. Finally, an additional comparison to multi-objective approaches is also worth pursuing — while this may improve solution quality however it is unlikely to offer the same insight into the entire search-space.

## References

1. Bertels, S., Fafle, T.: A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Comput. Oper. Res.* **33**(10), 2866–2890 (2006)
2. Braekers, K., Hartl, R.F., Parragh, S.N., Tricoire, F.: A bi-objective home care scheduling problem: analyzing the trade-off between costs and client inconvenience. *Eur. J. Oper. Res.* **248**(2), 428–443 (2016)
3. Castillo-Salazar, J.A., Landa-Silva, D., Qu, R.: A survey on workforce scheduling and routing problems. In: *Proceedings of the 9th international conference on the practice and theory of automated timetabling*, pp. 283–302 (2012)
4. Cully, A., Clune, J., Tarapore, D., Mouret, J.B.: Robots that can adapt like animals. *Nature* **521**(7553), 503 (2015)
5. Hart, E., Sim, K., Urquhart, N.: A real-world employee scheduling and routing application. In: *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 1239–1242. ACM (2014)
6. Laporte, G., Toth, P.: Vehicle routing: historical perspective and recent contributions. *EURO J. Transp. Logistics* **2**(1–2), 1–4 (2013)
7. Mouret, J., Clune, J.: Illuminating search spaces by mapping elites. *CoRR* (2015)
8. Pugh, J.K., Soros, L.B., Stanley, K.O.: Quality diversity: a new frontier for evolutionary computation. *Front. Robot. AI* **3**, 40 (2016)
9. Pugh, J.K., Soros, L.B., Stanley, K.O.: Searching for quality diversity when diversity is unaligned with quality. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) *PPSN 2016. LNCS*, vol. 9921, pp. 880–889. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_82](https://doi.org/10.1007/978-3-319-45823-6_82)
10. Smith, D., Tokarchuk, L., Wiggins, G.: Rapid phenotypic landscape exploration through hierarchical spatial partitioning. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) *Parallel Problem Solving from Nature - PPSN XIV*, pp. 911–920. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_85](https://doi.org/10.1007/978-3-319-45823-6_85)
11. Tarapore, D., Clune, J., Cully, A., Mouret, J.B.: How do different encodings influence the performance of the map-elites algorithm? In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO 2016*, pp. 173–180. ACM, New York (2016)
12. TFL: Travel in london: Key trends and developments. Technical report Transport for London (2009)
13. Urquhart, N., Fonzona, A.: Evolving solution choice and decision support for a real-world optimisation problem. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017*, pp. 1264–1271. ACM (2017)
14. Urquhart, N.B., Hart, E., Judson, A.: Multi-modal employee routing with time windows in an urban environment. In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*. pp. 1503–1504. ACM (2015)
15. Vargha, A., Delaney, H.D.: A critique and improvement of the cl common language effect size statistics of McGraw and Wong. *J. Educ. Behav. Stat.* **25**(2), 101–132 (2000)
16. Vassiliades, V., Chatzilygeroudis, K., Mouret, J.B.: Using centroidal voronoi tessellations to scale up the multi-dimensional archive of phenotypic elites algorithm, p. 1. August 2017



# Prototype Discovery Using Quality-Diversity

Alexander Hagg<sup>1,2</sup>(✉), Alexander Asteroth<sup>1</sup>, and Thomas Bäck<sup>2</sup>

<sup>1</sup> Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany  
{alexander.hagg,alexander.asteroth}@h-brs.de

<sup>2</sup> Leiden Institute of Advanced Computer Science,  
Leiden University, Leiden, The Netherlands  
t.h.w.baeck@liacs.leidenuniv.nl

**Abstract.** An iterative computer-aided ideation procedure is introduced, building on recent quality-diversity algorithms, which search for diverse as well as high-performing solutions. Dimensionality reduction is used to define a similarity space, in which solutions are clustered into classes. These classes are represented by prototypes, which are presented to the user for selection. In the next iteration, quality-diversity focuses on searching within the selected class. A quantitative analysis is performed on a 2D airfoil, and a more complex 3D side view mirror domain shows how computer-aided ideation can help to enhance engineers' intuition while allowing their design decisions to influence the design process.

**Keywords:** Ideation · Quality-diversity · Prototype theory  
Dimensionality reduction

## 1 Introduction

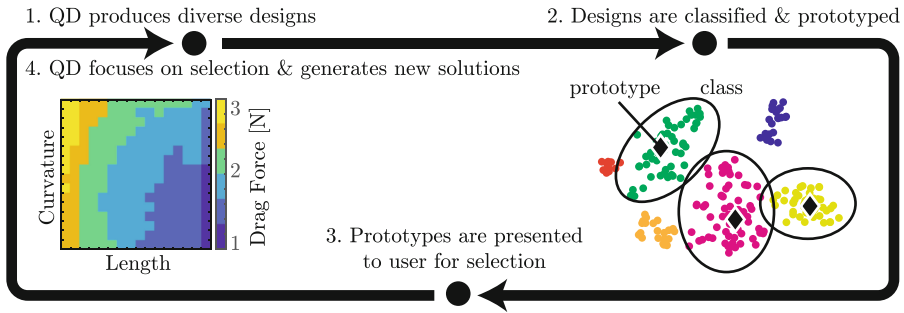
Conceptual engineering design is an iterative process [5]. Under the paradigm commonly called ideation [3] a design problem is defined, the design space explored, candidate solutions evaluated, and finally design decisions are taken, which put constraints onto the next design iteration.

In a 2014 interview study by Bradner [3] on the real-world usage of automation in design optimization, “participants reported consulting Pareto plots iteratively in the conceptual design phase to rapidly identify and select interesting solutions”. This process of *a posteriori articulation of preference* [9] is described by the “design by shopping” paradigm [1, 23]. A Pareto front of optima is created by a multi-objective optimization algorithm, after which engineers choose a solution to their liking. That participants used optimization algorithms to develop preliminary solutions to solve a problem surprised the interviewers.

Design optimization has been applied to multi-modal problems, using niching and crowding to enforce diversity in evolutionary optimization algorithms [21, 22]. For optimization algorithms to operate effectively in cases where evaluation of designs is computationally expensive, surrogate assistance is applied, using

predictive models that replace most of the evaluations [10]. Recently introduced quality-diversity (QD) algorithms, like NSLC [11] and MAP-Elites [13], are evolutionary algorithms capable of producing a large array of solutions constituting different behaviors or design features. Surrogate assistance was introduced for QD algorithms [6] as well. It enables finding thousands of designs, using orders of magnitude fewer evaluations than running MAP-Elites without a surrogate. However, this large number of solutions can hinder the engineer’s ability to select interesting designs.

As automated diversity gives too many solutions, their more concise presentation makes QD more useful to designers. In this paper we apply the design by shopping paradigm to QD, assisting design decisions by representing similar solutions succinctly with a representative solution using prototype theory. Therein, objects are part of the same class based on resemblance. Wittgenstein [26] questioned whether classes can even be rigidly limited, implying that there is such a thing as a distance to a class. Rosch later introduced prototype theory [18], stating that natural classes consist of a prototype, the best representative of its class, and non-prototypical examples, which can be ranked in terms of distance to the prototype. However, while feature diversity is enforced, surrogate-assisted QD uses no metric for genotypic similarity in terms of the actual design space.



**Fig. 1.** Computer-aided ideation loop. Step 1: QD algorithm is used to discover diverse optimal solutions. Step 2: similar solutions are grouped into classes. Step 3: prototypes are visualized to allow the engineer to select the prototype they want to further explore. Step 4: QD focuses on the user’s selection to generate further solutions.

By applying prototype theory to the variety of designs produced by QD algorithms, *computer-aided ideation* (CAI) is introduced (Fig. 1), allowing the same a posteriori articulation of preference as in the design by shopping paradigm. Although performance and diversity can both be formally described and optimized, design decisions are based on the intuition of the engineer, and cannot be automated. QD is used to discover a first set of optimal solutions. Then, by clustering similar solutions into classes and representative prototypes, the optimization process is guided by extracting seeds from the classes selected by the user, zooming in on a particular region in design space.



QD allows a paradigm shift in optimal engineering design, but integration of QD algorithms into the ideation process has yet to be studied extensively. In this work we introduce a CAI algorithm that takes advantage of recently introduced QD algorithms [11, 13]. Prototype Discovery using Quality-Diversity (PRODUQD) [prə'dakt], which performs a representative selection of designs, enables engineers to make design decisions more easily and influence the search for optimal solutions. PRODUQD can find solutions similar to a selection of prototypes that perform similarly well as solutions that were found by searching the entire design space with QD. By integrating QD algorithms and ideation a new framework for design is created; a paradigm which uses optimization tools to empower human intuition rather than replace it.

## 2 Related Work

### 2.1 Quality-Diversity and Surrogate Assistance

QD algorithms, like Novelty Search with Local Competition (NSLC) [11] and Multidimensional Archive of Phenotypic Elites (MAP-Elites) [13], use a low-dimensional behavior or feature characterization, such as neural network complexity or curvature of a design, to determine similarity between solutions [16]. Solutions compete locally in feature space, superseding similar yet less fit solutions. In MAP-Elites, the feature space consists of a discrete grid of behavior or feature dimensions, called a feature map. Every bin in the map is either empty or holds a solution, called an elite, that is currently the best performing one in its niche. QD is able to produce many solutions with a diverse set of behaviors and is very similar to the idea of Zwicky's morphological box [27] as it allows new creations by combining known solution configurations. QD algorithms perform many evaluations, making them unsuitable for design problems that need computationally expensive or real world evaluation.

To decrease the number of expensive objective evaluations, approximative surrogate models replace the objective function close to optimal solutions using appropriate examples [10]. To sample the design space effectively and efficiently, Bayesian Optimization is used. Given a prior over the objective function, evidence from known samples is used to select the next best observation. This decision is based on an acquisition function that balances exploration of the design space, sampling from unknown regions, and exploitation, choosing samples that are likely to perform well. This way, the surrogate model becomes more accurate in optimal regions during sampling. The most common surrogates used are Gaussian Process (GP) regression models [17].

Surrogate assistance has been applied to QD with Surrogate-Assisted Illumination (SAIL) [6]. In SAIL the GP model is pretrained with solutions evenly sampled in the parameter space with a Sobol sequence [14]. The sequence allows iteratively finer sampling, approximating a uniform distribution. Then, using the upper confidence bound (UCB) acquisition function, an *acquisition map* is created, containing a diverse set of candidate training samples. UCB, described by the function  $UCB(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x})$ , is a balance between exploitation

( $\mu(\mathbf{x})$ , the mean prediction of the model), and exploration ( $\sigma(\mathbf{x})$ , the model’s uncertainty), parameterized by  $\kappa$ .

The acquisition map is first seeded with the previously acquired samples, assigning them to empty bins or replacing less performant solutions. MAP-Elites is then used in conjunction with the GP model to fill and optimize the acquisition map, using UCB as a fitness function and combining existing solutions from bins, illuminating the surrogate model through the “lens” of feature map. A candidate sample is created for every bin in the map. Then, the acquisition map is sampled using a Sobol sequence and selected solutions are evaluated using the expensive evaluation function. After gathering a given number of samples, the acquisition function is adapted by removing the model’s uncertainty and the final prediction map, seeded with the set of known samples, is illuminated, producing a discrete map of diverse yet high-performing solutions.

## 2.2 Dimensionality Reduction

Clustering depends on a notion of distance between points. The curse of dimensionality dictates that the relative difference of the distances of the closest and farthest data points goes to zero as the dimensionality increases [2]. Clustering methods using a distance metric break down and cannot differentiate between points belonging to the same or to other clusters [25]. Dimensionality reduction (DR) methods are applied to deal with this problem. Data is often located at or close to a manifold of lower dimension than the original space. DR transforms the data into a lower-dimensional space, enabling the clustering method to better distinguish clusters [25]. Common DR methods are Principal Component Analysis (PCA) [15], kernel PCA (kPCA) [20], Isomap [24], Autoencoders [8] and t-distributed Stochastic Neighbourhood Embedding (t-SNE) [12].

t-SNE is commonly used for visualization and has been shown to be capable of retaining the local structure of the data, as well as revealing clusters at several scales. It does so by finding a lower-dimensional distribution of points  $\mathbb{Q}$  that is similar to the original high-dimensional distribution  $\mathbb{P}$ . The similarity of datapoint  $\mathbf{x}_j$  to datapoint  $\mathbf{x}_i$  is the conditional probability ( $p_{j|i}$  for  $\mathbb{P}$  and  $q_{j|i}$  for  $\mathbb{Q}$ , Eq. 1), that  $\mathbf{x}_i$  would pick  $\mathbf{x}_j$  as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian distribution centered at  $\mathbf{x}_i$ . The Student-t distribution is used to measure similarities between low-dimensional points  $\mathbf{y}_i \in \mathbb{Q}$  in order to allow dissimilar objects to be modeled far apart in the map (Eq. 1).

$$p_{j|i} = \frac{e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}{\sum_{k \neq i} \left( e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}} \right)}, \quad q_{j|i} = \frac{1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}} \quad (1)$$

The local scale  $\sigma_i$  is adapted to the density of the data (smaller in denser parts).  $\sigma_i$  is set such that *perplexity* of the conditional distribution equals a predefined value. The perplexity of a distribution defines how many neighbors for each data

point have a significant  $p_{j|i}$  and can be calculated using the Shannon entropy  $H(P_i)$  of the distribution  $P_i$  around  $x_i$  (Eq. 2).

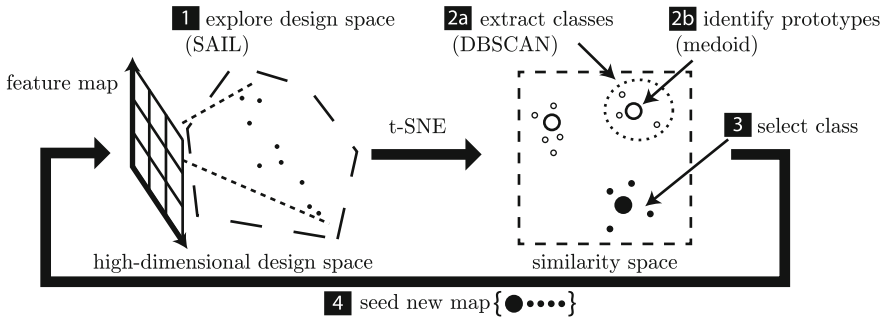
$$\text{Perp}(P_i) = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}} \tag{2}$$

$$KL(\mathbb{P}||\mathbb{Q}) = \sum_{i \neq j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \tag{3}$$

Using the bisection method,  $\sigma_i$  are changed such that  $\text{Perp}(P_i)$  approximates the preset value (commonly 5–50). The similarity of  $x_j$  to  $x_i$  and  $x_i$  to  $x_j$  is absorbed with the joint probability  $p_{ij}$ . A low-dimensional map is learned that reflects all similarities  $p_{ij}$  as well as possible. Locations  $y_i$  are determined by iteratively minimizing the Kullback-Leibler divergence of the distribution  $\mathbb{Q}$  from the distribution  $\mathbb{P}$  (Eq. 3) with gradient descent.

### 3 Prototype Discovery Using Quality-Diversity

PRODUQD is an example of a CAI algorithm (Fig. 2). Initially, the design space is explored with a QD algorithm. SAIL [6] is used to explore the design space, creating high-performing examples of designs with varying features. These features can be directly extracted from design metrics, for instance the amount of head space in a car. SAIL produces a prediction map that contains a set of diverse high-performing solutions.



**Fig. 2.** PRODUQD cycle - steps as in Fig. 1. Step 1: the design space is explored with the goal of filling the feature map. Step 2: (a) classes are extracted in a low-dimensional similarity space, and (b) prototypes are defined. Step 3: a selection is made. Step 4: seeds are extracted for the next iteration.

A similarity space is constructed using t-SNE. In this space, similar solutions are clustered into classes. Since no prior knowledge on the structure of optimality in design space is available and due to the stochastic nature of QD, the number

and density of clusters is unknown. To group the designs into clusters we use the well-known density based clustering algorithm DBSCAN [4]<sup>1</sup>.

A prototype is then extracted for every class. According to prototype theory [18], prototypes are those members of a class, “with the most attributes in common with other members of the class and least attributes in common with other classes”. The most representative solution of a class is the member of a cluster that has the minimum distance to other members. The medoid is taken rather than a mean of the parameters, as this mean could lie in non-optimal or even invalid regions of the design space.

The prototypes are presented to the user, offering them a concise overview of the diversity in the generated designs. After the user selects one or more prototypes, the affiliated class members are used as seeds for the next SAIL iteration, serving as initial solutions in the acquisition and prediction maps. Initializing SAIL with individuals from the chosen class forces SAIL to start its search within the class boundaries. Using a subset of untested solutions of a particular class stands in contrast to SAIL, which focuses on searching the entire design space, seeding both maps with actual samples. Within each SAIL iteration, the GP surrogate is retrained whenever new solutions are evaluated.

A precise definition of PRODUQD can be found in Algorithm 1, including the use of the selected seeds  $\mathcal{S}$  in SAIL. This ideation process explores the design space while taking into account on-line design decisions.

---

**Algorithm 1.** Prototype Discovery using Quality-Diversity (PRODUQD)

---

<p><math>\mathcal{X} \leftarrow \text{Sobol}_{1:G}, \mathcal{Y} \leftarrow PE(\mathcal{X}), \mathcal{S}_0 \leftarrow \mathcal{X}</math>  <math>\triangleright PE = \text{precise eval.}, \mathcal{S}_0 = \text{initial seed}</math>  <b>for</b> iter = 1 <math>\rightarrow PE\_budget</math> <b>do</b>              <b>[1] Explore Design Space</b>              <math>(\mathcal{X}_{pred}, \mathcal{Y}_{pred}) = \text{SAIL}(\mathcal{X}, \mathcal{S}_{iter-1})</math>              <b>[2] Extract Classes</b>              <math>\mathcal{X}_{red} = \text{T-SNE}(\mathcal{X})</math>              <math>C = \text{DBSCAN}(\mathcal{X}_{red})</math>                  <math>\triangleright C = \text{class assignments}</math>              <b>[3] Determine Prototypes</b>              <b>for</b> j = 1 <math>\rightarrow  C </math> <b>do</b>                  <math>\mathcal{P} \leftarrow \text{MEDOID}(x_{red}, c_j)</math>              <b>end for</b>              <b>[4] Select Prototype(s)</b>              <math>p_{sel} = \text{SELECT}(\mathcal{P})</math>                  <math>\triangleright p_{sel} = \text{user selected prototype}</math>              <b>[5] Extract Seeds</b>              <math>S = \mathcal{X}, x \in c_{sel}</math>                  <math>\triangleright c_{sel} = \text{class belonging to } p_{sel}</math>              <b>end for</b></p>	<p><math>\triangleright</math> Surrogate-Assisted Illumination  <b>function</b> SAIL(<math>\mathcal{X}, \mathcal{S}</math>) <math>\triangleright</math> samples, seeds              <b>[1] Produce Acquisition Map</b>              <b>for</b> iter = 1 <math>\rightarrow PE\_budget</math> <b>do</b>                  <math>\mathcal{D} \leftarrow (\mathcal{X}, \mathcal{Y})</math> <math>\triangleright</math> observation set                  <math>acq() \leftarrow \text{UCB}(GP\_model(\mathcal{D}))</math>                  <math>(\mathcal{X}_{acq}, \mathcal{Y}_{acq}) = \text{MAP-E.}(acq, \mathcal{S})</math>                      <math>\triangleright \text{MAP-E.} = \text{MAP-Elites}</math>                  <math>\mathbf{x} \leftarrow \mathcal{X}_{acq}(\text{Sobol}_{iter})</math>                      <math>\triangleright</math> Select from acquisition map                  <math>\mathcal{X} \leftarrow \mathcal{X} \cup \mathbf{x}, \mathcal{S} \leftarrow \mathcal{S} \cup \mathbf{x}</math>                  <math>\mathcal{Y} \leftarrow \mathcal{Y} \cup PE(\mathbf{x})</math>              <b>end for</b>              <b>[2] Produce Prediction Map</b>              <math>\mathcal{D} \leftarrow (\mathcal{X}, \mathcal{Y})</math>              <math>\mathcal{GP} \leftarrow GP\_model(\mathcal{D})</math>              <math>pred() \leftarrow \text{mean}(\mathcal{GP}(x))</math>              <math>(\mathcal{X}_{pred}, \mathcal{Y}_{pred}) = \text{MAP-E.}(pred, \mathcal{S})</math>              <b>return</b> <math>(\mathcal{X}_{pred}, \mathcal{Y}_{pred})</math>              <b>end function</b></p>
---	--

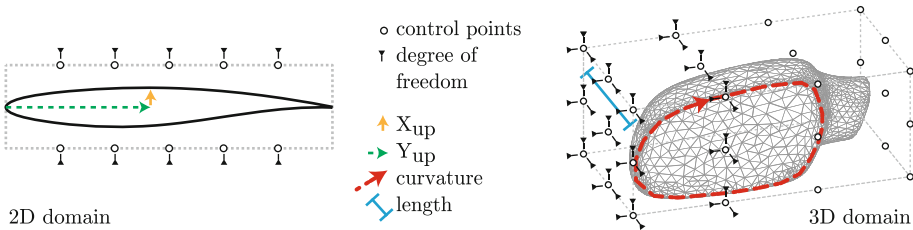
---

<sup>1</sup> DBSCAN’s parameterization is automated using the L Method [19].

## 4 Evaluation

PRODUQD is a tool for CAI which allows the optimization and exploration of QD to be focused to produce designs which resemble user-chosen prototypes. We show that PRODUQD creates solutions of comparable performance to SAIL, produces models with the same level of accuracy, while directing the search towards design regions chosen by the user.

**2D Domain.** PRODUQD and SAIL are applied to a classic design problem, similar to [6], but with a different representation. A high-performing 2D airfoil is optimized using free form deformation, with 10 degrees of freedom (Fig. 3). The base shape, an RAE2822 airfoil, is evaluated in XFOIL<sup>2</sup>, at an angle of attack of 2.7° at Mach 0.5 and Reynolds number 10<sup>6</sup>.



**Fig. 3.** Left: 2D airfoil with control points and features ( $X_{up}$ ,  $Y_{up}$ ), right: control points of 3D mirror representation and features (curvature and length).

The objective is to find diverse deformations, minimizing the drag coefficient  $c_D$  while keeping a similar lift force and area, described by  $\text{fit}(x) = \text{drag}(x) \times p_{c_L}(x) \times p_A(x)$ ,  $\text{drag} = -\log(c_D(x))$ ,  $A = \text{area}$  and Eqs. 4–5. The feature map, consisting of the x and y coordinates of the highest point on the foil ( $X_{up}$  and  $Y_{up}$ , see Fig. 3), is divided into a  $25 \times 25$  grid.

$$p_{c_L}(x) = \begin{cases} \frac{c_L(x)^2}{c_{L0}}, & c_L(x) < c_{L0} \\ 1, & \text{otherwise} \end{cases} \quad (4)$$

$$p_A(x) = \left(1 - \frac{|A - A_0|}{A_0}\right)^7 \quad (5)$$

**3D Domain.** To showcase CAI on a more complex domain, the side mirror of the DrivAer [7] car model is optimized with a 51 parameter free form deformation (Fig. 3 (right)). The objective is to find many diverse solutions while minimizing the drag force (in Newtons) of the mirror. The numerical solver OpenFOAM<sup>3</sup> is

<sup>2</sup> <http://web.mit.edu/drela/Public/web/xfoil/>.

<sup>3</sup> <https://openfoam.org>, simulation at 11 m/s.

used to determine flow characteristics and calculate the drag force. The feature map, consisting of the curvature of the edge of the reflective part of the mirror and the length of the mirror in flow direction, is divided into a  $16 \times 16$  grid.

**Choice of Dimensionality Reduction Technique.** Various DR methods are analyzed as to whether they improve the clustering behavior of DBSCAN compared to applying clustering on the original dimensions.  $\tilde{G}_+$ , a measure of the discordance between pairs of point distances and is robust w.r.t. differences in dimensionality [25], is used as a metric. It indicates whether members of the same cluster are closer together than members of different clusters. A low value ( $\geq 0$ ) indicates a high quality of clustering. PCA, kPCA, Isomap, t-SNE<sup>4</sup> and an Autoencoder are compared using DBSCAN on the latent spaces. t-SNE has been heavily tested for a dimensionality reduction to two dimensions. To allow a fair comparison, the same reduction was performed with the alternative algorithms.

**Table 1.** Quality of DR methods. Variance of the Autoencoder in parentheses.

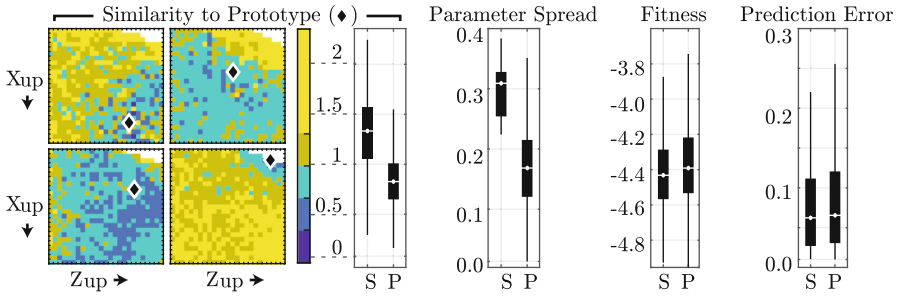
	Original	PCA	kPCA	Isomap	t-SNE	Autoencoder
Avg. $G_+$ score	0.36	0.33	0.22	0.30	<b>0.05</b>	0.454 (0.17)
Avg. number of clusters	4	5	7	4	<b>10</b>	4 (1.23)

SAIL is performed 30 times on the 2D airfoil domain. For every run of SAIL, the dimensionality of the resulting predicted optima is reduced with the various methods and the optima are clustered with DBSCAN. Table 1 shows that t-SNE allows DBSCAN to perform about an order of magnitude better than using other methods. Although t-SNE is not a convex method, it shows no variance, indicating that the method is quite robust. The number of clusters found is about twice as high as using other methods, and since the cluster separation is of higher quality, t-SNE is selected as a DR method in the rest of this evaluation.

**Quantitative Analysis.** To show PRODUQD’s ability to produce designs based on a chosen prototype, it is replicated five times, selecting a different class in each run. In every iteration of PRODUQD 10 iterations of SAIL are run to acquire 100 new airfoils. The first iteration starts with an initial set of 50 samples from a Sobol sequence. Then, the five classes containing the largest number of optima are selected, and the algorithm is continued in separate runs for each class. After each iteration, the we select the prototype that is closest to the one that was selected in the first iteration. PRODUQD runs are compared to the original SAIL algorithm, using the same number of samples, a total of 500.

The similarity of designs to prototypes of optima found in four separate runs, selecting a different prototype in each one, are shown in Fig. 4 (left). The usage

<sup>4</sup> Perplexity is set to 50, but at most half the number of samples.

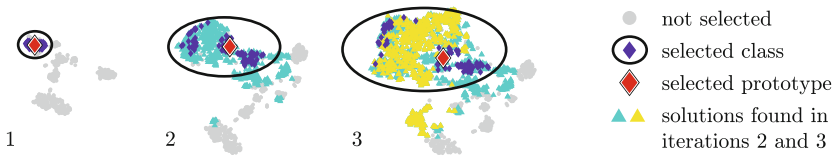


**Fig. 4.** PRODUQD (P) produces designs that are more similar to the selected prototype than using SAIL (S), which is also visible in the smaller parameter spread. The produced designs have similar performance compared to SAIL’s and the surrogate model is equally accurate. Left: final prototype similarity of four different PRODUQD prediction maps.

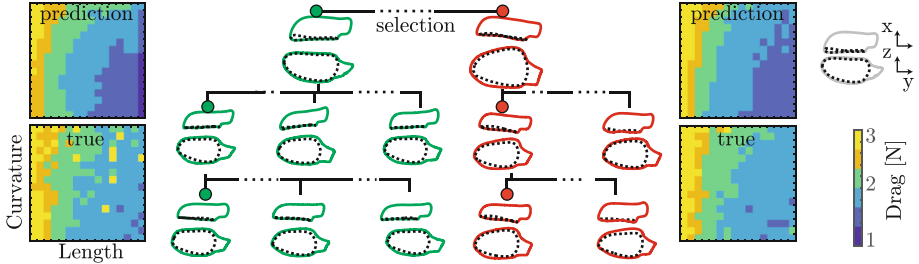
of seeds does not always prevent the ideation algorithm of finding optima outside of the selection, but PRODUQD produces solutions that are more similar to the selected prototype than SAIL. The parameter spread in solutions found with PRODUQD is lower than with SAIL. Yet the true fitness scores and surrogate model prediction errors of both SAIL and PRODUQD are very similar.

Figure 5 shows the similarity space of three consecutive iterations. The effect of selection, zooming in on a particular region, can be seen by the fact that later iterations cover a larger part of similarity space, close to the prototype that was selected. Some designs still end up close to non-selected classes (in gray), which cannot be fully prevented without using constraints. PRODUQD is able to successfully illuminate local structure of the objective function around a prototype. It finds optima within a selected class of similar fitness to optima found in SAIL using no selection, and is able to represent the solutions in a class in a more concise way, using prototypes as representatives, shown by the decreased variance within classes (Fig. 4).

The performance of PRODUQD’s designs is comparable to SAIL while directing the search towards design regions chosen by the user.



**Fig. 5.** The region around the selected class is enlarged in similarity space and structure is discovered as more designs are added in later iterations. In each iteration the feature map is filled with solutions from the selected class.



**Fig. 6.** Phylogenetic tree of two PRODUQD runs diverging after first iteration, and predicted drag force maps (ground truth values are shown underneath).

**Qualitative Analysis.** A two-dimensional feature map, consisting of the curvature and the length of the mirror in flow direction (Fig. 3), is illuminated from an initial set of 100 car mirror designs from a Sobol sequence. After acquiring 200 new samples with SAIL, a prediction map is produced and from this set of solutions the two prototypes having the greatest distance to each other are selected and PRODUQD is continued in two separate instances, sampling another 100 examples. Then the newly discovered prototype that is closest to the one first selected is used to perform two more iterations, resulting in a surrogate model trained with 600 samples. The two resulting runs are shown in Fig. 6. Every branch in the phylogenetic tree of designs represents a selected prototype and every layer contains the prototypes found in an iteration. 18.8 prototypes were found on average in each iteration. The surrogate model gives an accurate prediction of the drag force of all classes.

## 5 Conclusion

Quality-diversity algorithms can produce a large array of solutions, possibly impeding an engineer’s capabilities of making a design decision. We introduce computer-aided ideation, using QD in conjunction with a state of the art dimensionality reduction and a standard clustering technique, grouping similar solutions into classes and their representative prototypes. These prototypes can be selected to constrain QD in a next iteration of design space exploration by seeding it with the selected class. A posteriori articulation of preference allows automated design exploration under the design by shopping paradigm. Decisions can be based on an engineer’s personal experience and intuition or other “softer” design criteria that can not be easily formalized. PRODUQD, an example of such a CAI algorithm, allows an engineer to partially unfold a phylogenetic tree of designs by selecting prototypical solutions.

The similarity space can be used continuously as it is decoupled from the feature map. This allows the diversity metric, the feature characterization, to change between iterations. The order in which the feature dimensions are chosen can be customized depending on the design process. For example, the engineer



can start searching the design space in terms of diversity of design and later on switch to functional features. In future work, changes in the feature map and their effects on PRODUQD will be analyzed. Although seeding the map proves to be sufficient to guide QD towards the selected prototype, it is not sufficient to guarantee that QD only produces solutions within its class. Constraints could limit the search operation. Adding the distance to the selected prototype to the acquisition function could bias sampling to take place within the class. Finally, although the median solution might be most similar to all solutions within a class, one indeed might choose the fittest solution of a class as its representative.

CAI externalizes the creative design process, building up a design vocabulary by concisely describing many possible optimal designs with representative prototypes. Engineers can cooperate using this vocabulary to make design decisions, whereby ideation allows them to understand the design space not only in general, but around selected prototypes. CAI, a new engineering design paradigm, automates human-like search whilst putting the human back into the loop.

**Acknowledgments.** This work received funding from the German Federal Ministry of Education and Research (BMBF) under the Forschung an Fachhochschulen mit Unternehmen programme (grant agreement number 03FH012PX5 project “Aeromat”). The authors would like to thank Adam Gaier for their feedback.

## References

1. Balling, R.: Design by shopping: a new paradigm? In: Third World Congress of Structural and Multidisciplinary Optimization, pp. 295–297. ISSMO, New York (1999)
2. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is “Nearest Neighbor” meaningful? In: Beeri, C., Buneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1999). [https://doi.org/10.1007/3-540-49257-7\\_15](https://doi.org/10.1007/3-540-49257-7_15)
3. Bradner, E., Iorio, F., Davis, M.: Parameters tell the design story: ideation and abstraction in design optimization. In: Symposium on Simulation for Architecture & Urban Design, pp. 172–197. SCSi, San Diego (2014)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231. AAAI Press, Portland (1996)
5. Flager, F., Haymaker, J.: A comparison of multidisciplinary design, analysis and optimization processes in the building construction and aerospace industries. In: 24th W78 Conference on Bringing ITC Knowledge to Work, pp. 625–630. Elsevier, Maribor (2007)
6. Gaier, A., Asteroth, A., Mouret, J.: Data-efficient exploration, optimization, and modeling of diverse designs through surrogate-assisted illumination. In: Genetic and Evolutionary Computation Conference, pp. 99–106. ACM, Berlin (2017)
7. Heft, A.I., Indinger, T., Adams, N.A.: Introduction of a new realistic generic car model for aerodynamic investigations. SAE 2012 World Congress, Technical report. SAE, Detroit (2012)
8. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)

9. Hwang, C.L., Masud, A.S.M.: Multiple Objective Decision Making - Methods and Applications: A State-of-the-Art Survey, vol. 164. Springer, New York (1979). <https://doi.org/10.1007/978-3-642-45511-7>
10. Jin, Y.: Surrogate-assisted evolutionary computation: recent advances and future challenges. *Swarm Evol. Comput.* **1**(2), 61–70 (2011)
11. Lehman, J., Stanley, K.O.: Evolving a diversity of virtual creatures through novelty search and local competition. In: Genetic and Evolutionary Computation Conference, pp. 211–218. ACM, Dublin (2011)
12. van der Maaten, L., Hinton, G.: Visualizing data using T-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008)
13. Mouret, J.B., Clune, J.: Illuminating Search Spaces by Mapping Elites. [arXiv:1504.04909](https://arxiv.org/abs/1504.04909) (2015)
14. Niederreiter, H.: Low-discrepancy and low-dispersion sequences. *J. Number Theory* **30**, 51–70 (1988)
15. Pearson, K.: On lines and planes of closest fit to systems of points in space. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **2**, 559–572 (1901)
16. Pugh, J.K., Soros, L.B., Stanley, K.O.: Quality diversity: a new frontier for evolutionary computation. *Front. Robot. AI* **3**, 1–17 (2016)
17. Rasmussen, C.E.: Gaussian Processes for Machine Learning. MIT press, Cambridge (2006)
18. Rosch, E.: Cognitive reference points. *Cognit. Psychol.* **7**(4), 532–547 (1975)
19. Salvador, S., Chan, P.: Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In: 16th IEEE International Conference on Tools with Artificial Intelligence, pp. 576–584. IEEE, Boston (2003)
20. Schölkopf, B., Smola, A., Müller, K.-R.: Kernel principal component analysis. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 583–588. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0020217>
21. Shir, O.M., Bäck, T.: Niching in evolution strategies. In: 7th Annual Conference on Genetic and Evolutionary Computation, pp. 915–916. ACM, Washington (2005)
22. Singh, G., Deb, K.: Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: 8th Annual Conference on Genetic and Evolutionary Computation, pp. 1305–1312. ACM, Seattle (2006)
23. Stump, G.: Design space visualization and its application to a design by shopping paradigm. In: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pp. 795–804. ASME, Chicago (2003)
24. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global framework for nonlinear dimensionality reduction. *Science* **290**(5500), 2319–2323 (2000)
25. Tomašev, N., Radovanović, M.: Clustering evaluation in high-dimensional data. In: Celebi, M.E., Aydin, K. (eds.) Unsupervised Learning Algorithms, pp. 71–107. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-24211-8\\_4](https://doi.org/10.1007/978-3-319-24211-8_4)
26. Wittgenstein, L.: Philosophische Untersuchungen. Basil Blackwell, Oxford (1953)
27. Zwicky, F.: Discovery, Invention Research Through the Morphological Approach. Macmillan, New York (1969)



# Sparse Incomplete LU-Decomposition for Wave Farm Designs Under Realistic Conditions

Dídac Rodríguez Arbonès<sup>1,4</sup>, Nataliia Y. Sergiienko<sup>2</sup>, Boyin Ding<sup>2</sup>,  
Oswin Krause<sup>1</sup>, Christian Igel<sup>1</sup>, and Markus Wagner<sup>3(✉)</sup>

<sup>1</sup> Department of Computer Science,

University of Copenhagen, Copenhagen, Denmark

<sup>2</sup> School of Mechanical Engineering, University of Adelaide, Adelaide, Australia

<sup>3</sup> School of Computer Science, University of Adelaide, Adelaide, Australia

markus.wagner@adelaide.edu.au

<sup>4</sup> NETS A/S, Ballerup, Denmark

**Abstract.** Wave energy is a widely available but still largely unexploited energy source, which has not yet reached full commercial development. A common design for a wave energy converter is called a point absorber (or buoy), which either floats on the surface or just below the surface of the water. Since a single buoy can only capture a limited amount of energy, large-scale wave energy production requires the deployment of buoys in large numbers called arrays. However, the efficiency of these arrays is affected by highly complex constructive and destructive intra-buoy interactions. We tackle the multi-objective variant of the buoy placement problem: we are taking into account the highly complex interactions of the buoys, while optimising critical design aspects: the energy yield, the necessary area, and the cable length needed to connect all buoys – while considering realistic wave conditions for the first time, i.e., a real wave spectrum and waves from multiple directions. To make the problem computationally feasible, we use sparse incomplete *LU* decomposition for solving systems of equations, and caching of integral computations. For the optimisation, we employ modern multi-objective solvers that are customised to the buoy placement problems. We analyse the wave field of final solutions to confirm the quality of the achieved layouts.

**Keywords:** Ocean wave energy · Wave energy converter array  
Simulation speed-up · Multi-objective optimisation

## 1 Introduction

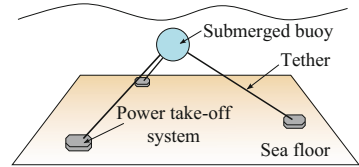
With ever-increasing global energy demand and finite reserves of fossil fuels, renewable forms of energy are becoming increasingly important to consider [16]. Wave energy is a widely available but unexploited source of renewable energy with the potential to make a considerable contribution to future energy production [12]. A multitude of techniques for extracting wave energy are currently

being explored [12,13]. A wave energy converter (WEC) is a device that captures and converts wave energy to electricity. One common WEC design is the point absorber or buoy, which typically floats on the surface or just below the surface of the water, and captures energy from the movement of the waves [12]. In our research, we consider three-tether WECs (Fig. 1) inspired by the next generation of CETO systems developed by the Australian wave energy company called Carnegie Clean Energy. These buoys operate under water surface (fully submerged) and tethered to the seabed in an offshore location.

One of the central goals in designing and operating a wave energy device is to maximise its overall energy absorption. As a result, the optimisation of various aspects of wave energy converters is an important and active area of research. Three key aspects that are often optimised are geometry, control, and positioning of the WECs within the wave energy farm (or array). Geometric optimisation seeks to improve the shape and/or dimensions of a wave energy converter (or some part of it) with the objective of maximising energy capture [17,19]. On the other hand, the optimisation of control is concerned with finding good strategies for actively controlling a WEC [22]. A suitable control strategy is needed for achieving high WEC performance in real seas and oceans, due to the presence of irregular waves [6]. In this article we focus on the third aspect, namely the positioning of multiple wave energy converters while considering constraints, additional objectives, and realistic wave conditions.

To evaluate the performance of our arrays, we use a frequency domain model for arrays of fully submerged three-tether WECs [24]. This model enables us to investigate design parameters, such as number of devices and array layout. In addition to the objective of producing energy, we consider two more objectives: the area needed to place all buoys, and the cable length needed to connect all buoys. This results in an optimisation problem: what are the best trade-offs of the area needed, the buoys' locations, and the cable length needed? To the best of our knowledge, this study is the first to investigate this question to reduce costs and to increase efficiency, *while considering realistic wave conditions* in a multi-objective setting. A first related study is that by Wu et al. [28] where a single objective (power output) was considered and only a single wave frequency and single direction to keep the computational cost at bay. Arbonès et al. [1] investigated multiple objectives by considering parallel architectures and varying numbers of wave frequencies, while again being limited to a single wave direction. Neshat et al. [21] characterised the intra-buoy effects given realistic conditions and exploited this knowledge in custom single-objective hillclimbers.

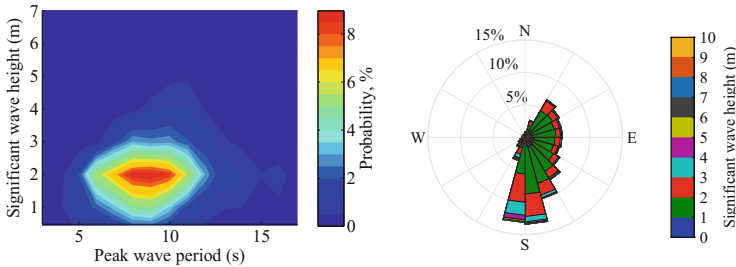
We take this as a starting point for our four contributions here: (i) we use a realistic wave scenario with multiple directions, (ii) we speedup the calculations,



**Fig. 1.** Schematic representation of a three-tether WEC [28].

(iii) we employ a different constraint handling approach to allow the use of other algorithms, and (iv) we provide insights by characterising the wave field.

We proceed as follows. In Sect. 2, we describe the WEC power generation model used in our study and introduce the multi-objective buoy placement problem. We describe the different objectives that are subject to our investigations, and the constraints used and how we implemented them. We note the problem complexity, which is the factor preventing study of large farms. Then, we present in Sect. 3 our methods to reduce running times and the constraint handling used. We describe and present our experiments in Sect. 4, provide a discussion of the results in Sect. 5, and conclude with a summary in Sect. 6.



**Fig. 2.** Australia/New South Wales (NSW) test site near Sydney: wave data statistics (left) and the directional wave rose [2] (right).

## 2 Preliminaries

The total performance of a wave energy farm is not only dependent on the number of WEC units in the array, but also on their mutual arrangement and separating distances. The total capital expenditure per single unit decreases significantly with increase in the farm scale [20]. When operating in a group, WECs interact with each other modifying the incident wave front which can lead to the significant reduction in generated power [3]. Moreover, the interference between converters can be destructive as well as constructive which purely depends on their hydrodynamic parameters and coupling. Thus, the array layout is of great importance for the efficient operation of the whole farm, as well as the wave conditions (dominant wave periods and wave directions).

The WEC chosen for this study is a fully submerged spherical buoy connected to three tethers (taut moored) that are equally distributed around the buoy hull (Fig. 1). Each tether is attached to the individual power generator at the sea floor, which allows to extract energy from surge and heave motions simultaneously [23]. The geometric parameters of the buoys are as follows: they have a 5 m radius, are submerged at 6 m below the water surface, have a weight of 376 tons, and the tether inclination angle from the vertical is  $55^\circ$ . A particular site on the east coast of Australia has been selected as one of the potential locations for the farm installation (see Fig. 2 for sea site statistics).

## 2.1 Objectives

We consider a multi-objective optimisation scenario, using various evolutionary algorithms, where multiple goals are leveraged to obtain a set of solutions.

**Power Output.** The frequency domain model of this kind of WEC arrays has been derived by Sergiienko et al. [24], and used by in related work [1, 28]. In the model, the hydrodynamic interaction of submerged spheres is taken from [27] and the machinery force of each power take-off unit is modelled as a linear spring-damper system. The output from the model is a power absorbed by the array of WECs  $P(\mathbf{x}, \mathbf{y}, \omega, \beta)$  that is a function of their spatial position (coordinates)  $(\mathbf{x}, \mathbf{y})$ , wave frequency  $\omega$ , and wave angle  $\beta$ . As a result, the optimisation problem that corresponds to the power production of the array is expressed:

$$\max_{(\mathbf{x}, \mathbf{y})} \int_{\beta} f_{\beta} \cdot \left( \int_{\omega} f_{\omega} \cdot P(\mathbf{x}, \mathbf{y}, \omega, \beta) d\omega \right) d\beta, \tag{1}$$

There is no closed form solution for this equation. The result is computed by a discrete set of wave frequencies and angles sampled from the distribution.

**Additional Objectives.** As the second objective after the wave farm’s power output, we use the Euclidean minimum spanning tree (MST) to calculate the minimum length of cable or pipe required to connect all buoys.

Thirdly, the cost of the convex hull is defined as the area contained by the set of buoys that form the convex hull. This corresponds to the minimum land area that is required for a wave farm layout. While we omit it here, a safety distance at the perimeter of the wave farm should be included for production purposes.

**Constraints.** The problem uses two types of constraints. Box constraints restrict the available sea surface, and prevent the use of unrealistic amounts of space. The second constraint ensures that no two buoys are placed closer than 50m. This prevents damage and allows for installation and maintenance ships (such as the ATLANTIC HAWK VESSEL) to navigate between the buoys safely.

## 2.2 Problem Complexity

The main computational burden is coming from the evaluation of the power output, which involves (i) the approximation of singular numerical integrals involved in the hydrodynamic model [27], and (ii) solution of the linear system of  $3 \times N$  motion equations of the form  $Ax = b$ , where  $N$  corresponds to the number of buoys in the array. As a result, the complexity of a function evaluation depends on a number of factors, including, but not limited to, the number of buoys, wave directions and number of frequencies considered. To obtain a reliable power prediction, we sample a set of wave frequencies and angles. The accuracy of the result depends on quantity and probability of parameters chosen. Therefore there is an accuracy/time trade-off. The problem quickly becomes untractable for farm sizes of practical interest. In this article, we prioritize reducing the runtime of the power output computation to obtain the largest benefits.

Furthermore, the interbuoy-distance constraint is non-convex, which prevents the use of some algorithms that cannot handle this type of constraints. Relaxation of this constraint is not considered, as it would discard potentially good solutions.

### 3 Computational Speed-Ups and Constraint Handling

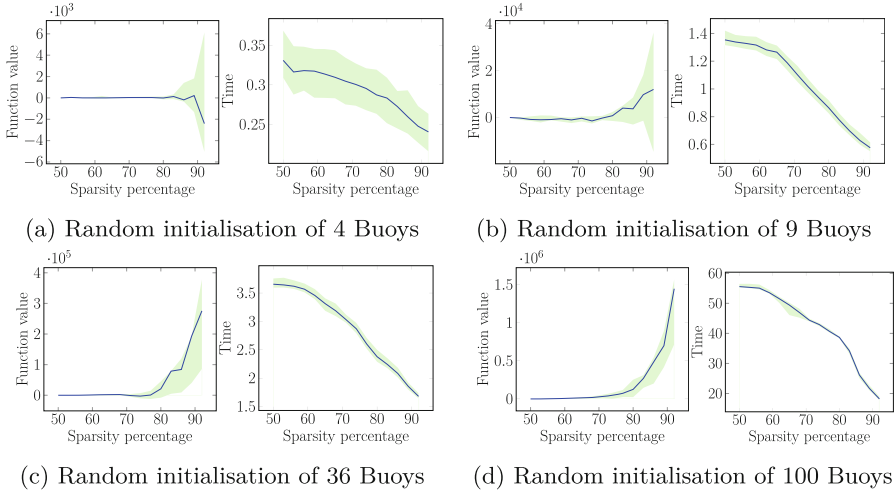
**Numerical Integration.** The integrals in the hydrodynamic model span over an infinite interval and contain a singularity at some point  $K$ . To obtain an approximation, we use an implementation of Cauchy principal value for the interval  $(0, 1.5K)$ , and an algorithm based on a 21-point Gauss-Kronrod rule (provided by the GNU Scientific Library [5]) for the remaining infinite interval.

**Caching.** During evaluation of the power output function, the integral is evaluated several times with different parameters, pertaining to the positioning of the buoys. These integrals appear often with the same parameters, and thus, do not have to be recomputed. We cache the results, which allows for a more efficient use of computational resources and avoids unnecessary calculations.

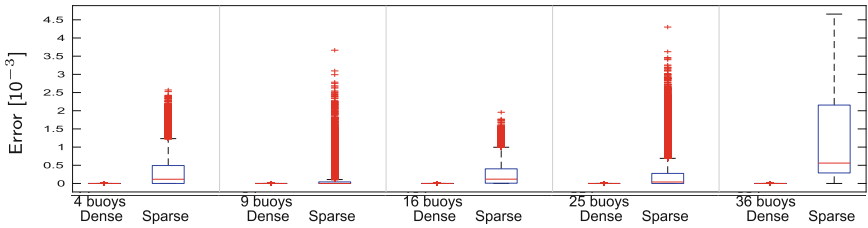
**Linear Algebra.** The linear systems of the form  $Ax = b$  become the bottleneck after the approximation of the integrals. The typical choice for solving this type of system of equation is the  $LU$ -factorization with partial pivoting. However, for our application this approach is too slow as we need to solve several thousand systems of equations throughout the optimisation process. Instead, we make use of the fact that this system has many variables with values very close to zero and thus their contribution to the final solution is negligible. One approach is to compute a sparse incomplete  $LU$ -decomposition as a pre-conditioner for an iterative algorithm. This procedure adds the cost of computing the approximate decomposition in trade for fast solving of the system of equations. This approach works best when the system has to be solved with several right hand sides as in this case, where the cost of computing the  $LU$ -decomposition amortises.

In our case, we can not reuse the  $LU$ -decomposition. Instead we use the fact that for a low percentage of zero-entries the incomplete  $LU$ -decomposition gives a good approximation to the original system. Thus we can approximate the original system by a sparse variation where we discard the smallest percentile of values and solve it approximately using the incomplete  $LU$ -decomposition. This saves time approximately linear in the percentage of discarded values.

We have to evaluate experimentally at which percentage of discarded values we can still obtain a reasonable accuracy. For this, we generate 100 random feasible buoy layouts. While keeping the layouts fixed, we discard values and compare the computed power output to the dense solution. Figure 3 shows the obtained solutions with respect to matrix sparsity, where the power output of each layout has been subtracted for comparison. We can see that run-time decreases linearly with the increasing number of discarded values. The accuracy of the solution remains stable until 75% sparsity, where it starts to degrade. The accuracy loss of the 70% sparse solution with respect to the dense implementation is shown in Fig. 4. To obtain the error of the linear system  $Ax = b$ , we use the formula  $\|As - b\|/\|b\|$ , where  $s$  is the solution obtained.



**Fig. 3.** Relative power output (left) and time per iteration (right), against sparsity percentage; medians of 100 runs (blue), 5%/95% percentiles (green). (Color figure online)



**Fig. 4.** Relative residual error of 100 different random feasible layouts using dense and sparse solver. For the sparse, 70% of the smallest values were discarded.

**Constraint Handling.** The box constraint to allow buoy placements only in the designated area is enforced by a sinusoidal function of the form [7]:  $x = a + (b - a) * (1 + \cos(\pi * x / (b - a) - \pi)) / 2$ . The range of this function is  $(a, b)$ , and provides a smooth transition near the boundaries which is beneficial for the algorithms. By setting  $a, b \in \mathbb{R}$  to the box limits, we guarantee that any solution obtained will lay within the feasible range.

We implemented the inter-buoy constraint with a penalty function proportional to the square of the violation distance. The function takes the set of all buoys  $(\mathbf{b}_1 \dots \mathbf{b}_n)$ , and a minimum distance parameter  $M$ :  $v(\mathbf{b}_1 \dots \mathbf{b}_n) = \sum_{i=1}^n \sum_{j \neq i}^n \max(M^2 - \|\mathbf{b}_i - \mathbf{b}_j\|^2, 0)$ . The objectives  $\mathbf{F}$  of a given layout are then scaled according to a penalty regularisation parameter  $K$ :  $\mathbf{F}' = \mathbf{F} (1 + K v)$ .

Other constraint handling approaches, e.g. as they are used for handling geo-constraints, could have been considered [14, 15], however, this is beyond the scope of this present paper.



## 4 Experimental Study

**Experimental Setup.** To obtain a realistic output estimate and to generate solutions robust to the changing nature of the sea we choose to use 25 linearly-spaced frequencies and 7 wave directions sampled from Fig. 2. Note that a direction of  $0^\circ$  indicates waves coming from the south.

We run experiments for farms of 4, 9, 16, 25 and 36 buoys. We set the boundaries of the farm depending on the amount of buoys to be placed, using  $20.000\text{ m}^2$  per buoy. This results in squares of sides 283 m, 424 m, 566 m, 707 m, and 849 m. We limit most of our report here to 4, 9, and 36 buoys.

We use UNBOUNDED-POPULATION-MO-CMA-ES (UP-MO-CMA-ES) [11], STEADY-STATE-MO-CMA-ES (SS-MO-CMA-ES) [9], SMS-EMOA [4]. Furthermore, for comparison purposes, we use the variant of SMS-EMOA with custom operators presented in [1] (SMS-EMOA\*). These operators are specific to our kind of placement problem and have been used in wind farm turbine placement as well as WEC placement optimisation [1, 25, 26]. In particular, MOVEMENTMUTATION moves single WECs along corridors for local search purposes, and BLOCKSWAPCROSSOVER recombines sub-layouts from complete layouts in order to potentially recombine good sub-layouts into higher-performing ones. We run each combination of algorithm and amount of buoys 100 times.

We initialise with a population size of  $\mu = 50$ , and run the experiments for 8000 iterations (for 25 and 36 buoys the budget is 10000). For SS-MO-CMA-ES and UP-MO-CMA-ES we set  $\sigma = 50$ . We initialise the algorithms with  $\mu = 50$  grids of different sizes, i.e., from the smallest grid (inter-buoy distance 50 m) to the largest grid where the outermost buoys are at the boundary.

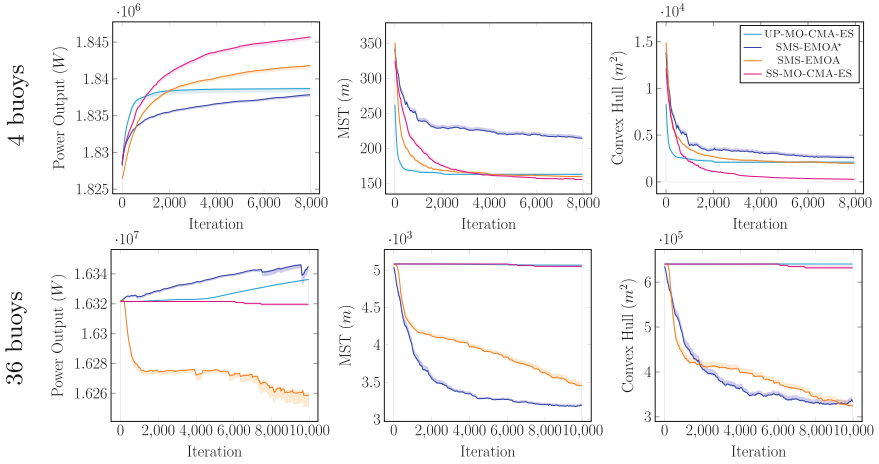
We use  $K = 100$  in the regularisation of infeasible layouts, as we found this to be a good trade-off between preventing the algorithms from using infeasible solutions, and allowing exploration of regions close to the boundaries.

We focus on the power output because it is the objective of highest practical importance. The convex hull and minimum spanning tree attempt to decrease the cost and resource utilization of the final solution, while the power output is the target driving the funding and development of the farm infrastructure.

**Experimental Results.** We present the results of our experiments for the different multi-objective algorithms used. Our inter-buoy penalty does not guarantee that infeasible solutions will not be produced, therefore we ignore them here.

As the power objective is most important, we first present the evolution of the points with the highest power output. For all farm sizes considered, we show the means over the points with highest power output of all fronts and their 75% confidence intervals for each iteration. Additionally, Fig. 5 shows the values of minimum spanning tree (MST) and convex hull (CH) of those points.

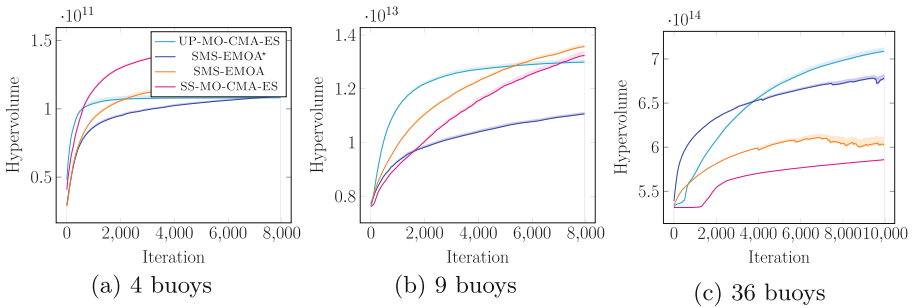
To compare the performance of the multi-objective algorithms we use the so-called hypervolume, which is the volume of the space dominated by the found solutions and a chosen reference point as in [4]. We show the evolution of the volume over the course of optimisation in Fig. 6 for all algorithms.



**Fig. 5.** Evolution of the three objectives for all algorithms. Shown are the means of 100 runs with 75% confidence intervals.

**Table 1.** Objectives attained by initial and optimised individuals.

Buoys	Highest power initial solution			Highest overall power solution		
	Power (MW)	MST (m)	CH (m <sup>2</sup> )	Power (MW)	MST (m)	CH (m <sup>2</sup> )
4	1.8258	396	17635	1.8497	152.29	10.8
9	4.1042	1008	63635	4.1590	493	10465
16	7.2873	1734	124906	7.3254	1263	98797
25	11.3506	2520	183542	11.4145	1823	156958
36	16.3215	5082	640442	16.3757	3080	323946



**Fig. 6.** Hypervolumes: means of 100 runs with 75% confidence intervals. The reference point is based on the worst values obtained for each objective.

In Fig. 7, we show the set of non-dominated feasible solutions found by any algorithm after the last iteration. The objective value achieved by the layouts with highest power outputs are given in Table 1. As we can see, the power output

of the best solutions always increased slightly over the initial best layouts, while the MST length and the area needed both decreased significantly. This means that the newly found layouts not only produce more energy, but also require shorter pipes and a smaller area.

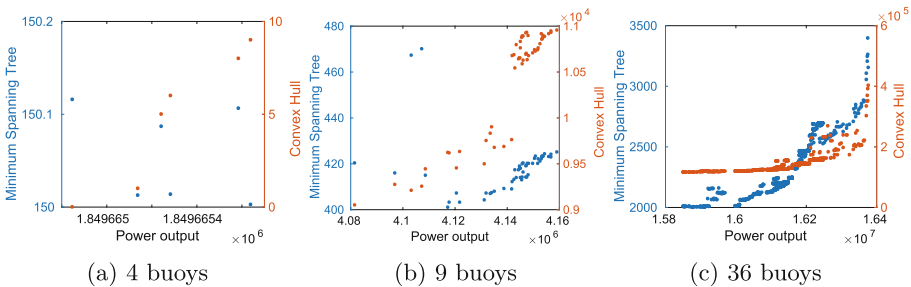
## 5 Discussion

**Optimisation Interpretation.** The modified SMS-EMOA worked better for the best individuals except in 4 dimensions. In terms of hypervolume, the UP-MO-CMA consistently outperformed the other variants for larger layouts. We obtained a roughly 1% improvement on average over the best initial grid.

The SS-MO-CMA-ES consistently performs well on the 4-buoy layout, however it becomes worse on the larger layouts and fails for layouts with more than 9 buoys. The UP-MO-CMA-ES performs better in comparison. We argue that the reason for this is the complex function landscape with constraints in conjunction with the different measures of progress. The UP-MO-CMA-ES only requires a point to be non-dominated to make progress. Thus it has more chances to adapt to the function landscape. The SS-MO-CMA-ES in comparison must create points which are non-dominated but also an improvement in covered volume. Thus the SS-MO-CMA-ES will quickly adapt to evaluate solutions close to existing solutions and thus might easily get stuck in local optima.

The SMS-EMOA has good performance when used in farm sizes of 4 buoys, but lags behind for larger farms. In contrast, SMS-EMOA\* consistently outperforms all other algorithms and produces the best solutions. This shows that the operators developed for wind turbine placement generalise to the similar task of WEC positioning. However, in terms of hypervolume covered, it lags behind the UP-MO-CMA-ES.

One might wonder whether our best performing layouts (in terms of power output) are optimal. While we have no means of proving optimality, we do know that the UP-MO-CMA-ES used in the experiment uses 20% of the given budget on the corner points. This means it spends a considerable amount of effort on

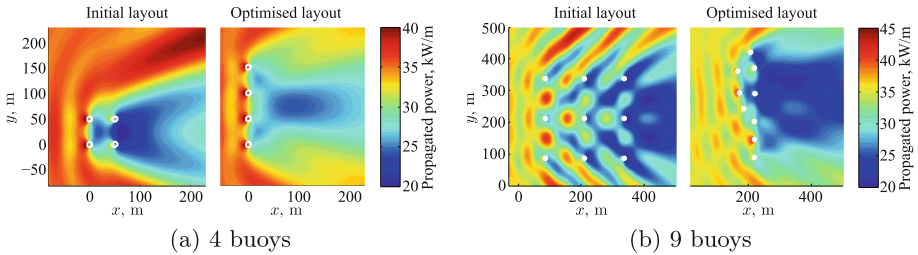


**Fig. 7.** Aggregated fronts of all algorithms' non-dominated solutions. The three dimensional objective space is plotted twice into the two-dimensional space.

exploring extreme trade-offs, among which are the layouts with highest power output. Therefore, the results of UP-MO-CMA-ES given here provide a good intuition of how UP-MO-CMA-ES's single-objective cousin CMA-ES [8] would perform, albeit with a smaller budget.

**Hydrodynamic Interpretation.** In order to analyse the optimisation results, it is necessary to understand how a particular array layout modifies the wave field and how much power propagates downstream as waves travel through the farm. Firstly, we explore the behaviour of the wave farm for the dominant wave period of 9 s ( $\omega = 0.7 \text{ rad/s}$ ) and the wave angle of  $0^\circ$ . For the following interpretation we use WAMIT, which is a state-of-the-art tool used by the industry and research community for analysing wave interactions.

When a wave hits the buoy, a part of the wave front passes through the object creating a wake field behind, a part of the wave is diffracted back and the rest is absorbed by the converter. Other wave types are the radiated waves that spread uniformly in all directions from the oscillating structure (wave source). Depending on the phase information, these three types of waves can be superimposed on each other creating a more energetic wave field, or in other case they can eliminate each other leading to the smaller or zero wave amplitude. Thus, for the wave farm design it is important to place buoys in such locations when waves create a *constructive interaction* resulting in more wave power.



**Fig. 8.** The wave field around the 4 and 9-unit arrays of WECs with the initial (left) and optimised (right) layouts. White circles show the location of submerged spherical buoys. The wave propagates from left.

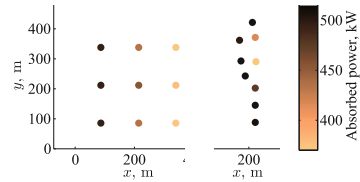
In Fig. 8 (left), we show the wave energy transport per unit frontage of the incident and radiated wave for the 4-unit array. It can be seen that the initial square layout has two converters located in a wake of the first row which decreases their power output. The incident wave energy transport for this wave period is around  $35 \text{ kW/m}$ , while only  $25 \text{ kW/m}$  are propagated to the back row. As has been stated in [3], the park effect in the wave farm is the most significant for the front buoys as they benefit from radiated waves of a row behind. Interestingly, WECs in the optimised layout are lined up perpendicular to the wave front. An inter-buoy distance is about 51 m which is equal to  $0.43\lambda$ , if we consider only one dominant frequency of the spectrum (here  $\lambda$  is a wavelength). Comparing this result with existing literature, this particular scenario buoys should

be separated by  $0.85\lambda = 100\text{ m}$  [10,18] in order to achieve the maximum constructive interaction in the array leading to a quality factor of 1.5. However, the other optimisation objectives came into place limiting the inter-buoy distance.

Similar behaviour of the optimisation algorithm is observed for the case of 9 buoys (see Fig. 8, right) resulting in the decreased number of rows as compared to the initial layout. From the hydrodynamic point of view, it would be even better to have only one row perpendicular to the wave front. However, single-line initialisation is not robust when a spectrum of wave directions is considered, and they would also require larger-than-allowed maximal dimensions.

With increasing number of units in the array, a more complex interaction between buoys takes place leading to the non-trivial optimisation results. In comparison to the 4-buoys array, more interesting effects can be observed looking at the wave field created by the 9-buoy array with the initial layout (see Fig. 8 left). It becomes obvious that initially all converters have been placed to the areas, where radiated waves from adjacent buoys create disadvantageous conditions for power generation. In contrast, the coordinates of all converters in the optimised layout (see Fig. 8 right) coincide with locations where more energy can be captured (similar to the local maxima on the surface plot), especially it is observed for the buoys placed in front.

Going deeper in the analysis, power outputs from all WECs within the 9-unit array are shown in Fig. 9 for the initial and optimised layouts. As expected, for arrays with a regular grid (initial case), the amount of generated power from each row is reduced by about 10% as compared to the row ahead. In the final layout almost all WECs have power output higher than 450 kW, which proves the effectiveness of the optimisation algorithms.



**Fig. 9.** Levels of absorbed power by the 9-unit arrays for the initial (left) and optimised (right) layouts. WECs sizes are not to scale.

## 6 Conclusions

Wave energy is widely available around the globe, however, it is a largely unexploited source of renewable energy. Over the last years, the interest in it has increased tremendously, with dozens of wave energy projects being at various stages of development right now. In our studies we focused on point absorbers (also known as buoys). As the energy capture of a single buoy is limited, the deployment of large numbers of them is necessary to satisfy energy demands. In such scenarios, it is important to consider realistic intra-buoy interactions in order to optimise the operations of a wave energy farm.

In this article, we investigated the placement optimisation with respect to three competing objectives. To speed up the simulations of the intra-buoy interactions, we considered the use of sparse incomplete decompositions to solve linear systems. We tested different evolutionary optimisation algorithms, including custom variation operators developed for wind turbine placement. All simulations

were done assuming realistic scenarios with waves coming from various directions with different probabilities and different wave spectra.

The volume covered by the solutions of the different algorithms showcases the complexity of the wave energy model for larger farm sizes. The highest power obtained from the experiments achieved a 1% increase in power on average over the best grid-based initial layout. In addition, the optimised layouts require significantly shorter cables (or pipes) for the interconnection, and a significantly smaller area for the installation.

In summary, our results show that the fast and effective multi-objective placement optimisation of wave energy farms under realistic conditions is possible and yields significant benefit. Furthermore, our results are consistent with previous results obtaining optimal separation between buoys.

## References

1. Arbonès, D.R., Ding, B., Sergiienko, N.Y., Wagner, M.: Fast and effective multi-objective optimisation of submerged wave energy converters. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 675–685. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_63](https://doi.org/10.1007/978-3-319-45823-6_63)
2. Australian Wave Energy Atlas (2016). <http://awavea.csiro.au/>. Accessed 07 June 2016
3. Babarit, A.: On the park effect in arrays of oscillating wave energy converters. *Renew. Energy* **58**, 68–78 (2013)
4. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
5. GNU Scientific Library. Version 1.16 (2013). <http://www.gnu.org/software/gsl/>. Accessed 2 Apr 2017
6. Hals, J., Falnes, J., Moan, T.: A comparison of selected strategies for adaptive control of wave energy converters. *J. Offshore Mech. Arctic Eng.* **133**(3), 031101 (2011)
7. Hansen, N.: CMA-ES Source Code: Practical Hints (2014). [https://www.lri.fr/~hansen/cmaes\\_inmatlab.html](https://www.lri.fr/~hansen/cmaes_inmatlab.html). Accessed 2 Apr 2017
8. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: *IEEE Congress on Evolutionary Computation*, pp. 312–317 (1996)
9. Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. *Evol. Comput.* **15**(1), 1–28 (2007)
10. Justino, P., Clément, A.: Hydrodynamic performance for small arrays of submerged spheres. In: *5th European Wave Energy Conference* (2003)
11. Krause, O., Glasmachers, T., Hansen, N., Igel, C.: Unbounded population MO-CMA-ES for the Bi-objective BBOB test suite. In: *Genetic and Evolutionary Computation Conference*, pp. 1177–1184. ACM (2016)
12. Lagoun, M., Benalia, A., Benbouzid, M.: Ocean wave converters: state of the art and current status. In: *IEEE International Energy Conference*, pp. 636–641 (2010)
13. López, I., Andreu, J., Ceballos, S., de Alegría, I.M., Kortabarria, I.: Review of wave energy technologies and the necessary power-equipment. *Renew. Sustain. Energy Rev.* **27**, 413–434 (2013)

14. Lückehe, D., Wagner, M., Kramer, O.: On evolutionary approaches to wind turbine placement with geo-constraints. In: Genetic and Evolutionary Computation Conference, pp. 1223–1230. ACM (2015)
15. Lückehe, D., Wagner, M., Kramer, O.: Constrained evolutionary wind turbine placement with penalty functions. In: IEEE Congress on Evolutionary Computation (CEC), pp. 4903–4910 (2016)
16. Lynn, P.A.: Electricity from Wave and Tide: An Introduction to Marine Energy. Wiley, Hoboken (2013)
17. McCabe, A., Aggidis, G., Widden, M.: Optimizing the shape of a surge-and-pitch wave energy collector using a genetic algorithm. *Renew. Energy* **35**(12), 2767–2775 (2010)
18. McIver, P.: Arrays of wave-energy devices. In: 5th International Workshop on Water Waves and Floating Bodies, Oxford, UK (1995)
19. Mohamed, M., Janiga, G., Pap, E., Thévenin, D.: Multi-objective optimization of the airfoil shape of Wells turbine used for wave energy conversion. *Energy* **36**(1), 438–446 (2011)
20. Neary, V.S., et al.: Methodology for design and economic analysis of marine energy conversion (MEC) technologies. Technical report, Sandia National Laboratories (2014)
21. Neshat, M., Alexander, B., Wagner, M., Xia, Y.: A detailed comparison of meta-heuristic methods for optimising wave energy converter placements. In: Genetic and Evolutionary Computation. ACM (2018, accepted)
22. Nunes, G., Valério, D., Beirao, P., Da Costa, J.S.: Modelling and control of a wave energy converter. *Renew. Energy* **36**(7), 1913–1921 (2011)
23. Scruggs, J.T., Lattanzio, S.M., Taflanidis, A.A., Cassidy, I.L.: Optimal causal control of a wave energy converter in a random sea. *Appl. Ocean Res.* **42**(2013), 1–15 (2013)
24. Sergiienko, N.Y., Cazzolato, B.S., Ding, B., Arjomandi, M.: Frequency domain model of the three-tether WECs array (2016). <http://tiny.cc/ThreeTether>. Code: <http://tiny.cc/OptEn>. Accessed 1 Mar 2018
25. 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, pp. 1381–1388. ACM (2013)
26. Wagner, M., Day, J., Neumann, F.: A fast and effective local search algorithm for optimizing the placement of wind turbines. *Renew. Energy* **51**, 64–70 (2013)
27. Wu, G.X.: Radiation and diffraction by a submerged sphere advancing in water waves of finite depth. *Math. Phys. Sci.* **448**(1932), 29–54 (1995)
28. Wu, J., et al.: Fast and effective optimisation of arrays of submerged wave energy converters. In: GECCO, pp. 1045–1052. ACM (2016)



# Understanding Climate-Vegetation Interactions in Global Rainforests Through a GP-Tree Analysis

Anuradha Kodali<sup>1</sup>, Marcin Szubert<sup>2</sup>, Kamalika Das<sup>1</sup>(✉), Sangram Ganguly<sup>3</sup>,  
and Joshua Bongard<sup>2</sup>

<sup>1</sup> USRA, NASA Ames Research Center, Moffett Field, CA, USA  
anu.uconn@gmail.com, kamalika.das@nasa.gov

<sup>2</sup> University of Vermont, Burlington, VT, USA  
{marcin.szubert, j bongard}@uvm.edu

<sup>3</sup> BAERI Inc., NASA Ames Research Center, Moffett Field, CA, USA  
sangram.ganguly@nasa.gov

**Abstract.** The tropical rainforests are the largest reserves of terrestrial carbon and therefore, the future of these rainforests is a question that is of immense importance in the geoscience research community. With the recent severe Amazonian droughts in 2005 and 2010 and on-going drought in the Congo region for more than two decades, there is growing concern that these forests could succumb to precipitation reduction, causing extensive carbon release and feedback to the carbon cycle. However, there is no single ecosystem model that quantifies the relationship between vegetation health in these rainforests and climatic factors. Small scale studies have used statistical correlation measure and simple linear regression to model climate-vegetation interactions, but suffer from the lack of comprehensive data representation as well as simplistic assumptions about dependency of the target on the covariates. In this paper we use genetic programming (GP) based symbolic regression for discovering equations that govern the vegetation climate dynamics in the rainforests. Expecting micro-regions within the rainforests to have unique characteristics compared to the overall general characteristics, we use a modified regression-tree based hierarchical partitioning of the space to build individual models for each partition. The discovery of these equations reveal very interesting characteristics about the Amazon and the Congo rainforests. Our method GP-tree shows that the rainforests exhibit tremendous resiliency in the face of extreme climatic events by adapting to changing conditions.

**Keywords:** Hierarchical modeling · Symbolic regression  
Genetic programming · Earth science · Nonlinear models

---

A. Kodali—Currently at AllState Innovations.

M. Szubert—Currently at Google Inc., Zürich.

This is a U.S. government work and its text is not subject to copyright protection in the United States; however, its text may be subject to foreign copyright protection 2018

A. Auger et al. (Eds.): PPSN 2018, LNCS 11101, pp. 525–536, 2018.

[https://doi.org/10.1007/978-3-319-99253-2\\_42](https://doi.org/10.1007/978-3-319-99253-2_42)



## 1 Introduction

Physics based modeling and perturbation theory has long been used to study the eco-climatic interactions by scientists in order to explain observed phenomena. However, these models, derived under various assumptions of equilibrium, are often only suitable for ideal conditions, and fail to explain the complex dynamics of ecosystem responses to varying environmental factors, especially in the context of a progressively warming global climate. Given the vast amounts of data being collected by different ground-based and remote sensing instruments over long periods of time, the Earth Science research community is extremely data rich. As a result, there has been a slow and steady shift towards the use of machine learning for answering many of their science questions. Ensemble approaches for climate modeling, uncertainty analysis for model evaluation, network based analysis for discovery of new climate phenomena are examples [1]. However, most of the analysis approaches used for climate-vegetation dynamics have been restricted to simple statistical correlation analysis or linear regression [17], thereby limiting discoveries to only linear dependencies. In this work, we formulate the problem of understanding vegetation-climate relationship in rainforests as a regression problem where different climate variables and other influencing factors form the set of independent regressors, and data representing vegetation in the rainforests is the target. In the hope of understanding how climate affects vegetation, we discover regression equations that best fit the observed data. We alleviate the limitation of linear models through the use of a genetic programming based symbolic regression [5] which is a data driven white-box model that allows us to learn both the structure and weights of the regression equation, thereby revealing previously unknown nonlinear interactions in the data. We combine symbolic regression with hierarchical modeling using regression trees in order to partition the large space of spatio-temporal interactions for discovering micro regions within the vast rainforest expanses.

The tropical rainforests are the largest reserves of terrestrial carbon sink, predominantly due to the presence of homogeneous, dense, moist forests over extensive regions. The Amazon forests, for example, are a critical component of the global carbon cycle, storing about 100 billion tons of carbon in woody biomass [7], and accounting for about 15% of global net primary production (NPP) and 66% of its inter-annual variability [19]. Together with the Congo basin in Africa and the Indo-Malay rainforests in Southeast Asia, tropical forests store 40–50% of carbon in terrestrial vegetation and annually process approximately six times as much carbon via photosynthesis and respiration as humans emit from fossil fuel use [6]. With the recent severe Amazonian droughts in 2005 and 2010 [13, 17] and on-going multi-decadal drought in the Congo region [20], there is growing concern that these forests could succumb to precipitation reduction, causing extensive carbon release and feedback to the carbon cycle [3]. Interestingly, the two largest rainforests display different characteristic drought patterns with Amazonia encountering episodic and abrupt droughts during the dry season (July–September) and Congo experiencing a gradual and persistent

water shortage. Individual studies of these forests or small areas within them fail to identify any unifying theory that holds for these global rainforests.

In this work we learn from the various observations pertaining to these rainforests in the context of a single modeling framework. We develop a regression tree approach called GP-tree where the models at each node of the tree are built using symbolic regression [5]. This framework discovers dynamics that are local to different partitions within the forests and can be used to explain why certain areas of the rainforests have responded very differently to the extreme climate events of the recent times. The discoveries have been validated by domain scientists conversant with the rainforest ecosystem modeling problem. Precipitation and temperature are the two most relevant climatic factors affecting the rainforests. Other relevant physiological factors that have been included based on domain science expertise are elevation and slope which directly affect how rainfall (or lack thereof) can influence vegetation. Given that forest greenness is an established indicator of tree health, we use satellite-based vegetation greenness observations as our target for this ecosystem model. The goal of the GP tree method is to learn the dependency of greenness on the climatic and physiological factors from historical data spanning multiple years of observations. An additional goal is to identify boundaries in this spatial data set where the equations of dependency change.

## 2 Related Work

Standard methods in ecosystem modeling use pairwise correlation analysis of vegetation with each climate variable [16]. Trend analysis on standard anomalies of different time series is commonly used for understanding long term dependencies. Nemani et al. [10] use trend analysis for understanding limiting environmental factors in different zones of the earth. Ordinary least squares regression has been used to model the relationship between vegetation and multiple climate variables [8]. Geographic Weighted Regression (GWR) has also been traditionally used to allow for local spatial correlations while explaining climate-vegetation interactions [18]. However, GWR suffers from serious scaling issues. Cubist [11] is another popular analysis tool that automatically partitions the data into geographic regions while learning linear models in each partition. However, none of the methods allow discovery of nonlinear relationships, which severely restricts the discovery process. Nature inspired learning techniques such as deep learning, although very powerful in extracting nonlinear relationships, are not particularly useful in this context due to their blackbox nature.

## 3 Modeling Framework

Genetic programming based symbolic regression (SR) [5] allows for discovery of nonlinear dependencies in the data by allowing to learn the equation structure along with the regression coefficients. Occasionally when the data is diverse, a single nonlinear model does not suffice. Hierarchical partitioning techniques such

as classification and regression trees (CART) [2] and model trees [11] help in the identification of low variance regions in the data for building individual models. In this paper we describe GP-tree that combines these two powerful algorithms in order to build nonlinear regression models at each partition.

### 3.1 Symbolic Regression

Symbolic regression's (SR's) main defining features are that it is data driven, white box, and nonlinear. Given training and validation data, SR distills equations of arbitrary form and complexity to explain the data. An example equation explaining vegetation-climate interactions for a specific spatio-temporal extent may look like

$$Y = -0.01 \log(e^{X_8} (0.03 e^{4X_6 + X_8 + 2X_9} ((X_5 + X_6)^2 - X_2 - X_3)^2 + 0.2 e^{X_{10}}))$$

where  $X_i, \forall i$  and  $Y$  represent the independent climate variables and greenness respectively. Symbolic regression is instantiated using population-based stochastic optimization method, genetic programming (GP), whose underlying search algorithm is biologically-inspired and consists of 3 major operations, namely, mutation, crossover, and selection [5]. Using these operations, the algorithm iteratively searches the space of possible models by probabilistically recombining previous expressions, modifying their components and adding new random terms to the randomly initialized model population. In each iteration the candidate solutions are evaluated and less accurate and less parsimonious models are replaced by randomly-modified copies of more accurate and more parsimonious models. A squared error measure is used to judge the goodness of fit of the various candidate solutions. The set of solutions form a Pareto front where error on the validation set and model complexity are two competing parameters.

### 3.2 Regression Trees

Decision tree is a machine learning technique for recursively partitioning a space of explanatory (independent) variables in order to better describe a discrete target variable. When the target variables are continuous instead of discrete, regression trees are used. In a regression tree each intermediate node splits the data using a greedy search algorithm that minimizes variance at that node and the leaf nodes contain constant values. A special kind of regression tree called model tree contain leaf nodes which have linear models that can predict the value of a previously unknown example. Regression trees are used in place of a global simple linear regression model where the data has many features that interact in complicated nonlinear ways, and the assumption of linearity falls apart on the entire data set, but might hold true in small subsets. There are different variants of the regression tree algorithms. The original model tree approach proposed by Quinlan [11] relies on building a regression tree with the objective of reducing the standard deviation of the target variable at each split whereas CART [2] chooses to minimize the mean squared error (MSE) of the predicted target value

at each node using decision thresholds. The goodness of fit is determined using the squared error on a validation set and overfitting is handled through tree pruning and cross validation.

### 3.3 GP-Tree

Our approach, GP-tree consists of two steps: induction of a model tree to partition the data into subsets and then learning of governing equations for each partition using symbolic regression. The overall approach for the GP-tree framework is described in Algorithm 1. The details of the framework are described next.

---

#### Algorithm 1. Hierarchical regression: GP-tree

---

**Input:**  $\mathbf{X} \in \mathbb{R}^{n \times D}$ ,  $\mathbf{y} \in \mathbb{R}^n$ , *max\_depth*, *gp\_params*  
**Output:** Tree:  $\mathbf{T}$ , Models:  $M_i$ ,  $i \in k$  (no. of partitions)  
**Step 1:** Build tree: Partition data into  $k$  groups  
 $\mathbf{T} = \text{PolynomialRegressionTree}(\mathbf{X}, \mathbf{y}, \text{max\_depth})$   
 $[\mathbf{X}_1, \dots, \mathbf{X}_k] = \text{Partitiondata}(\mathbf{X}, \mathbf{T})$   
**Step 2:** Train GP models  
**for** each data partition  $(\mathbf{X}_i, \mathbf{y}_i)$  ( $i \in k$ ) **do**  
 $M_i = \text{learnGP}(\mathbf{X}_i, \mathbf{y}_i, \text{gp\_params})$   
**end for**

---

Our tree induction differs from the model tree approach in that, instead of the target variance, we consider the MSE approach of CART. Since we are interested in nonlinear models, we compute the MSE for each split using a second order polynomial regression. We hypothesize that the standard deviation of the target variable may not be enough to find homogeneous partitions with respect to the models. In each recursive call of the algorithm (see Algorithm 2), we attempt to find the best binary splitting criterion that divides the dataset  $\mathbf{X}$  into two subsets that can be accurately explained by second order polynomial models, which is equivalent of running LASSO on the second order feature combinations of the original data set. To this end, for each feature  $f$  we consider a fixed number (100) of scalar threshold values (evenly distributed in the feature domain). For every such pair (feature, threshold) we evaluate the quality of the resulting split by running polynomial regression on the two data subsets  $S_1 = \{\mathbf{X} | \mathbf{X}_f < t\}$  and  $S_2 = \{\mathbf{X} | \mathbf{X}_f \geq t\}$ . The best pair is the one that minimizes the sum of mean squared errors in these subsets. Finally, we invoke the algorithm recursively for the resulting partitions until we reach the maximum depth of the tree. The output of the algorithm is a regression tree with  $2^{\text{depth}-1}$  internal nodes and  $2^{\text{depth}}$  leaves which correspond to partitions of the original dataset. Various methods are available for determining the choice of depth for the model tree [12]; here we use model complexity at the leaf nodes. Although the model tree described above could be used as a predictive model by itself, we attempt to further improve its prediction performance by replacing the second order polynomial models in the terminal leaves of the tree with symbolic regression based models. For each partitions we perform an independent GP

run (see Algorithm 3) using a variant of the Age-Fitness Pareto Optimization (AFPO, [14]) algorithm – a multi-objective method that relies on the concept of genotypic age of an individual (model), defined as the number of generations its genetic material has been in the population. The age attribute is intended to protect young individuals before being dominated by older already optimized solutions.

---

### Algorithm 2. Polynomial Regression Tree

---

```

1: Input:  $\mathbf{X} \in \mathbb{R}^{n \times D}$ ,  $\mathbf{y} \in \mathbb{R}^n$ , depth
2: Output: Tree:  $\mathbf{T}$ 
3: if depth == 0 then
4:   return TerminalNode(LASSO( $\mathbf{X}$ ,  $\mathbf{y}$ ))
5: else
6:   feature, threshold  $\leftarrow \arg \min_{f,t} (LR_{error}(\mathbf{X}|\mathbf{X}_f < t, \mathbf{y}) + LR_{error}(\mathbf{X}|\mathbf{X}_f \geq t, \mathbf{y}))$ 
7:   leftSubtree  $\leftarrow$  LinearRegressionTree( $\mathbf{X}|\mathbf{X}_f < t, \mathbf{y}, \text{depth} - 1$ )
8:   rightSubtree  $\leftarrow$  LinearRegressionTree( $\mathbf{X}|\mathbf{X}_f \geq t, \mathbf{y}, \text{depth} - 1$ )
9:   return InternalNode(feature, threshold, leftSubtree, rightSubtree)
10: end if

```

---



---

### Algorithm 3. Genetic Programming

---

```

1: Input:  $\mathbf{X} \in \mathbb{R}^{n \times D}$ ,  $\mathbf{y} \in \mathbb{R}^n$ , gp-params
2: Output: GP model:  $\mathbf{M}$ 
3: Initialize population of  $n$  random models
4: for number of generations do
5:   Select random parents
6:   Recombine and mutate parents to produce  $n$  offspring
7:   Add offspring to the population
8:   Calculate (error, age, size, complexity) for each model in the population
9:   while population size >  $n$  do
10:    Select  $k$  random models from the population
11:    Determine local Pareto front among  $k$  selected models
12:    Remove Pareto-dominated models from the population
13:   end while
14: end for

```

---

The algorithm starts with a population of  $n$  randomly initialized individuals each of which has age of one which is then incremented by one every generation. In each generation, the algorithm proceeds by selecting random parents from the population and applying crossover and mutation operators (with certain probability) to produce  $n$  offsprings. The offspring is added to the population extending its size to  $2n$ . Then, Pareto tournament selection is iteratively applied by randomly selecting a subset of individuals and removing the dominated ones until the size of the population is reduced back to  $n$ . To determine which individuals are dominated, the algorithm identifies the Pareto front using four objectives (all minimized): prediction error, age, size and expressional complexity. We measure the size of an individual (candidate solution) as the number of nodes in its tree representation. It should be noted here that the regression equation is derived as a tree structure and this tree is different than the hierarchical model tree that is being constructed for the data. For assessing the model complexity, we estimate the order of nonlinearity of the model [15].

## 4 Data and Computation

MODIS (MODerate-resolution Imaging Spectroradiometer<sup>1</sup>) product MYD13Q1 at 250 m-16day spatio-temporal resolution is used to obtain the Normalized Difference Vegetation Index (NDVI), the most commonly used surrogate for vegetation [9]. Land surface temperature (LST) is similarly derived from MODIS product MYD11A1, but at 1 km-1day spatio-temporal resolution. TRMM (Tropical Rainfall Measuring Mission<sup>2</sup>) observations at 25 km-1month spatio-temporal resolution is used for precipitation measurements. GTOPO30<sup>3</sup> is a global digital elevation model (DEM) at 1 km resolution that is used for obtaining elevation data for the rainforests. Slope is derived from elevation using standard differentials [4]. Since broadleaf evergreens constitute the largest vegetation type found in rainforests, we use a MODIS-derived landcover mask MCD12Q1 to retain only the broadleaf evergreen pixels from the MODIS imagery of the rainforests. All data sets (temporal and spatial resolutions) are selected on the basis of data quality and availability.

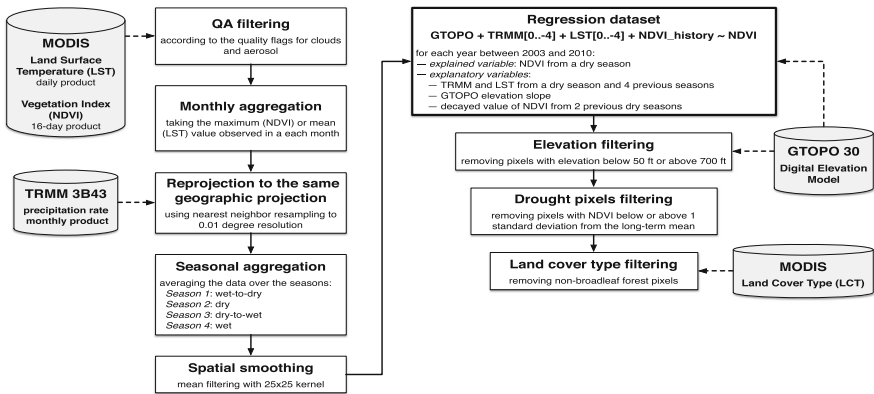


Fig. 1. Data preprocessing pipeline for regression analysis.

For setting up the regression problem, significant amount of preprocessing is needed for collocating and aligning these data products from various sources. Figure 1 shows the end-to-end data preprocessing pipeline. Based on the need of the problem, and the various data sets available, all data sets have been reprojected into the same viewing angle and aligned at 1 km spatial resolution through nearest neighbor interpolation, and averaging based compression. Since seasons largely determine how rainforests respond to environmental influences, we choose a monthly temporal granularity for the study and define the seasons

<sup>1</sup> <https://modis.gsfc.nasa.gov/>.

<sup>2</sup> <https://pmm.nasa.gov/trmm>.

<sup>3</sup> <https://lpdaac.usgs.gov/>.

by aggregating monthly time series for each variable as follows: dry season (D) from July to September, dry-to-wet transition (DW) during October, wet season (W) from November to February, and wet-to-dry transition (WD) from March to June. Noise removal is achieved using QA flags available from the MODIS data products. Spatial smoothing over a square neighborhood surrounding each pixel also helps in noise reduction. Land cover filtering indicates removing non-broadleaf pixels while elevation and wetlands filtering removes highly elevated and flooded areas, respectively. Lastly, drought pixels are anomalies with lower vegetation values over years and are removed from the training data.

**Regression Setup.** Our regression problem is modeling the dry season vegetation as a function of climate and physiological variables in the current (dry) season as well as past seasons going back up to one year. It is set up as follows:  $NDVI_k = f(LST_i, TRMM_i, Elev, Slope)$ , where  $k = current_D$  and  $i \in (D_{current}, D_{last}, WD, W, DW)$  are season indices up to one year back in time. The assumption that vegetation in the current season is only affected by rainfall and precipitation within the last one year is based on Subject Matter Expert (SME) feedback and exploratory analysis with different temporal dependencies. We randomly pick 100K examples (out of 700K) from the years 2003–2006 for training our GP-tree model. Year 2007 containing 160K samples is used for validation. The training years chosen using domain knowledge represent drought years and normal years in precipitation. We set the depth of the polynomial decision tree to 2 based on analysis of MSE and model complexity at each leaf node. A tree of depth 2 produces 4 partitions. Once the partitions are obtained using the polynomial regression tree, we spawn the GP optimization routines on each partition with 5000 generations and population size of 50. We use crossover probability of 0.9 and mutation probability of 0.1. Our list of mathematical operations include addition, subtraction, multiplication, logarithm, exponential, square, and cubic. We initialize 30 different optimizations that generate 30 Pareto fronts of GP models. We pick the best model by comparing a subset of models from each front based on size, model complexity, and mean squared error on validation set.

**Infrastructure.** The data preprocessing pipeline, as well as the modeling and analysis framework have been run on NASA’s Pleiades Supercomputer with the following hardware and software configuration. Each of the worker nodes are based on the Intel Sandy Bridge architecture with dual 8 core 2.6 GHz processors and with 32 GB of memory. All nodes’ operating systems are running SGI ProPack for Linux kernel version 3.0. Pleiades utilizes a PBS scheduler for job submission. The GP-tree algorithm is centralized and uses a master-slave architecture only for parallelizing the splitting decisions for the various feature-threshold choices (see Sect. 3.3). Once the data is partitioned, the symbolic regression equations are computed at each node using massively parallel search based optimization through genetic programming.

## 5 Results Analysis

The GP-tree analysis yields 4 different partitions: two of them are temperature limited and precipitation limited zones while two other partitions have a mix of temperature, precipitation, and elevation affecting vegetation. Figure 2 shows the nonlinear equations for each partition. Partitions are identified using blue (leaf 0), cyan (leaf 1), yellow (leaf 2), and red (leaf 3) colors corresponding to the spatial partitions in Fig. 3.

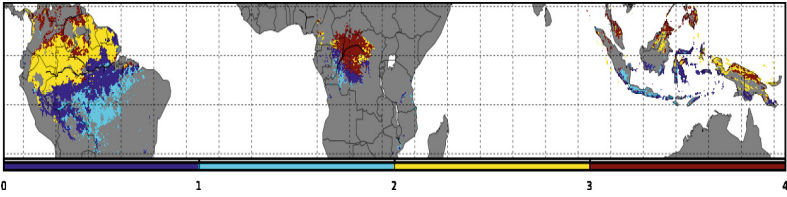
$$\begin{aligned}
 \text{leaf0} &= 0.43 * LST_{DW} * (-0.13 * Elev^c + 0.13 * TRMM_{D_{curr}} * LST_{DW}) + 0.06 * TRMM_{D_{last}}^c - 0.12 * TRMM_{D_{last}} \\
 &\quad * (-LST_{D_{last}} + TRMM_W - (TRMM_{WD} + 0.6 * TRMM_W - 0.26)^2) - 0.09 * LST_{WD}^2 - 0.23 * LST_{WD} \\
 &\quad - 0.23 * LST_W - 0.06 * (Elev + LST_{WD})^2 + 0.16 \quad (1)
 \end{aligned}$$

$$\begin{aligned}
 \text{leaf1} &= -0.2 * LST_{DW} - 0.4 * LST_{D_{last}} - 0.1 * TRMM_{WD}^2 + 0.1 * TRMM_{WD} + 0.2 * TRMM_W * TRMM_{D_{last}}^2 \\
 &\quad * (TRMM_{WD}^2 - TRMM_{WD} - TRMM_W + TRMM_{DW} - LST_{D_{curr}}) + 0.7 * TRMM_W \\
 &\quad - 0.2 * TRMM_{D_{last}}^3 + 0.2 * TRMM_{D_{last}}^2 - 0.3 * LST_{D_{curr}} - 0.3 * LST_{WD} + 0.1 \quad (2)
 \end{aligned}$$

$$\begin{aligned}
 \text{leaf2} &= 0.05 * TRMM_{WD} - 0.1 * LST_{D_{curr}} - 0.05 * LST_W^2 - 0.2 * LST_W + 0.05 * (-Elev + TRMM_{WD}) \\
 &\quad * (-Slope * LST_W + Elev + LST_{WD}) - 0.05 * (-TRMM_{WD} + 0.7)^2 + 0.3 \quad (3)
 \end{aligned}$$

$$\begin{aligned}
 \text{leaf3} &= -0.12 * LST_W * LST_{DW} - 0.12 * LST_{D_{last}} - 0.02 * TRMM_{D_{curr}}^2 + 0.12 * TRMM_{D_{curr}} + 0.12 * TRMM_{WD} \\
 &\quad - 0.12 * LST_{D_{curr}} - 0.12 * LST_{WD} - 0.12 * LST_W * (TRMM_{D_{last}} + 0.12) - 0.12 * LST_W - 0.12 * \log(Elev) \\
 &\quad - TRMM_{D_{curr}} + 1.04) - 0.12 * \log(\log(Elev - 1.09)) \quad (4)
 \end{aligned}$$

**Fig. 2.** Equations at 4 leaf nodes. Colored boxes indicate matching colors in spatial map in Fig. 3 (Color figure online)

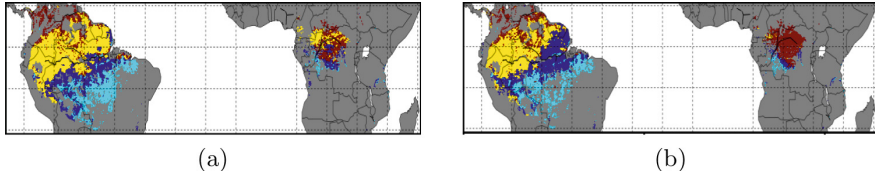


**Fig. 3.** (a) Partitions of the rainforests obtained through GP-tree (Color figure online)

Figure 3 makes it evident that the Amazonian and African rainforests have characteristically different responses to climate, whereas the Indo-Malay rainforests have no defining nature, comprising of an equal mix of the different partitions. The two main partitions encompassing the bulk of the Amazon river basin are yellow described by Eq. 3 and blue described by Eq. 1 in Fig. 2.

The blue region occupying the central Amazon area is heavily dependent on temperature from the month of October ( $LST_{DW}$ ), the positive sign indicating that vegetation in that area prefers colder temperatures during the dry to wet season transition. The presence of the  $TRMM$  terms in Eq. 1 indicates vegetation dependence on seasonal rainfall as well. It shows resilience since a relatively dry wet season (low rainfall during November–February) is compensated by a wetter transition and vice versa. It also shows that vegetation in this





**Fig. 4.** Partitioning (a) 2005 and (b) 2010 pixels of Amazon and Africa using learned GP-tree model (Color figure online)

region does not thrive in excessive rainfall. This can be explained as an effect of the interruption of the adiabatic cooling process that forces temperatures to rise in extreme cloud conditions, thereby effecting vegetation negatively. The yellow partition in the north of the Amazon governed by Eq. 3 requires colder temperatures along with longer rainfall spells overflowing from the wet season to the transition season for increased greening of the trees. The cyan and red partitions representing Eqs. 2 and 4 respectively are spread across the peripheral regions of the Amazon basin. The southern periphery (cyan region) is heavily dominated by wet season rainfall, as seen in Eq. 2. A similar cyan area can also be seen flanking the southern Congo basin Africa. Geographically, both these regions represent a transitional zone in the rainforests, where there is a mix of broadleaf evergreens and savannas (grasslands) that completely depend on rainfall for greening. On the other hand, it is apparent that bulk of the African forests is governed by Eq. 4 described in red in Fig. 3. This is the most complex model including precipitation and temperature covariates from almost all seasons. Lack of copious rainfall in this region for the last two decades has ruined all seasonal patterns for the broadleaf evergreens as they try to sustain themselves through the low to moderate rainfall received during all seasons, while relying on lower temperatures in this region.

These equations enable domain scientists to explain several observations made in the last decade about these rainforests. Given the dependence of any rainforest on appropriate rainfall and temperatures, the permanent state of drought in the African Congos in the last 15 years have led the trees in that region to gradually succumb to the drought indicated by a decreasing NDVI trend [20] over the years. Even slight improvement in rainfall in certain years results in those trees trying to adapt to a different steady state behavior, evident from the appearance of yellow patches in the African red partition in Fig. 4a. The Amazon droughts of 2005 and 2010 also manifest themselves similarly. The trees in the drought-stricken regions of the Amazon, in an attempt to survive under these extreme climatic conditions, adapt to a different steady state behavior (a different equation). As seen in Fig. 4a, a large part of the blue river basin region affected by the 2005 drought turns yellow to account for the sudden water deficiency through increased photosynthetic activity [13]. Similarly, a small part of the yellow region near the mouth of the Amazon river becomes blue after the 2010 drought hits that area, thereby resisting tree dieback due to the unfavorably low rainfall and high temperatures caused by the El Niño phenomenon in

that year. This study shows how the global rainforests, although suffering from frequent droughts and rising temperatures, generally show very strong resilience by adapting to changing conditions.

**Model Performance.** We compare performance of the GP-tree model with 4 different baselines: (i) a single linear model, (ii) a single symbolic regression model, (iii) linear regression tree with linear models at the leaves, and (iv) polynomial regression tree with linear models. We compare mean squared error on a standard validation set (examples for year 2007) for each model. The MSEs are shown in Table 1. The progressive improvement of error as we go from linear to nonlinear model, and from a single global model to multiple models obtained through hierarchical partitioning is evident from the error values. Our method improves the state of the art (first baseline) by almost 43%.

**Table 1.** Table showing mean squared error for GP-tree and the baseline methods for ecosystem modeling

GP-tree	Baseline 1	Baseline 2	Baseline 3	Baseline 4
0.28	0.49	0.31	0.45	0.38

## 6 Conclusion

For ages, scientists have been trying to understand the effect on climate and other environmental variables on vegetation. Given that the rainforests are the largest carbon sinks, it is particularly important to understand how these forests react under changing climatic conditions, and whether their future is at risk. Existing studies using simple correlation analysis or linear regression models built at a global level, have failed to capture the nuanced dependencies of vegetation in micro regions within these rainforests on environmental factors. In this study we use genetic programming based approach symbolic regression for discovering equations that model the vegetation climate dynamics in the rainforests of the world. Expecting micro-regions within the rainforests to have unique characteristics compared to the overall general characteristics, we hierarchically partition the space using a regression tree approach called GP-tree and nonlinear regression models for each partition. Our GP-tree framework discovers that these rainforests exhibit very different characteristics in different regions. We also see that in the face of extreme climate events, the trees adapt to reach a different steady state and therefore, exhibit resiliency.

**Acknowledgments.** This research is supported in part by the NASA Advanced Information Systems Technology (AIST) Program’s grant (NNX15AH48G) and in part by the NASA contract NNA-16BD14C. The authors would also like to thank Dr. Ramakrishna Nemani, a senior Earth Scientist and an expert on this topic, for his insightful comments and perspective on some of the research findings.

## References

1. Banerjee, A., Monteleoni, C.: Climate change: challenges for machine learning. Tutorial at NIPS 2014 (2014)
2. Breiman, L., Friedman, J., Stone, C., Olshen, R.: Classification and Regression Trees. Taylor & Francis, Milton Park (1984)
3. Cox, P.M., Betts, R.A., Jones, C.D., Spall, S.A., Totterdell, I.J.: Acceleration of global warming due to carbon-cycle feedbacks in a coupled climate model. *Nature* **408**(6809), 184–187 (2000)
4. Horn, B.: Hill shading and the reflectance map. *IEEE Proc.* **69**, 14–47 (1981)
5. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT Press, Cambridge (1992)
6. Lewis, S.L., et al.: Increasing carbon storage in intact african tropical forests. *Nature* **457**(7232), 1003–1006 (2009)
7. Malhi, Y., et al.: The regional variation of aboveground live biomass in old-growth amazonian forests. *Glob. Change Biol.* **12**(7), 1107–1138 (2006)
8. Mao, K., et al.: Estimating relationships between NDVI and climate change in Quizhou province, Southwest China. In: 2010 18th International Conference on Geoinformatics, pp. 1–5, June 2010
9. Myneni, R., Hall, F., Sellers, P., Marshak, A.: The interpretation of spectral vegetation indexes. *IEEE Trans. Geosci. Remote Sens.* **33**(2), 481–486 (1995)
10. Nemani, R.R., et al.: Climate-driven increases in global terrestrial net primary production from 1982 to 1999. *Science* **300**(5625), 1560–1563 (2003)
11. Quinlan, J.R.: Learning with continuous classes. In: Proceedings of the Australasian Joint Conference on Artificial Intelligence, pp. 343–348. World Scientific, Singapore (1992)
12. Rokach, L., Maimon, O.: Top-down induction of decision trees classifiers - a survey. *IEEE Trans. Syst. Man. Cybern. Part C (Appl. Rev.)* **35**(4), 476–487 (2005)
13. Saleska, S.R., Didan, K., Huete, A.R., Da Rocha, H.R.: Amazon forests green-up during 2005 drought. *Science* **318**(5850), 612–612 (2007)
14. Schmidt, M., Lipson, H.: Age-fitness Pareto optimization. In: Riolo, R., McConaghy, T., Vladislavleva, E. (eds.) Genetic Programming Theory and Practice VIII. GEVO, vol. 8, pp. 129–146. Springer, New York (2011). [https://doi.org/10.1007/978-1-4419-7747-2\\_8](https://doi.org/10.1007/978-1-4419-7747-2_8)
15. Vladislavleva, E.J., Smits, G.F., den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Trans. Evol. Comput.* **13**(2), 333–349 (2009)
16. Xiao, J., Moody, A.: Geographical distribution of global greening trends and their climatic correlates: 1982–1998. *Int. J. Rem. Sens.* **26**(11), 2371–2390 (2005)
17. Xu, L., Samanta, A., Costa, M.H., Ganguly, S., Nemani, R.R., Myneni, R.B.: Widespread decline in greenness of Amazonian vegetation due to the 2010 drought. *Geophys. Res. Lett.* **38**(7) (2011)
18. Yuan, F., Roy, S.: Analysis of the relationship between NDVI and climate variables in minnesota using geographically weighted regression and spatial interpolation, vol. 2, pp. 784–789 (2007)
19. Zhao, M., Running, S.W.: Drought-induced reduction in global terrestrial net primary production from 2000 through 2009. *Science* **329**(5994), 940–943 (2010)
20. Zhou, L., Tian, Y., Myneni, R.B., Ciais, P., Saatchi, S., et al.: Widespread decline of congo rainforest greenness in the past decade. *Nature* **509**, 86 (2014)

# Author Index

- Aalvanger, G. H. I-146  
Abreu, Salvador I-436  
Aguirre, Hernán II-181, II-232  
Aldana-Montes, José F. I-274, I-298  
Amaya, Ivan II-373  
Antipov, Denis II-117  
Arbonès, Dídac Rodríguez I-512  
Arnold, Dirk V. I-16  
Ashrafzadeh, Homayoon I-451  
Asteroth, Alexander I-500  
Auger, Anne I-3
- Bäck, Thomas I-54, I-500  
Baiolletti, Marco II-436  
Bakurov, Ilyya I-41, I-185  
Barba-González, Cristóbal I-274, I-298  
Barbaresco, Frédéric I-3  
Bartashevich, Palina I-41  
Bartoli, Alberto I-223  
Bartz-Beielstein, Thomas II-220  
Bazzan, Ana II-477  
Benítez-Hidalgo, Antonio I-298  
Bian, Chao II-165  
Blot, Aymeric I-323  
Bongard, Joshua I-525  
Bosman, P. A. N. I-146  
Brabazon, Anthony II-387  
Brockhoff, Dimo I-3  
Browne, Will II-477  
Buzdalov, Maxim I-347
- Castelli, Mauro I-185  
Chen, Gang II-347  
Chicano, Francisco II-449  
Coello Coello, Carlos A. I-298, I-335, I-372, II-373  
Conant-Pablos, Santiago Enrique II-373  
Corus, Dogan II-16, II-67  
Cotta, Carlos I-411  
Covantes Osuna, Edgar II-207  
Cussat-Blanc, Sylvain II-490
- Daniels, Steven J. II-296  
Daolio, Fabio II-257
- Das, Kamalika I-525  
De Lorenzo, Andrea I-223  
de Sá, Alex G. C. II-308  
Deb, Kalyanmoy II-477  
Del Ser, Javier I-298  
Derbel, Bilel II-181, II-232  
Diaz, Daniel I-436  
Ding, Boyin I-512  
Doerr, Benjamin II-117  
Doerr, Carola I-54, II-29, II-360, II-477  
Duan, Qiqi I-424  
Đurasević, Marko II-477  
Durillo, Juan J. I-298
- Eiben, A. E. I-476  
Ekárt, Anikó I-236  
ElHara, Ouassim Ait I-3  
Emmerich, Michael T. M. II-477  
Epitropakis, Michael G. II-477, II-490  
Everson, Richard M. II-296
- Fagan, David I-197  
Falcón-Cardona, Jesús Guillermo I-335  
Fieldsend, Jonathan E. II-296  
Flasch, Oliver II-220  
Fontanella, Francesco I-185  
Forstenlechner, Stefan I-197  
Frahnow, Clemens II-129  
Freitas, Alex A. II-308  
Friedrich, Tobias I-134
- Gallagher, Marcus II-284, II-490  
Ganguly, Sangram I-525  
García, Marcos Diez II-194  
García-Nieto, José I-274, I-298  
García-Valdez, J. Mario I-399  
Ghasemishabankareh, Behrooz I-69  
Glasmachers, Tobias II-411  
Göbel, Andreas I-134  
Griffiths, Thomas D. I-236
- Haasdijk, Evert I-476  
Hagg, Alexander I-500  
Hansen, Nikolaus I-3

- Haqqani, Mohammad I-451  
 Haraldsson, Saemundur O. II-477  
 Hart, Emma I-170, I-488  
 Helsingaun, Keld I-95  
 Herrmann, Sebastian II-245  
 Hirsch, Rachel II-55  
 Hoos, Holger II-271  
 Horn, Daniel II-399  
  
 Igel, Christian I-512  
 Imada, Ryo I-384  
 Ishibuchi, Hisao I-249, I-262, I-311, I-384  
  
 Jakobovic, Domagoj I-121, II-477  
 Jansen, Thomas II-153, II-490  
 Jelisavcic, Milan I-476  
 Jourdan, Laetitia I-323  
 Jurczuk, Krzysztof II-461  
  
 Karunakaran, Deepak II-347  
 Kassab, Rami I-3  
 Kayhani, Arash I-16  
 Kazakov, Dimitar II-321  
 Kerschke, Pascal II-477, II-490  
 Kessaci, Marie-Éléonore I-323  
 Kodali, Anuradha I-525  
 Kordulewski, Hubert I-29  
 Kötzing, Timo II-42, II-79, II-92, II-129  
 Kramer, Oliver II-424  
 Krause, Oswin I-512  
 Krawiec, Krzysztof II-477  
 Krejca, Martin S. II-79, II-92  
 Kretowski, Marek II-461  
  
 Lagodzinski, J. A. Gregor II-42  
 Lan, Gongjin I-476  
 Lardeux, Frédéric I-82  
 Le, Nam II-387  
 Legrand, Pierrick I-209  
 Lehre, Per Kristian II-105, II-477  
 Lengler, Johannes II-3, II-42  
 Leporati, Alberto I-121  
 Li, Xiaodong I-69, I-451, II-477, II-490  
 Liefoghe, Arnaud II-181, II-232  
 Lissvoei, Andrei II-477  
 Liu, Yiping I-262, I-311  
 Lobo, Fernando G. II-490  
 López, Jheisson I-436  
 López, Uriel I-209  
  
 López-Ibáñez, Manuel I-323, II-232, II-321  
 Luong, N. H. I-146  
  
 Malo, Pekka II-477  
 Manoatl Lopez, Edgar I-372  
 Mariot, Luca I-121  
 Markina, Margarita I-347  
 Martí, Luis II-477  
 Masuyama, Naoki I-262, I-311, I-384  
 McDermott, James II-334  
 Medvet, Eric I-223  
 Mei, Yi II-347, II-477  
 Melnichenko, Anna II-42  
 Merelo Guervós, Juan J. I-399, II-477  
 Miettinen, Kaisa I-274, I-286  
 Milani, Alfredo II-436  
 Miller, Julian F. II-477, II-490  
 Moraglio, Alberto II-194, II-334, II-477  
 Mostaghim, Sanaz I-41  
 Mukhopadhyay, Anirban II-55  
 Müller, Nils II-411  
 Múnera, Danny I-436  
  
 Nagata, Yuichi I-108  
 Narvaez-Teran, Valentina I-82  
 Nebro, Antonio J. I-274, I-298, II-477  
 Neumann, Aneta I-158  
 Neumann, Frank I-69, I-158, II-141  
 Nguyen, Phan Trung Hai II-105  
 Nguyen, Su II-477  
 Nicolau, Miguel I-197  
 Noguerras, Rafael I-411  
 Nojima, Yusuke I-262, I-311, I-384  
  
 O'Neill, Michael I-197, II-387  
 Ochoa, Gabriela II-245, II-257, II-477  
 Ojalehto, Vesa I-274  
 Okulewicz, Michał I-29  
 Oliveto, Pietro S. II-16, II-67, II-477, II-490  
 Ortiz-Bayliss, José Carlos II-373  
 Ozlen, Melih I-69  
  
 Paechter, Ben I-170  
 Pappa, Gisele Lobo II-308, II-477  
 Picek, Stjepan I-121, II-477  
 Pillay, Nelishia II-477  
 Pinto, Eduardo Carvalho II-29  
 Prellberg, Jonas II-424  
 Preuss, Mike II-477, II-490

- Purshouse, Robin II-490  
 Pushak, Yasha II-271
- Qian, Chao II-165  
 Quinzan, Francesco I-134
- Rahat, Alma A. M. II-296  
 Reska, Daniel II-461  
 Rodriguez-Tello, Eduardo I-82  
 Roijers, Diederik M. I-476  
 Roostapour, Vahid I-158
- Saleem, Sobia II-284  
 Santucci, Valentino II-436  
 Schoenauer, Marc II-477  
 Semet, Yann I-3  
 Senkerik, Roman II-477  
 Sergiienko, Nataliia Y. I-512  
 Shang, Ke I-262, I-311  
 Sharma, Mudita II-321  
 Shi, Yuhui I-424  
 Shir, Ofer II-477  
 Sinha, Ankur II-477  
 Squillero, Giovanni II-490  
 Stone, Christopher I-170  
 Stork, Jörg II-220  
 Sudholt, Dirk II-207, II-477  
 Sun, Lijun I-424  
 Sutton, Andrew M. II-141  
 Szubert, Marcin I-525
- Tabor, Gavin R. II-296  
 Tagawa, Kiyoharu I-464  
 Tanabe, Ryoji I-249  
 Tanaka, Kiyoshi II-181, II-232  
 Tang, Ke II-165  
 Tarlao, Fabiano I-223
- Terashima-Marín, Hugo II-373  
 Thierens, D. I-146  
 Tinós, Renato I-95, II-449  
 Tomassini, Marco II-257  
 Tonda, Alberto II-490  
 Trujillo, Leonardo I-209
- Uliński, Mateusz I-29  
 Urquhart, Neil I-488
- van Rijn, Sander I-54  
 Vanneschi, Leonardo I-41, I-185  
 Varadarajan, Swetha II-55  
 Varelas, Konstantinos I-3  
 Verel, Sébastien II-181, II-232, II-257
- Wagner, Markus I-134, I-512, II-360, II-490  
 Weise, Thomas II-490  
 Whitley, Darrell I-95, II-55, II-449, II-477  
 Wilson, Dennis II-490  
 Wineberg, Mark II-477  
 Wood, Ian II-284  
 Woodward, John II-477  
 Wróbel, Borys II-490
- Yazdani, Donya II-16, II-67  
 Yu, Xinghuo I-451
- Zaborski, Mateusz I-29  
 Zaefferer, Martin II-220, II-399  
 Zamuda, Aleš II-490  
 Zarges, Christine II-153, II-490  
 Zhang, Hanwei I-359  
 Zhang, Mengjie II-347, II-477  
 Zhou, Aimin I-359  
 Zhou-Kangas, Yue I-286  
 Żychowski, Adam I-29