

Thomas Bäck · Mike Preuss ·  
André Deutz · Hao Wang ·  
Carola Doerr · Michael Emmerich ·  
Heike Trautmann (Eds.)

LNCS 12270

# Parallel Problem Solving from Nature – PPSN XVI

16th International Conference, PPSN 2020  
Leiden, The Netherlands, September 5–9, 2020  
Proceedings, Part II

2  
Part II



 Springer

## Founding Editors

Gerhard Goos

*Karlsruhe Institute of Technology, Karlsruhe, Germany*

Juris Hartmanis

*Cornell University, Ithaca, NY, USA*

## Editorial Board Members

Elisa Bertino

*Purdue University, West Lafayette, IN, USA*

Wen Gao

*Peking University, Beijing, China*

Bernhard Steffen 

*TU Dortmund University, Dortmund, Germany*

Gerhard Woeginger 

*RWTH Aachen, Aachen, Germany*

Moti Yung

*Columbia University, New York, NY, USA*



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

Thomas Bäck · Mike Preuss ·  
André Deutz · Hao Wang ·  
Carola Doerr · Michael Emmerich ·  
Heike Trautmann (Eds.)

# Parallel Problem Solving from Nature – PPSN XVI


16th International Conference, PPSN 2020  
Leiden, The Netherlands, September 5–9, 2020  
Proceedings, Part II

*Editors*

Thomas Bäck   
Leiden University  
Leiden, The Netherlands


André Deutz   
Leiden University  
Leiden, The Netherlands

Carola Doerr   
Sorbonne University  
Paris, France

Heike Trautmann   
University of Münster  
Münster, Germany

Mike Preuss   
Leiden University  
Leiden, The Netherlands

Hao Wang   
Sorbonne University  
Paris, France

Michael Emmerich   
Leiden University  
Leiden, The Netherlands

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-030-58114-5              ISBN 978-3-030-58115-2 (eBook)  
<https://doi.org/10.1007/978-3-030-58115-2>

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer Nature Switzerland AG 2020

Chapter “A Hybrid Evolutionary Algorithm for Reliable Facility Location Problem” is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>). For further details see licence information in the chapter.

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, expressed 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

Welcome to the two volumes of the proceedings of the Conference on Parallel Problem Solving from Nature, PPSN XVI, September 5–9, 2020, Leiden, The Netherlands! When we applied to host PPSN XVI in Leiden, we were not able to imagine anything like the COVID-19 pandemic. Then the new reality hit us, and we were forced to make decisions under uncertain, dynamically changing conditions and constraints, and certainly with multiple, conflicting objectives. Scientific expertise in evolutionary computation was only partially helpful for this. At the time of writing this preface, June 2020, we believed that a hybrid conference format would be the best approach for dealing with the situation: For those who were not able to travel to Leiden, we decided to run PPSN on-site, with printed posters, workshops, tutorials, keynotes, food, and drinks. For those who could not travel to Leiden, we offered it online, with keynote live streams, poster and tutorial videos, and poster discussion rooms in which attendees could discuss with the poster presenters. The virtual part of the conference also allowed participants to meet other attendees online and start a conversation. The challenging and exciting experiment combining the on-site and online world gave attendees the best of both worlds and the flexibility needed in these difficult times – hopefully giving attendees the best of both worlds and the flexibility needed in these difficult times. Not every detail of our hybrid plan turned out as expected, but we are quite sure that some of the changes to conference organization we have tried will remain, and with the help of applied AI and the digitalization of communication, conference experiences in future will not only change but also improve.

PPSN 2020 was also quite a special event since it was the 30th anniversary of the PPSN conference! In particular for Hans-Paul Schwefel, the founder of PPSN, this is a wonderful confirmation of a successful concept – so our congratulations go to you in particular, Hans-Paul. For Thomas Bäck, who was a first-year PhD student of Hans-Paul in 1990, at PPSN I, it is an honor to be involved in this as a general co-chair, and both Mike Preuss and he share the great experience of having been supervised in their PhD studies by Hans-Paul. Although, as Thomas admits, 1990 was easier since the final conference responsibility was with Hans-Paul. We are particularly proud to have Hans-Paul and Grzegorz Rozenberg, the founder and magician of Natural Computing in Leiden, as our honorary chairs for PPSN 2020.

PPSN 2020 received a total of 268 paper submissions written by 690 authors from 44 different countries. Our Program Committee (PC) comprised 271 members from 39 countries. Together, and despite the individual challenges that the coronavirus crisis imposed on each one of us, the PC members wrote 812 review reports in total, which corresponds to an average 3 reviews per paper. Each review was read and evaluated by one of the PC chairs. Where reviewers disagreed in their assessment, a discussion among PC members was started. In some cases, authors were contacted to provide

additional clarification about a technical aspect of their work. In other cases, additional reviews were solicited. The review process resulted in a total number of 99 accepted papers, which corresponds to an acceptance rate of 36.9%. All accepted papers can be found in these LNCS proceedings of PPSN. In addition to the main conference program, an attractive selection of 14 tutorials, 6 workshops, and 3 competitions was offered to participants.

The topics covered classical subjects such as Genetic and Evolutionary Algorithms, Combinatorial Optimization, Multi-objective Optimization, and Real-World Applications of Nature-Inspired Optimization Heuristics. The conference also included quite a number of papers dealing with broader aspects of Artificial Intelligence, reflecting the fact that search and optimization algorithms indeed form an important pillar of modern AI.

As always, PPSN is an interactive forum for inspiring discussions and exchanges, stimulated by on-site and online poster presentations. Three distinguished invited speakers give keynotes at the conference: Carme Torras on assistive and collaborative robotics, Eric Postma on machine learning in image recognition and cognitive modeling, and Christian Stöcker on the direction of AI in general and its effects on society. We are grateful that they accepted our invitation to present their keynotes on-site.

The list of people who made this conference possible is very long, showing the impressive collaborative effort and commitment both of the scientific community that is behind PPSN and of the organizers. This includes all authors, who recognize and acknowledge the scientific quality of this conference series by their submission, and all Program Committee members, who are volunteering although everybody in the community is overloaded with reviewing requests. Our thanks go to the tutorial speakers, workshop organizers, and attendees of the conference and its events.

We are also very grateful for the contributions of the workshop chair, Anna Esparcia-Alcázar, competition chair, Vanessa Volz, and tutorial chair, Ofer Shir. The keynote chair, Aske Plaat, and industrial liaison chair, Bernhard Sendhoff. Our financial chair, Felix Wittleben, who had a difficult time due to the dynamically changing situation. Our publicity chairs, Bas van Stein and Wenjian Luo, who made sure the community heard about PPSN 2020. Our local organization team, Jayshri Murli, Hestia Tamboer, and Roshny Kohabir, who took care of a million things and made the impossible possible. And then, for the conference days, the PhD and master students who helped manage the small but important details. Moreover, all of a sudden, we needed an online conference chair team, for which Bas van Stein, Diederick Vermetten, and Jiawen Kong volunteered to make the online part of the conference happen, and Anna Kononova also joined the team to help with many aspects of the organization. Finally, we would like to express our gratitude to the Leiden Institute of Advanced Computer Science (LIACS), Leiden University for hosting this event, to Leiden University, for its support, particularly to Springer Nature for financing the Best Paper Award, and to the Confederation of Laboratories for Artificial Intelligence

Research in Europe (CLAIRE) and Honda Research Institute Europe GmbH for their invaluable support in countless ways.

Thank you very much to all of you, for making PPSN 2020 possible! We are very proud that we have managed this, under difficult conditions, as a team effort.

July 2020

Thomas Bäck  
Mike Preuss  
General Chairs

André Deutz  
Hao Wang  
Proceedings Chairs

Carola Doerr  
Michael Emmerich  
Heike Trautmann  
Program Chairs

# Organization

PPSN 2020 was organized and hosted by the Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands. Leiden University was founded in 1575 and is the oldest university of The Netherlands. Sixteen persons associated with Leiden University (either as PhD student or (guest-) researcher) became Nobel prize winners and it was the home of many illustrious individuals such as René Descartes, Rembrandt van Rijn, Christiaan Huygens, Hugo Grotius, Baruch Spinoza, and Baron d'Holbach.

## General Chairs

Thomas Bäck	Leiden University, The Netherlands
Mike Preuss	Leiden University, The Netherlands

## Honorary Chairs

Hans-Paul Schwefel	TU Dortmund, Germany
Grzegorz Rozenberg	Leiden University, The Netherlands

## Program Committee Chairs

Carola Doerr	Sorbonne Université, France
Michael Emmerich	Leiden University, The Netherlands
Heike Trautmann	Westfälische Wilhelms-Universität Münster, Germany

## Proceedings Chairs

André Deutz	Leiden University, The Netherlands
Hao Wang	Sorbonne Université, France

## Keynote Chair

Aske Plaat	Leiden University, The Netherlands
------------	------------------------------------

## Workshop Chair

Anna I. Esparcia-Alcázar	SPECIES, Europe
--------------------------	-----------------

## Tutorial Chair

Ofer M. Shir	Tel-Hai College, Israel
--------------	-------------------------

## **Competition Chair**

Vanessa Volz modl.ai, Denmark

## **Industrial Liaison Chair**

Bernhard Sendhoff Honda Research Institute Europe GmbH, Germany

## **Financial Chair**

Felix Wittleben Leiden University, The Netherlands

## **Online Conference Chairs**

Bas van Stein Leiden University, The Netherlands

Diederick Vermetten Leiden University, The Netherlands

Jiawen Kong Leiden University, The Netherlands

## **Publicity Chairs**

Bas van Stein Leiden University, The Netherlands

Wenjian Luo Harbin Institute of Technology, China

## **Local Chair**

Anna V. Kononova Leiden University, The Netherlands

## **Local Organizing Committee**

Jayshri Murli Leiden University, The Netherlands

Roshny Kohabir Leiden University, The Netherlands

Hestia Tamboer Leiden University, The Netherlands

## **Steering Committee**

David W. Corne Heriot-Watt University, UK

Carlos Cotta Universidad de Málaga, Spain

Kenneth De Jong George Mason University, USA

Gusz E. Eiben Vrije Universiteit Amsterdam, The Netherlands

Bogdan Filipič Jožef Stefan Institute, Slovenia

Emma Hart Edinburgh Napier University, UK

Juan Julián Merelo Guervós Universida de Granada, Spain

Günter Rudolph TU Dortmund, Germany

Thomas P. Runarsson University of Iceland, Iceland

Robert Schaefer University of Krakow, Poland

Marc Schoenauer Inria, France

Xin Yao University of Birgmingham, UK



## Keynote Speakers

Carme Torras	Institut de Robòtica i Informàtica Industrial, Spain
Eric Postma	Tilburg University, The Netherlands
Christian Stöcker	Hochschule für Angewandte Wissenschaften Hamburg, Germany

## Program Committee

Michael Affenzeller	Upper Austria University of Applied Sciences, Austria
Hernán Aguirre	Shinshu University, Japan
Youhei Akimoto	University of Tsukuba, Japan
Brad Alexander	The University of Adelaide, Australia
Richard Allmendinger	The University of Manchester, UK
Lucas Almeida	Universidade Federal de Goiás, Brazil
Marie Anastacio	Leiden University, The Netherlands
Denis Antipov	ITMO University, Russia
Dirk Arnold	Dalhousie University, Canada
Dennis Assenmacher	Westfälische Wilhelms-Universität Münster, Germany
Anne Auger	Inria, France
Dogan Aydin	Dumlupinar University, Turkey
Jaume Bacardit	Newcastle University, UK
Samineh Bagheri	TH Köln, Germany
Helio Barbosa	Laboratório Nacional de Computação Científica, Brazil
Thomas Bartz-Beielstein	TH Köln, Germany
Andreas Beham	University of Applied Sciences Upper Austria, Austria
Heder Bernardino	Universidade Federal de Juiz de Fora, Brazil
Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria
Mauro Birattari	Université Libre de Bruxelles, Belgium
Aymeric Blot	University College London, UK
Christian Blum	Spanish National Research Council, Spain
Markus Borschbach	FHDW Bergisch Gladbach, Germany
Peter Bosman	Centrum Wiskunde & Informatica, The Netherlands
Jakob Bossek	The University of Adelaide, Australia
Jürgen Branke	University of Warwick, UK
Dimo Brockhoff	Inria, France
Will Browne	Victoria University of Wellington, New Zealand
Alexander Brownlee	University of Stirling, UK
Larry Bull	University of the West of England, UK
Maxim Buzdalov	ITMO University, Russia
Arina Buzdalova	ITMO University, Russia
Stefano Cagnoni	University of Parma, Italy
Fabio Caraffini	De Montfort University, UK
Matthias Carnein	Westfälische Wilhelms-Universität Münster, Germany
Mauro Castelli	Universidade NOVA de Lisboa, Portugal
Josu Ceberio	University of the Basque Country, Spain

Ying-Ping Chen	National Chiao Tung University, Taiwan
Francisco Chicano	Universidad de Málaga, Spain
Miroslav Chlebik	University of Sussex, UK
Sung-Bae Cho	Yonsei University, South Korea
Tinkle Chugh	University of Exeter, UK
Carlos Coello Coello	CINVESTAV-IPN, Mexico
Dogan Corus	The University of Sheffield, UK
Ernesto Costa	University of Coimbra, Portugal
Carlos Cotta	Universidad de Málaga, Spain
Agostinho Da Rosa	ISR-IST, Portugal
Nguyen Dang	St Andrews University, UK
Kenneth A. De Jong	George Mason University, USA
Kalyanmoy Deb	Michigan State University, USA
Antonio Della-Cioppa	University of Salerno, Italy
Bilel Derbel	University of Lille, France
André Deutz	Leiden University, The Netherlands
Benjamin Doerr	École Polytechnique, France
Carola Doerr	Sorbonne Université, France
John Drake	University of Leicester, UK
Rafal Drezewski	AGH University of Science and Technology, Poland
Paul Dufossé	Inria, France
Tome Eftimov	Jožef Stefan Institute, Slovenia
Gusz E. Eiben	Vrije Universiteit Amsterdam, The Netherlands
Mohamed El Yafrani	Aalborg University, Denmark
Talbi El-Ghazali	University of Lille, France
Michael Emmerich	Leiden University, The Netherlands
Anton Eremeev	Sobolev Institute of Mathematics, Russia
Richard Everson	University of Exeter, UK
Pedro Ferreira	Universidade de Lisboa, Portugal
Jonathan Fieldsend	University of Exeter, UK
Bogdan Filipič	Jožef Stefan Institute, Slovenia
Steffen Finck	Vorarlberg University of Applied Sciences, Austria
Andreas Fischbach	TH Köln, Germany
Peter Fleming	The University of Sheffield, UK
Carlos M. Fonseca	University of Coimbra, Portugal
Marcus Gallagher	The University of Queensland, Australia
José García-Nieto	Universidad de Málaga, Spain
António Gaspar-Cunha	University of Minho, Portugal
Mario Giacobini	University of Torino, Italy
Kyriakos Giannakoglou	National Technical University of Athens, Greece
Tobias Glasmachers	Ruhr-Universität Bochum, Germany
Christian Grimme	Westfälische Wilhelms-Universität Münster, Germany
Roderich Gross	The University of Sheffield, UK
Andreia Guerreiro	University of Coimbra, Portugal
Alexander Hagg	Bonn-Rhein-Sieg University of Applied Sciences, Germany

Jussi Hakanen	University of Jyväskylä, Finland
Julia Handl	The University of Manchester, UK
Jin-Kao Hao	University of Angers, France
Emma Hart	Napier University, UK
Verena Heidrich-Meisner	University of Kiel, Germany
Carlos Henggeler Antunes	University of Coimbra, Portugal
Martin Holena	Academy of Sciences of the Czech Republic, Czech Republic
Christian Igel	University of Copenhagen, Denmark
Dani Irawan	TH Köln, Germany
Hisao Ishibuchi	Osaka Prefecture University, Japan
Christian Jacob	University of Calgary, Canada
Domagoj Jakobovic	University of Zagreb, Croatia
Thomas Jansen	Aberystwyth University, UK
Dreo Johann	THALES Research & Technology, France
Laetitia Jourdan	Inria, LIFL CNRS, France
Bryant Julstrom	St. Cloud State University, USA
George Karakostas	McMaster University, Canada
Edward Keedwell	University of Exeter, UK
Pascal Kerschke	Westfälische Wilhelms-Universität Münster, Germany
Marie-Eleonore Kessaci	University of Lille, France
Ahmed Kheiri	Lancaster University, UK
Wolfgang Konen	TH Köln, Germany
Anna Kononova	Leiden University, The Netherlands
Peter Korošec	Jožef Stefan Institute, Slovenia
Lars Kotthoff	University of Wyoming, USA
Oliver Kramer	Universität Oldenburg, Germany
Oswin Krause	University of Copenhagen, Denmark
Krzysztof Krawiec	Poznan University of Technology, Poland
Martin S. Krejca	Hasso-Plattner-Institut, Germany
Timo Kötzing	Hasso-Plattner-Institut, Germany
William La Cava	University of Pennsylvania, USA
Jörg Lässig	University of Applied Sciences Zittau/Görlitz, Germany
William B. Langdon	University College London, UK
Algirdas Lančinskas	Vilnius University, Lithuania
Frederic Lardeux	LERIA, University of Angers, France
Per Kristian Lehre	University of Birmingham, UK
Johannes Lengler	ETH Zurich, Switzerland
Ke Li	University of Exeter, UK
Arnaud Liefoghe	University of Lille, France
Marius Lindauer	Leibniz Universität Hannover, Germany
Giosuè Lo Bosco	Università di Palermo, Italy
Fernando Lobo	University of Algarve, Portugal
Daniele Loiacono	Politecnico di Milano, Italy
Nuno Lourenço	University of Coimbra, Portugal

Jose A. Lozano	University of the Basque Country, Spain
Rodica Ioana Lung	Babeş-Bolyai University, Romania
Chuan Luo	Peking University, China
Gabriel Luque	Universidad de Málaga, Spain
Evelyne Lutton	INRAE, France
Manuel López-Ibáñez	The University of Manchester, UK
Penousal Machado	University of Coimbra, Portugal
Luigi Malagò	Romanian Institute of Science and Technology, Romania
Katherine Malan	University of South Africa, South Africa
Vittorio Maniezzo	University Bologna, Italy
Elena Marchiori	Radboud University, The Netherlands
Luis Marti	Inria, Chile
Asep Maulana	Tilburg University, The Netherlands
Giancarlo Mauri	University of Milano-Bicocca, Italy
Jacek Mańdziuk	Warsaw University of Technology, Poland
James McDermott	National University of Ireland, Ireland
Jörn Mehnen	University of Strathclyde, UK
Alexander Melkozerov	Tomsk State University of Control Systems and Radioelectronics, Russia
Juan J. Merelo	University of Granada, Spain
Marjan Mernik	University of Maribor, Slovenia
Silja Meyer-Nieberg	Bundeswehr Universität München, Germany
Efrén Mezura-Montes	University of Veracruz, Mexico
Krzysztof Michalak	Wroclaw University of Economics, Poland
Kaisa Miettinen	University of Jyväskylä, Finland
Julian Miller	University of York, UK
Edmondo Minisci	University of Strathclyde, UK
Gara Miranda	University of La Laguna, Spain
Mustafa Misir	Istinye University, Turkey
Marco A. Montes De Oca	clypd, Inc., USA
Sanaz Mostaghim	Otto von Guericke Universität Magdeburg, Germany
Mario Andrés Muñoz Acosta	The University of Melbourne, Australia
Boris Naujoks	TH Köln, Germany
Antonio J. Nebro	Universidad de Málaga, Spain
Ferrante Neri	University of Nottingham, UK
Aneta Neumann	The University of Adelaide, Australia
Frank Neumann	The University of Adelaide, Australia
Phan Trung Hai Nguyen	University of Birmingham, UK
Miguel Nicolau	University College Dublin, Ireland
Ellen Norgård-Hansen	NORCE, Norway
Michael O'Neill	University College Dublin, Ireland
Gabriela Ochoa	University of Stirling, UK
Pietro S. Oliveto	The University of Sheffield, UK
Unamay O'Reilly	MIT, USA

José Carlos Ortiz-Bayliss	Tecnológico de Monterrey, Mexico
Patryk Orzechowski	University of Pennsylvania, USA
Ender Ozcan	University of Nottingham, UK
Ben Paechter	Napier University, UK
Gregor Papa	Jožef Stefan Institute, Slovenia
Gisele Pappa	UFMG, Brazil
Luis Paquete	University of Coimbra, Portugal
Andrew J. Parkes	University of Nottingham, UK
Mario Pavone	University of Catania, Italy
David Pelta	University of Granada, Spain
Leslie Perez-Caceres	Pontificia Universidad Católica de Valparaíso, Chile
Stjepan Picek	Delft University of Technology, The Netherlands
Martin Pilat	Charles University, Czech Republic
Nelishia Pillay	University of KwaZulu-Natal, South Africa
Petr Pošík	Czech Technical University in Prague, Czech Republic
Raphael Prager	Westfälische Wilhelms-Universität Münster, Germany
Mike Preuss	Leiden University, The Netherlands
Chao Qian	University of Science and Technology of China, China
Alma Rahat	Swansea University, UK
Günther Raidl	University of Vienna, Austria
William Rand	North Carolina State University, USA
Khaled Rasheed	University of Georgia, USA
Tapabrata Ray	University of New South Wales, Australian Defence Force Academy, Australia
Frederik Rehbach	TH Köln, Germany
Eduardo Rodriguez-Tello	CINVESTAV-Tamaulipas, Mexico
Andrea Roli	University of Bologna, Italy
Jonathan Rowe	University of Birmingham, UK
Günter Rudolph	TU Dortmund, Germany
Thomas A. Runkler	Siemens Corporate Technology, Germany
Conor Ryan	University of Limerick, Ireland
Frédéric Saubion	University of Angers, France
Robert Schaefer	AGH University of Science and Technology, Poland
Andrea Schaerf	University of Udine, Italy
David Schaffer	Binghamton University, USA
Manuel Schmitt	Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
Marc Schoenauer	Inria, France
Oliver Schütze	CINVESTAV-IPN, Mexico
Michèle Sebag	Université Paris-Sud, France
Eduardo Segredo	Universidad de La Laguna, Spain
Moritz Seiler	Westfälische Wilhelms-Universität Münster, Germany
Bernhard Sendhoff	Honda Research Institute Europe GmbH, Germany
Marc Sevaux	Université de Bretagne Sud, France
Jonathan Shapiro	The University of Manchester, UK
Ofer M. Shir	Tel-Hai College, Israel

Shinichi Shirakawa	Yokohama National University, Japan
Moshe Sipper	Ben-Gurion University of the Negev, Israel
Jim Smith	University of the West of England, UK
Christine Solnon	CITI Inria and INSA Lyon, France
Patrick Spettel	Vorarlberg University of Applied Sciences, Germany
Giovanni Squillero	Politecnico di Torino, Italy
Sebastian Urban Stich	École Polytechnique Fédérale de Lausanne, Switzerland
Catalin Stoean	University of Craiova, Romania
Jörg Stork	TH Köln, Germany
Thomas Stützel	Université Libre de Bruxelles, Belgium
Mihai Suciuc	Babeş-Bolyai University, Romania
Dirk Sudholt	The University of Sheffield, UK
Andrew Sutton	University of Minnesota, USA
Jerry Swan	University of York, UK
Ricardo H. C. Takahashi	Universidade Federal de Minas Gerais, Brazil
Daniel Tauritz	Auburn University, USA
Olivier Teytaud	Inria, France
Dirk Thierens	Utrecht University, The Netherlands
Sarah Thomson	University of Stirling, UK
Kevin Tierney	Universität Bielefeld, Germany
Renato Tinós	University of São Paulo, Brazil
Julian Togelius	New York University, USA
Marco Tomassini	University of Lausanne, Switzerland
Alberto Tonda	INRA, France
Cheikh Touré	Inria, France
Heike Trautmann	Westfälische Wilhelms-Universität Münster, Germany
Leonardo Trujillo	Instituto Tecnológico de Tijuana, Mexico
Tea Tušar	Jožef Stefan Institute, Slovenia
Ryan J. Urbanowicz	University of Pennsylvania, USA
Koen van der Blom	Leiden University, The Netherlands
Bas van Stein	Leiden University, The Netherlands
Leonardo Vanneschi	Universidade de NOVA de Lisboa, Portugal
Sébastien Verel	Université du Littoral Côte d'Opale, France
Diederick Vermetten	Leiden University, The Netherlands
Marco Virgolin	Centrum Wiskunde & Informatica, The Netherlands
Vanessa Volz	modl.ai, Denmark
Markus Wagner	The University of Adelaide, Australia
Stefan Wagner	University of Applied Sciences Upper Austria, Austria
David Walker	University of Plymouth, UK
Hao Wang	Sorbonne Université, France
Hui Wang	Leiden University, The Netherlands
Yali Wang	Leiden University, The Netherlands
Elizabeth Wanner	CEFET, Brazil
Thomas Weise	University of Science and Technology of China, China
Dennis Wilson	ISAE-Supaero, France

Carsten Witt	Technical University of Denmark, Denmark
Man Leung Wong	Lingnan University, China
John Woodward	Queen Mary University of London, UK
Ning Xiong	Mälardalen University, Sweden
Bing Xue	Victoria University of Wellington, New Zealand
Kaifeng Yang	University of Applied Sciences Upper Austria, Austria
Shengxiang Yang	De Montfort University, UK
Furong Ye	Leiden University, The Netherlands
Martin Zaefferer	TH Köln, Germany
Ales Zamuda	University of Maribor, Slovenia
Christine Zarges	Aberystwyth University, UK
Mengjie Zhang	Victoria University of Wellington, New Zealand

## Contents – Part II

### Genetic Programming

- Generation of New Scalarizing Functions Using Genetic Programming . . . . . 3  
*Amin V. Bernabé Rodríguez and Carlos A. Coello Coello*
- The Usability Argument for Refinement Typed Genetic Programming . . . . . 18  
*Alcides Fonseca, Paulo Santos, and Sara Silva*
- Program Synthesis in a Continuous Space Using Grammars  
and Variational Autoencoders . . . . . 33  
*David Lynch, James McDermott, and Michael O’Neill*
- Cooperative Co-Evolutionary Genetic Programming  
for High Dimensional Problems . . . . . 48  
*Lino Rodríguez-Coayahuil, Alicia Morales-Reyes, Hugo Jair Escalante,  
and Carlos A. Coello Coello*
- Image Feature Learning with Genetic Programming. . . . . 63  
*Stefano Ruberto, Valerio Terragni, and Jason H. Moore*
- Learning a Formula of Interpretability to Learn Interpretable Formulas . . . . . 79  
*Marco Virgolin, Andrea De Lorenzo, Eric Medvet,  
and Francesca Randone*

### Landscape Analysis

- On Stochastic Fitness Landscapes: Local Optimality and Fitness Landscape  
Analysis for Stochastic Search Operators . . . . . 97  
*Brahim Aboutaib, Sébastien Verel, Cyril Fonlupt, Bilel Derbel,  
Arnaud Liefoghe, and Belaïd Ahiod*
- Fitness Landscape Analysis of Dimensionally-Aware Genetic Programming  
Featuring Feynman Equations. . . . . 111  
*Marko Durasevic, Domagoj Jakobovic,  
Marcella Scoczynski Ribeiro Martins, Stjepan Picek,  
and Markus Wagner*
- Global Landscape Structure and the Random MAX-SAT Phase Transition . . . 125  
*Gabriela Ochoa, Francisco Chicano, and Marco Tomassini*



Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy . . . . .	139
<i>Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr</i>	
One PLOT to Show Them All: Visualization of Efficient Sets in Multi-objective Landscapes. . . . .	154
<i>Lennart Schäpermeier, Christian Grimme, and Pascal Kerschke</i>	
<b>Multi-objective Optimization</b>	
On Sharing Information Between Sub-populations in MOEA/S. . . . .	171
<i>Lucas de Almeida Ribeiro, Michael Emmerich, Anderson da Silva Soares, and Telma Woerle de Lima</i>	
Multi-objective Optimization by Uncrowded Hypervolume Gradient Ascent . . . . .	186
<i>Timo M. Deist, Stefanus C. Maree, Tanja Alderliesten, and Peter A. N. Bosman</i>	
An Ensemble Indicator-Based Density Estimator for Evolutionary Multi-objective Optimization . . . . .	201
<i>Jesús Guillermo Falcón-Cardona, Arnaud Liefooghe, and Carlos A. Coello Coello</i>	
Ensuring Smoothly Navigable Approximation Sets by Bézier Curve Parameterizations in Evolutionary Bi-objective Optimization. . . . .	215
<i>Stefanus C. Maree, Tanja Alderliesten, and Peter A. N. Bosman</i>	
Many-Objective Test Database Generation for SQL. . . . .	229
<i>Zhilei Ren, Shaozheng Dong, Xiaochen Li, Zongzheng Chi, and He Jiang</i>	
A New Paradigm in Interactive Evolutionary Multiobjective Optimization . . .	243
<i>Bhupinder Singh Saini, Jussi Hakanen, and Kaisa Miettinen</i>	
Hypervolume Optimal $\mu$ -Distributions on Line-Based Pareto Fronts in Three Dimensions . . . . .	257
<i>Ke Shang, Hisao Ishibuchi, Weiyu Chen, and Lukáš Adam</i>	
Adaptive Operator Selection Based on Dynamic Thompson Sampling for MOEA/D . . . . .	271
<i>Lei Sun and Ke Li</i>	
A Study of Swarm Topologies and Their Influence on the Performance of Multi-Objective Particle Swarm Optimizers . . . . .	285
<i>Diana Cristina Valencia-Rodríguez and Carlos A. Coello Coello</i>	

Visualising Evolution History in Multi- and Many-objective Optimisation . . .	299
<i>Mathew J. Walter, David J. Walker, and Matthew J. Craven</i>	
Improving Many-Objective Evolutionary Algorithms by Means of Edge-Rotated Cones . . . . .	313
<i>Yali Wang, André Deutz, Thomas Bäck, and Michael Emmerich</i>	
<b>Real-World Applications</b>	
Human-Like Summaries from Heterogeneous and Time-Windowed Software Development Artefacts . . . . .	329
<i>Mahfouth Alghamdi, Christoph Treude, and Markus Wagner</i>	
A Search for Additional Structure: The Case of Cryptographic S-boxes . . . . .	343
<i>Claude Carlet, Marko Djurasevic, Domagoj Jakobovic, and Stjepan Picek</i>	
Evolutionary Multi-objective Design of SARS-CoV-2 Protease Inhibitor Candidates . . . . .	357
<i>Tim Cofala, Lars Elend, Philip Mirbach, Jonas Prellberg, Thomas Teusch, and Oliver Kramer</i>	
Generic Relative Relations in Hierarchical Gene Expression Data Classification . . . . .	372
<i>Marcin Czajkowski, Krzysztof Jurczuk, and Marek Kretowski</i>	
A Variable Neighborhood Search for the Job Sequencing with One Common and Multiple Secondary Resources Problem . . . . .	385
<i>Thomas Kaufmann, Matthias Horn, and Günther R. Raidl</i>	
Evolutionary Graph-Based V+E Optimization for Protection Against Epidemics . . . . .	399
<i>Krzysztof Michalak</i>	
Human-Derived Heuristic Enhancement of an Evolutionary Algorithm for the 2D Bin-Packing Problem . . . . .	413
<i>Nicholas Ross, Ed Keedwell, and Dragan Savic</i>	
Towards Novel Meta-heuristic Algorithms for Dynamic Capacitated Arc Routing Problems . . . . .	428
<i>Hao Tong, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao</i>	
Robust Evolutionary Bi-objective Optimization for Prostate Cancer Treatment with High-Dose-Rate Brachytherapy . . . . .	441
<i>Marjolein C. van der Meer, Arjan Bel, Yury Niatsetski, Tanja Alderliesten, Bradley R. Pieters, and Peter A. N. Bosman</i>	

A Hybrid Evolutionary Algorithm for Reliable Facility Location Problem . . . 454  
*Han Zhang, Jialin Liu, and Xin Yao*

**Reinforcement Learning**

Optimality-Based Analysis of XCSF Compaction in Discrete Reinforcement Learning. . . . . 471  
*Jordan T. Bishop and Marcus Gallagher*

Hybridizing the 1/5-th Success Rule with Q-Learning for Controlling the Mutation Rate of an Evolutionary Algorithm. . . . . 485  
*Arina Buzdalova, Carola Doerr, and Anna Rodionova*

Fitness Landscape Features and Reward Shaping in Reinforcement Learning Policy Spaces . . . . . 500  
*Nathaniel du Preez-Wilkinson and Marcus Gallagher*

ClipUp: A Simple and Powerful Optimizer for Distribution-Based Policy Evolution. . . . . 515  
*Nihat Engin Toklu, Paweł Liskowski, and Rupesh Kumar Srivastava*

Warm-Start AlphaZero Self-play Search Enhancements . . . . . 528  
*Hui Wang, Mike Preuss, and Aske Plaat*

**Theoretical Aspects of Nature-Inspired Optimization**

Runtime Analysis of a Heavy-Tailed  $(1 + (\lambda, \lambda))$  Genetic Algorithm on Jump Functions . . . . . 545  
*Denis Antipov and Benjamin Doerr*

First Steps Towards a Runtime Analysis When Starting with a Good Solution . . . . . 560  
*Denis Antipov, Maxim Buzdalov, and Benjamin Doerr*

Optimal Mutation Rates for the  $(1 + \lambda)$  EA on OneMax . . . . . 574  
*Maxim Buzdalov and Carola Doerr*

Maximizing Submodular or Monotone Functions Under Partition Matroid Constraints by Multi-objective Evolutionary Algorithms. . . . . 588  
*Anh Viet Do and Frank Neumann*

Lower Bounds for Non-elitist Evolutionary Algorithms via Negative Multiplicative Drift. . . . . 604  
*Benjamin Doerr*

<p>Exponential Upper Bounds for the Runtime of Randomized Search Heuristics . . . . .</p> <p style="padding-left: 2em;"><i>Benjamin Doerr</i></p>	619
<p>Analysis on the Efficiency of Multifactorial Evolutionary Algorithms . . . . .</p> <p style="padding-left: 2em;"><i>Zhengxin Huang, Zefeng Chen, and Yuren Zhou</i></p>	634
<p>Improved Fixed-Budget Results via Drift Analysis . . . . .</p> <p style="padding-left: 2em;"><i>Timo Kötzing and Carsten Witt</i></p>	648
<p>On Averaging the Best Samples in Evolutionary Computation . . . . .</p> <p style="padding-left: 2em;"><i>Laurent Meunier, Yann Chevaleyre, Jeremy Rapin, Clément W. Royer, and Olivier Teytaud</i></p>	661
<p>Filter Sort Is <math>\Omega(N^3)</math> in the Worst Case . . . . .</p> <p style="padding-left: 2em;"><i>Sumit Mishra and Maxim Buzdalov</i></p>	675
<p>Approximation Speed-Up by Quadratzation on LeadingOnes . . . . .</p> <p style="padding-left: 2em;"><i>Andrew M. Sutton and Darrell Whitley</i></p>	686
<p>Benchmarking a <math>(\mu + \lambda)</math> Genetic Algorithm with Configurable Crossover Probability . . . . .</p> <p style="padding-left: 2em;"><i>Furong Ye, Hao Wang, Carola Doerr, and Thomas Bäck</i></p>	699
<p><b>Author Index</b> . . . . .</p>	715

# Contents – Part I

## Automated Algorithm Selection and Configuration

Evolving Deep Forest with Automatic Feature Extraction for Image Classification Using Genetic Programming . . . . .	3
<i>Ying Bi, Bing Xue, and Mengjie Zhang</i>	
Fast Perturbative Algorithm Configurators . . . . .	19
<i>George T. Hall, Pietro S. Oliveto, and Dirk Sudholt</i>	
Dominance, Indicator and Decomposition Based Search for Multi-objective QAP: Landscape Analysis and Automated Algorithm Selection. . . . .	33
<i>Arnaud Liefooghe, Sébastien Verel, Bilel Derbel, Hernan Aguirre, and Kiyoshi Tanaka</i>	
Deep Learning as a Competitive Feature-Free Approach for Automated Algorithm Selection on the Traveling Salesperson Problem . . . . .	48
<i>Moritz Seiler, Janina Pohl, Jakob Bossek, Pascal Kerschke, and Heike Trautmann</i>	
Automatic Configuration of a Multi-objective Local Search for Imbalanced Classification . . . . .	65
<i>Sara Tari, Holger Hoos, Julie Jacques, Marie-Eléonore Kessaci, and Laetitia Jourdan</i>	

## Bayesian- and Surrogate-Assisted Optimization

Multi-fidelity Optimization Approach Under Prior and Posterior Constraints and Its Application to Compliance Minimization. . . . .	81
<i>Youhei Akimoto, Naoki Sakamoto, and Makoto Ohtani</i>	
Model-Based Algorithm Configuration with Default-Guided Probabilistic Sampling . . . . .	95
<i>Marie Anastacio and Holger Hoos</i>	
Evolving Sampling Strategies for One-Shot Optimization Tasks . . . . .	111
<i>Jakob Bossek, Carola Doerr, Pascal Kerschke, Aneta Neumann, and Frank Neumann</i>	
A Surrogate-Assisted Evolutionary Algorithm with Random Feature Selection for Large-Scale Expensive Problems . . . . .	125
<i>Guoxia Fu, Chaoli Sun, Ying Tan, Guochen Zhang, and Yaochu Jin</i>	

Designing Air Flow with Surrogate-Assisted Phenotypic Niching . . . . .	140
<i>Alexander Hagg, Dominik Wilde, Alexander Asteroth, and Thomas Bäck</i>	
Variance Reduction for Better Sampling in Continuous Domains . . . . .	154
<i>Laurent Meunier, Carola Doerr, Jeremy Rapin, and Olivier Teytaud</i>	
High Dimensional Bayesian Optimization Assisted by Principal Component Analysis. . . . .	169
<i>Elena Raponi, Hao Wang, Mariusz Bujny, Simonetta Boria, and Carola Doerr</i>	
Simple Surrogate Model Assisted Optimization with Covariance Matrix Adaptation. . . . .	184
<i>Lauchlan Toal and Dirk V. Arnold</i>	
<b>Benchmarking and Performance Measures</b>	
Proposal of a Realistic Many-Objective Test Suite. . . . .	201
<i>Weiyu Chen, Hisao Ishibuchi, and Ke Shang</i>	
Approximate Hypervolume Calculation with Guaranteed or Confidence Bounds . . . . .	215
<i>A. Jaskiewicz, R. Susmaga, and P. Zielniewicz</i>	
Can Compact Optimisation Algorithms Be Structurally Biased? . . . . .	229
<i>Anna V. Kononova, Fabio Caraffini, Hao Wang, and Thomas Bäck</i>	
Parallelized Bayesian Optimization for Expensive Robot Controller Evolution . . . . .	243
<i>Margarita Rebolledo, Frederik Rehbach, A. E. Eiben, and Thomas Bartz-Beielstein</i>	
Revisiting Population Models in Differential Evolution on a Limited Budget of Evaluations . . . . .	257
<i>Ryoji Tanabe</i>	
Continuous Optimization Benchmarks by Simulation. . . . .	273
<i>Martin Zaefferer and Frederik Rehbach</i>	
Comparative Run-Time Performance of Evolutionary Algorithms on Multi-objective Interpolated Continuous Optimisation Problems . . . . .	287
<i>Alexandru-Ciprian Zăvoianu, Benjamin Lacroix, and John McCall</i>	

**Combinatorial Optimization**

On the Design of a Partition Crossover for the Quadratic Assignment Problem . . . . . 303  
*Omar Abdelkafi, Bilel Derbel, Arnaud Liefoghe, and Darrell Whitley*

A Permutational Boltzmann Machine with Parallel Tempering for Solving Combinatorial Optimization Problems . . . . . 317  
*Mohammad Bagherbeik, Parastoo Ashtari, Seyed Farzad Mousavi, Kouichi Kanda, Hirotaka Tamura, and Ali Sheikholeslami*

Solution Repair by Inequality Network Propagation in LocalSolver . . . . . 332  
*Léa Blaise, Christian Artigues, and Thierry Benoist*

Optimising Tours for the Weighted Traveling Salesperson Problem and the Traveling Thief Problem: A Structural Comparison of Solutions . . . . 346  
*Jakob Bossek, Aneta Neumann, and Frank Neumann*

Decentralized Combinatorial Optimization . . . . . 360  
*Lee A. Christie*

PbO-CCSAT: Boosting Local Search for Satisfiability Using Programming by Optimisation . . . . . 373  
*Chuan Luo, Holger Hoos, and Shaowei Cai*

Evaluation of a Permutation-Based Evolutionary Framework for Lyndon Factorizations . . . . . 390  
*Lily Major, Amanda Clare, Jacqueline W. Daykin, Benjamin Mora, Leonel Jose Peña Gamboa, and Christine Zarges*

Optimising Monotone Chance-Constrained Submodular Functions Using Evolutionary Multi-objective Algorithms . . . . . 404  
*Aneta Neumann and Frank Neumann*

Parameter-Less Population Pyramid for Permutation-Based Problems. . . . . 418  
*Szymon Wozniak, Michal W. Przewozniczek, and Marcin M. Komarnicki*

**Connection Between Nature-Inspired Optimization and Artificial Intelligence**

Biologically Plausible Learning of Text Representation with Spiking Neural Networks . . . . . 433  
*Marcin Białas, Marcin Michał Mironczuk, and Jacek Mańdziuk*

Multi-Objective Counterfactual Explanations . . . . . 448  
*Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl*

Multi-objective Magnitude-Based Pruning for Latency-Aware Deep Neural Network Compression . . . . .	470
<i>Wenjing Hong, Peng Yang, Yiwen Wang, and Ke Tang</i>	
Network Representation Learning Based on Topological Structure and Vertex Attributes. . . . .	484
<i>Shengxiang Hu, Bofeng Zhang, Ying Lv, Furong Chang, and Zhuocheng Zhou</i>	
A Committee of Convolutional Neural Networks for Image Classification in the Concurrent Presence of Feature and Label Noise . . . . .	498
<i>Stanisław Kazmierczak and Jacek Mańdziuk</i>	
Improving Imbalanced Classification by Anomaly Detection. . . . .	512
<i>Jiawen Kong, Wojtek Kowalczyk, Stefan Menzel, and Thomas Bäck</i>	
BACS: A Thorough Study of Using Behavioral Sequences in ACS2 . . . . .	524
<i>Romain Orhand, Anne Jeannin-Girardon, Pierre Parrend, and Pierre Collet</i>	
Nash Equilibrium as a Solution in Supervised Classification. . . . .	539
<i>Mihai-Alexandru Suciú and Rodica Ioana Lung</i>	
Analyzing the Components of Distributed Coevolutionary GAN Training. . . . .	552
<i>Jamal Toutouh, Erik Hemberg, and Una-May O'Reilly</i>	
Canonical Correlation Discriminative Learning for Domain Adaptation . . . . .	567
<i>Wenjing Wang, Yuwu Lu, and Zhihui Lai</i>	
<b>Genetic and Evolutionary Algorithms</b>	
Improving Sampling in Evolution Strategies Through Mixture-Based Distributions Built from Past Problem Instances . . . . .	583
<i>Stephen Friess, Peter Tiño, Stefan Menzel, Bernhard Sendhoff, and Xin Yao</i>	
The Hessian Estimation Evolution Strategy . . . . .	597
<i>Tobias Glasmachers and Oswin Krause</i>	
Large Population Sizes and Crossover Help in Dynamic Environments . . . . .	610
<i>Johannes Lengler and Jonas Meier</i>	
Neuromemetic Evolutionary Optimization . . . . .	623
<i>Paweł Liskowski, Krzysztof Krawiec, and Nihat Engin Toklu</i>	
Evolved Gossip Contracts - A Framework for Designing Multi-agent Systems . . . . .	637
<i>Nicola Mc Donnell, Enda Howley, and Jim Duggan</i>	



A SHADE-Based Algorithm for Large Scale Global Optimization. . . . . 650  
*Oscar Pacheco-Del-Moral and Carlos A. Coello Coello*

Evolutionary Algorithms with Self-adjusting Asymmetric Mutation . . . . . 664  
*Amirhossein Rajabi and Carsten Witt*

Behavior Optimization in Large Distributed Systems Modeled  
 by Cellular Automata . . . . . 678  
*Franciszek Seredyński and Jakub Gąsior*

Learning Step-Size Adaptation in CMA-ES . . . . . 691  
*Gresa Shala, André Biedenkapp, Noor Awad, Steven Adriaensen,  
 Marius Lindauer, and Frank Hutter*

Sparse Inverse Covariance Learning for CMA-ES with Graphical Lasso . . . . 707  
*Konstantinos Varelas, Anne Auger, and Nikolaus Hansen*

Adaptive Stochastic Natural Gradient Method for Optimizing Functions  
 with Low Effective Dimensionality . . . . . 719  
*Tepei Yamaguchi, Kento Uchida, and Shinichi Shirakawa*

**Author Index** . . . . . 733

# **Genetic Programming**



# Generation of New Scalarizing Functions Using Genetic Programming

Amín V. Bernabé Rodríguez<sup>(✉)</sup> and Carlos A. Coello Coello<sup>(ID)</sup>

CINVESTAV-IPN (Evolutionary Computation Group),  
Av. IPN 2508, 07360 Mexico City, San Pedro Zacatenco, Mexico  
abernabe@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

**Abstract.** In recent years, there has been a growing interest in multi-objective evolutionary algorithms (MOEAs) with a selection mechanism different from Pareto dominance. This interest has been mainly motivated by the poor performance of Pareto-based selection mechanisms when dealing with problems having more than three objectives (the so-called many-objective optimization problems). Two viable alternatives for solving many-objective optimization problems are decomposition-based and indicator-based MOEAs. However, it is well-known that the performance of decomposition-based MOEAs (and also of indicator-based MOEAs designed around  $R2$ ) heavily relies on the scalarizing function adopted. In this paper, we propose an approach for generating novel scalarizing functions using genetic programming. Using our proposed approach, we were able to generate two new scalarizing functions (called *AGSF1* and *AGSF2*), which were validated using an indicator-based MOEA designed around  $R2$  (MOMBI-II). This validation was conducted using a set of standard test problems and two performance indicators (hypervolume and  $s$ -energy). Our results indicate that *AGSF1* has a similar performance to that obtained when using the well-known *Achievement Scalarizing Function (ASF)*. However, *AGSF2* provided a better performance than *ASF* in most of the test problems adopted. Nevertheless, our most remarkable finding is that genetic programming can indeed generate novel (and possible more competitive) scalarizing functions.

**Keywords:** Multi-objective optimization · Genetic programming · Scalarizing functions

## 1 Introduction

A great variety of real-world problems require the simultaneous optimization of two or more (often conflicting) objective functions. These are known as

---

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 grant no. 2016-01-1920 (*Investigación en Fronteras de la Ciencia 2016*) and from a SEP-Cinvestav grant (application no. 4).

Multi-objective Optimization Problems (MOPs) and are mathematically defined as follows:

$$\min_{\mathbf{x} \in \Omega} \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}))^T \quad (1)$$

where  $\mathbf{x} \in \mathbb{R}^n$  is the vector of decision variables,  $\Omega \subset \mathbb{R}^n$  is the decision variable space and  $\mathbf{F}(\mathbf{x})$  is the vector of  $m$  objective functions, with  $m \geq 2$ .

When solving an MOP, the goal is to find the set of points that yield the best possible trade-offs among the objective functions. These points are known as Pareto optimal solutions, and they form the Pareto Optimal Set ( $\mathcal{P}^*$ ) of the problem. Its image in objective space is known as the Pareto Optimal Front ( $\mathcal{PF}^*$ ).

The use of evolutionary algorithms for solving MOPs (the so-called Multi-Objective Evolutionary Algorithms, or MOEAs) has become increasingly popular in recent years. MOEAs are population-based methods that allow obtaining a set of different Pareto optimal solutions in a single run, in contrast with traditional mathematical programming techniques, which normally generate a single element of the Pareto optimal set per run [3].

Many MOEAs have been proposed in the literature, but they can be broadly classified into 3 categories: (1) Pareto-based, (2) indicator-based and (3) decomposition-based MOEAs [13]. The work reported in this paper is particularly relevant for decomposition-based MOEAs, but it is also applicable for some indicator-based MOEAs that rely on scalarizing functions (e.g., those based on the  $R2$  indicator). Decomposition-based MOEAs decompose an MOP into several single-objective optimization problems, which are simultaneously solved [14]. In order to perform this decomposition, a scalarizing function is adopted. A scalarizing function (also known as utility function or aggregation function), transforms the original MOP into a single-objective problem using a predefined target direction or weights vector. There is empirical evidence that indicates that the performance of MOEAs that rely on scalarizing functions strongly depends on the particular scalarizing function adopted [12]. Consequently, it is relevant to find new scalarizing functions which should have a comparable performance or even better (at least in certain types of MOPs) than the scalarizing functions that are currently being used.

This paper proposes a strategy to evolve scalarizing functions combining two heuristics: genetic programming (GP) to create new functions and an MOEA to evaluate their corresponding fitness. Using the proposed approach, we were able to generate two new scalarizing functions and we compared their performance with respect to that obtained using the well-known *Achievement Scalarizing Function* (ASF). As will be seen later in this paper, our experimental results show that the scalarizing functions generated by our proposed approach have a similar performance, and that one of them outperforms ASF in more than half of the test problems adopted.

The remainder of this paper is organized as follows. Section 2 describes our approach for generating new scalarizing functions using genetic programming. Section 3 presents the experimental results obtained when assessing performance

of an MOEA using the new scalarizing functions generated by our proposed approach. Section 4 provides our conclusions and some potential paths for future work.

## 2 Our Proposed Approach

Genetic programming (GP) is a well-established evolutionary algorithm proposed by Koza [10], in which individuals encode computer programs [1]. Although trees are the most traditional data structure adopted by GP, over the years a variety of other data structures have been adopted as well (e.g., arrays, lists and graphs).

---

### Algorithm 1: Main procedure of our proposed approach

---

```

Input : MOP,  $t_{max}$ ;
Output: Final population  $P$ ;
1  $t \leftarrow 1$ ;
2 Randomly initialize the population  $P = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ ;
3 foreach  $\mathbf{x}_i \in P$  do
4    $sf_i \leftarrow$  decode genotype from  $\mathbf{x}_i$ ;
5    $\mathbf{x}_i.fitness \leftarrow$  MOEAFitness(MOP,  $sf_i$ );
6 end
7 while the stopping criterion is not met do
8    $P' \leftarrow$  select and recombine parents from  $P$ ;
9   foreach  $\mathbf{x}_i \in P'$  do
10     $sf_i \leftarrow$  decode genotype from  $\mathbf{x}_i$ ;
11     $\mathbf{x}_i.fitness \leftarrow$  MOEAFitness(MOP,  $sf_i$ );
12  end
13   $P \leftarrow P'$ ;
14   $t \leftarrow t + 1$ ;
15 end
16 return  $P$ ;

```

---

Epigenetic Linear Genetic Programming (ELGP)<sup>1</sup> is an implementation of GP coupled with a local search mechanism that was proposed in [11]. ELGP was originally used for the solution of symbolic regression problems. Individuals in ELGP are stored using a linear representation, which is decoded using stacks. Programs coded in the population are essentially mathematical functions, and the user can specify their number of variables (known as *terminals set*), as well as the operators used to manipulate them (known as the *functions set*). This is the GP implementation that we adopted to automatically generate scalarizing functions. However, we evidently had to modify the fitness function originally provided in ELGP, since it was designed to perform symbolic regression.

<sup>1</sup> Source code for ELGP is available at: <https://github.com/lacava/ellen>.

Algorithm 1 shows the main procedure of our proposal, which follows the essential steps of a generic GP algorithm. After the population of  $n$  individuals has been initialized (lines 1-2), the genotype of each individual  $\mathbf{x}_i$  is decoded to obtain a scalarizing function  $sf_i$ , which is in turn used to calculate the fitness of  $\mathbf{x}_i$  (lines 3-6). Then, the main loop is executed until one of the following stopping criteria is met: either the best fitness found in  $\mathbf{P}$  is under some threshold or the maximum number of generations  $t_{max}$  has been reached. The steps in this loop include the generation of a new population  $\mathbf{P}'$  using recombination and mutation (line 8), the evaluation of the new individuals (lines 9-12), as well as updating the population  $\mathbf{P}$  (line 13). Finally, the last population is returned as the output of the algorithm.

The major modification made to ELGP was the way of evaluating the fitness of the individuals. In order to measure the quality of the new scalarizing functions generated by our GP-based approach, we use them to solve an MOP adopting an MOEA and then we employ the hypervolume indicator [15] to assess the quality of the  $PF$ s obtained. For this sake, we used the *Many-Objective Metaheuristic Based on the R2 Indicator-II* (MOMBI-II)<sup>2</sup>, which is a metaheuristic that was originally proposed in [7]. MOMBI-II was developed to solve many-objective problems using scalarizing functions and it was able to outperform state-of-the-art MOEAs such as NSGA-III and MOEA/D in both real-world problems and benchmark problems [6]. This is, indeed, the reason why we selected MOMBI-II as our baseline algorithm to validate the new scalarizing functions generated by our proposed approach.

By default, MOMBI-II uses the Achievement Scalarizing Function (ASF) which is defined as follows:

$$ASF(\mathbf{f}', \mathbf{w}) := \max_i \left( \frac{f'_i}{w_i} \right) \quad (2)$$

where  $\mathbf{f}' := \mathbf{F}(\mathbf{x}) - \mathbf{z}$  is the image of  $\mathbf{x}$  in objective space modified by some given reference point  $\mathbf{z} \in \mathbb{R}^m$  and  $\mathbf{w} \in \mathbb{R}^m$  is a weights vector.

Our modified version of the fitness evaluation is outlined in Algorithm 2. The main loop calls MOMBI-II to solve the MOP given using the scalarizing function  $sf$  to be evaluated (line 3). Then, the hypervolume of the  $PF$  obtained is computed and stored (lines 4-5). This is repeated  $n$  times, in order to obtain an average value of the hypervolumes generated using  $sf$ . Finally, fitness is computed as  $HV_{max}$  minus the average hypervolume (lines 7-8). This adjustment using  $HV_{max}$  is needed since ELGP minimizes fitness, while we aim to maximize hypervolume values.

MOMBI-II uses the  $R2$  indicator to guide its search process, which is a weakly Pareto-compliant indicator with a low computational cost [2]. However, in spite of this, our strategy is indeed very time-consuming since we use the hypervolume to guide the search process of our approach. In order to improve this, we adopted

<sup>2</sup> The source code of MOMBI-II is available at:

<https://www.cs.cinvestav.mx/~EVOCINV/software/MOMBI-II/MOMBI-II.html>.

---

**Algorithm 2:** Procedure MOEAFitness
 

---

```

Input : MOP,  $sf$ ;
Output:  $fitness$ ;
1  $fitness \leftarrow 0$ ;
2 for  $i \in \{1, 2, \dots, n\}$  do
3    $PF_i \leftarrow \text{MOMBI2}(\text{MOP}, sf)$ ;
4    $HV \leftarrow$  compute hypervolume value of  $PF_i$ ;
5    $fitness \leftarrow fitness + HV$ ;
6 end
7  $fitness \leftarrow HV_{max} - fitness/n$ ;
8 return  $fitness$ ;

```

---

the approach reported in [6] to compute the hypervolume, which is one of the most computationally efficient algorithms currently available.

It is also worth emphasizing that we aim to generate scalarizing functions that can be as general as possible, in the sense of being able to attain a reasonably good performance over a wide range of test problems, rather than generating highly specialized scalarizing functions that can provide an outstanding performance in a single test problem. Thus, we argue that the high computational cost of our proposed approach is, consequently, justified.

Both ELGP and MOMBI-II require several parameters to be executed. However, for the sake of simplicity, we don't include them in the pseudocodes of the algorithms here presented. Nonetheless, the final implementation<sup>3</sup> of our proposed strategy includes all of the configuration files we used.

Using our proposed strategy we were able to perform multiple experiments. In this paper we present the results obtained in one of them. We used a population size of 30 individuals and a maximum number of generations of 50. Functions were initialized completely at random, considering two decision variables ( $\mathbf{f}'$  and  $\mathbf{w}$ ) and basic arithmetic operators (addition, subtraction, multiplication, and division). In the MOEAFitness procedure, we incorporated DTLZ4 with two objectives as the MOP to be solved. MOMBI-II was set to use a population size of 100 with a maximum number of objective function evaluations of 15,000. The reference point used to calculate hypervolume values was (1, 1). Consequently,  $HV_{max}$  was set to 1. The running time, using the setup previously described, was nearly one week, using a personal computer with an Intel Core i5-5200U processor and 8 GB of RAM.

At the end of the execution, the algorithm reports the last population as well as each individual's fitness (shown in Table 1). At this point, we performed a second phase of the experiment, where we hand-picked the most promising scalarizing functions obtained to analyze their performance. To do so, we used each of the final 30 functions to solve 7 test problems (DTLZ1 through DTLZ7) with 2 and 3 objectives. Also, we raised the limit of objective function evaluations

---

<sup>3</sup> The source code of our approach is available at:

[http://www.computacion.cs.cinvestav.mx/~abernabe/scalarizing\\_functions](http://www.computacion.cs.cinvestav.mx/~abernabe/scalarizing_functions).

**Table 1.** Scalarizing functions stored in last population.

Individual	Decodified scalarizing function	Fitness
1	$sf_1(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i - (w_i - (f'_i - w_i))) + ((f'_i * w_i) + f'_i)/w_i)) + f'_i$	0.789438
2	$sf_2(\mathbf{f}', \mathbf{w}) := \max_i (((((f'_i/f'_i) * ((f'_i + f'_i)/w_i) - f'_i)) * f'_i) - w_i) + f'_i$	0.78944
3	$sf_3(\mathbf{f}', \mathbf{w}) := \max_i (((((f'_i/f'_i) * ((f'_i + f'_i)/w_i) - f'_i)) * f'_i) - w_i) + f'_i$	0.78944
4	$sf_4(\mathbf{f}', \mathbf{w}) := \max_i (((((f'_i/f'_i) * ((f'_i + f'_i)/w_i) - f'_i)) * f'_i) - w_i) + f'_i$	0.78944
5	$sf_5(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i * f'_i) - w_i) + f'_i) + ((w_i - w_i) + f'_i)/w_i$	0.789522
6	$sf_6(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i * ((f'_i/f'_i)/w_i)) * w_i)/w_i) + w_i$	0.78953
7	$sf_7(\mathbf{f}', \mathbf{w}) := \max_i (f'_i + ((f'_i + (w_i * ((f'_i + f'_i) - w_i) - f'_i))) + (f'_i/w_i))$	0.789554
8	$sf_8(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i/w_i) + w_i + f'_i + 3)) + (f'_i/(2w_i))$	0.789556
9	$sf_9(\mathbf{f}', \mathbf{w}) := \max_i ((f'_i - (w_i - (f'_i - w_i))) + ((f'_i * w_i) + f'_i)/w_i)$	0.789559
10	$sf_{10}(\mathbf{f}', \mathbf{w}) := \max_i (f'_i * (((w_i)/(2/f'_i)) * f'_i) + w_i) - w_i)/f'_i$	0.789568
11	$sf_{11}(\mathbf{f}', \mathbf{w}) := \max_i (((2f'_i)/w_i) * f'_i) + ((w_i * f'_i) - (f'_i + w_i)) + f'_i$	0.789568
12	$sf_{12}(\mathbf{f}', \mathbf{w}) := \max_i (((w_i + f'_i)/f'_i) * f'_i)/(2w_i)$	0.789577
13	$sf_{13}(\mathbf{f}', \mathbf{w}) := \max_i (f'_i/((f'_i - (3w_i * 2f'_i + f'_i))/(f'_i + (w_i * f'_i) - w_i)))$	0.789596
14	$sf_{14}(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i * f'_i) + (f'_i - w_i))/(f'_i + (w_i/(w_i + (f'_i/f'_i)))))$	0.789617
15	$sf_{15}(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i * f'_i) + (f'_i - w_i))/(f'_i + (w_i/(w_i + (f'_i/f'_i)))))$	0.789617
16	$sf_{16}(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i * w_i) * f'_i) - w_i) + w_i$	0.789637
17	$sf_{17}(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i * (-w_i)) * f'_i) - w_i) + w_i$	0.789647
18	$sf_{18}(\mathbf{f}', \mathbf{w}) := \max_i (((w_i - f'_i) - f'_i) + f'_i) - (f'_i * w_i)/((w_i * w_i) + f'_i)$	0.789789
19	$sf_{19}(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i * f'_i) + w_i) + ((f'_i - (f'_i - w_i))/w_i))$	0.789846
20	$sf_{20}(\mathbf{f}', \mathbf{w}) := \max_i ((f'_i/((w_i + w_i) - f'_i) + (f'_i * ((f'_i - f'_i) + f'_i)))) * f'_i$	0.790033
21	$sf_{21}(\mathbf{f}', \mathbf{w}) := \max_i ((w_i - (f'_i/w_i)) - f'_i)$	0.790152
22	$sf_{22}(\mathbf{f}', \mathbf{w}) := \max_i ((f'_i/w_i) - (((f'_i * (-w_i)) * f'_i) - w_i) * (w_i/f'_i))$	0.790771
23	$sf_{23}(\mathbf{f}', \mathbf{w}) := \max_i (w_i + (f'_i/((w_i * w_i) * f'_i) - w_i))$	0.792996
24	$sf_{24}(\mathbf{f}', \mathbf{w}) := \max_i ((f'_i/f'_i) - (((f'_i * ((f'_i - f'_i) - w_i)) * f'_i) - w_i))$	0.796269
25	$sf_{25}(\mathbf{f}', \mathbf{w}) := \max_i ((f'_i/(w_i + w_i)) + f'_i)$	0.797929
26	$sf_{26}(\mathbf{f}', \mathbf{w}) := \max_i (f'_i + (f'_i/(w_i + w_i)))$	0.797929
27	$sf_{27}(\mathbf{f}', \mathbf{w}) := \max_i ((f'_i/w_i) + f'_i)$	0.797983
28	$sf_{28}(\mathbf{f}', \mathbf{w}) := \max_i ((f'_i/(f'_i + (w_i - f'_i))) * f'_i)$	0.798007
29	$sf_{29}(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i + w_i) + f'_i) + f'_i) * f'_i)/(w_i * f'_i)$	0.798761
30	$sf_{30}(\mathbf{f}', \mathbf{w}) := \max_i (((f'_i * ((f'_i - f'_i) - w_i)) * f'_i) - w_i)$	0.804673

up to 100,000. Then, we used the average hypervolume values obtained by each function in these problems. The main motivation behind this was to identify the best functions in terms of their generalization capabilities. Since DTLZ4 was the MOP used in the search process, all of the new scalarizing functions found are able to solve it relatively well, which can be seen from how similar their fitness values are. However, we are interested in finding functions that are able to solve a variety of MOPs, and not just one. Therefore, using this preliminary validation, we were able to identify solutions coded in individuals 8 and 21 as the most promising functions. We denoted these two newly found functions as *Artificially Generated Scalarizing Functions* (called **AGSF1** and **AGSF2**). They are defined as:

$$\mathbf{AGSF1}(\mathbf{f}', \mathbf{w}) := \max_i \left( |f'_i + w_i + \frac{f'_i}{w_i} + \frac{f'_i}{2w_i} + 3| \right) \quad (3)$$

$$\mathbf{AGSF2}(\mathbf{f}', \mathbf{w}) := \max_i \left( |w_i - \frac{f'_i}{w_i} - f'_i| \right) \quad (4)$$



### 3 Experimental Results

To evaluate the performance of **AGSF1** and **AGSF2**, we used a total of 23 test problems, including the Deb-Thiele-Laumanns-Zitzler (DTLZ) test suite [4], the Walking-Fish-Group (WFG) test suite [8], and the IDTLZ [9] test suite. The latter consists of a modification of the DTLZ test problems in which the Pareto Fronts are inverted in objective space.

In order to assess the scalability of the two generated scalarizing functions, each of the aforementioned problems was solved with 2, 3, 4, 5, 6 and 10 objectives, setting a limit of 150,000 objective function evaluations. Since we aimed to measure the improvement generated exclusively by the scalarizing functions, we used the same algorithm (MOMBI-II) in the solution of all problems, as well as the same parameters, and we only varied the scalarizing function used.

In [7] a quick scalability test was performed comparing three scalarizing functions commonly used in the area: **ASF**, the Weighted Tchebycheff Scalarizing Function (**WT**) and Penalty-based Boundary Intersection (**PBI**). Our results showed that when using more than three objectives, **ASF** clearly outperformed **WT** and **PBI**. For this same reason, we compared **AGSF1** and **AGSF2** with respect to **ASF**, since scalability is an important desirable feature for a new scalarizing function.

We performed 30 independent runs, with each of the three scalarizing functions, on all the test problems mentioned. For assessing performance, we adopted the hypervolume and the  $s$ -energy [5] indicators. The hypervolume is used to assess convergence (larger values indicate a better performance), while  $s$ -energy is used to measure how uniformly distributed the solutions generated are (smaller values indicate a better performance). In both cases, the values obtained were normalized within the range [0,1] to allow an easier comparison of results.

Tables 2 to 7 show the mean hypervolume values (along with their corresponding standard deviations) obtained by **AGSF1** and **AGSF2** with respect to **ASF**. Tables 8 to 13 show the corresponding  $s$ -energy values. The best values obtained are represented using **boldface**. Values shown in grayscale indicate that the best value is significantly better according to the Wilcoxon rank-sum test with a significance level of 5%.

We say that a given scalarizing function outperforms another one when the mean value is better and the differences are statistically significant. From the results obtained using the hypervolume, **AGSF1** outperformed **ASF** in 36.23% of the problems, while **ASF** outperformed **AGSF1** in 29.72% of the problems. Regarding **AGSF2**, it outperformed **ASF** in 55.07% of the problems, while **ASF** only outperformed **AGSF2** in 7.25% of the problems. We can observe that **AGSF1** significantly improved performance in the DTLZ test problems with 2 objectives. However, with an increasing number of objectives, this improvement begins to decay. A similar behavior can be seen in the WFG test problems. But, as the number of objectives increases, both **AGSF1** and **ASF** exhibit a similar performance. Finally, **AGSF1** clearly performs better than **ASF** in most of the IDTLZ test problems (with the exception of the bi-objective instances). In contrast, **AGSF2** improves the results obtained by **ASF** in most than half of the

**Table 2.** Comparison of results in test problems with 2 objectives using the hypervolume.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.94021</b>	5.39E-02	0.27743	6.65E-02	DTLZ1	0.94021	5.39E-02	<b>0.95971</b>	2.64E-02
DTLZ2	0.5726	1.09E-01	<b>0.98881</b>	4.13E-02	DTLZ2	0.5726	1.09E-01	<b>0.74690</b>	2.33E-03
DTLZ3	0.64588	1.18E-01	<b>0.81145</b>	1.93E-01	DTLZ3	<b>0.64588</b>	1.18E-01	0.62899	1.58E-01
DTLZ4	0.96411	1.79E-01	<b>0.96656</b>	1.79E-01	DTLZ4	0.96411	1.79E-01	<b>0.99829</b>	2.33E-05
DTLZ5	0.5726	1.09E-01	<b>0.98881</b>	4.13E-02	DTLZ5	0.5726	1.09E-01	<b>0.74690</b>	2.33E-03
DTLZ6	0.53421	2.04E-01	<b>0.60970</b>	1.86E-01	DTLZ6	<b>0.53421</b>	2.04E-01	0.50781	1.85E-01
DTLZ7	<b>0.95753</b>	3.06E-02	0.21037	7.59E-02	DTLZ7	0.95753	3.06E-02	<b>0.97201</b>	2.30E-02
WFG1	<b>0.57355</b>	2.12E-01	0.16113	9.55E-02	WFG1	<b>0.57355</b>	2.12E-01	0.51501	2.04E-01
WFG2	<b>0.42430</b>	1.12E-01	0.11345	5.32E-02	WFG2	0.4243	1.12E-01	<b>0.43835</b>	6.26E-02
WFG3	<b>0.65929</b>	1.65E-01	0.32225	1.76E-01	WFG3	0.65929	1.65E-01	<b>0.68387</b>	1.61E-01
WFG4	0.31754	1.47E-01	<b>0.52158</b>	2.61E-01	WFG4	0.31754	1.47E-01	<b>0.56102</b>	1.62E-01
WFG5	0.15006	1.17E-01	<b>0.51551</b>	1.69E-01	WFG5	0.15006	1.17E-01	<b>0.49791</b>	1.47E-01
WFG6	<b>0.43905</b>	1.82E-01	0.39697	1.46E-01	WFG6	<b>0.43905</b>	1.82E-01	0.40587	2.15E-01
WFG7	0.27839	1.41E-01	<b>0.63681</b>	2.03E-01	WFG7	0.27839	1.41E-01	<b>0.59735</b>	1.26E-01
WFG8	<b>0.65718</b>	1.18E-01	0.26518	1.34E-01	WFG8	0.65718	1.18E-01	<b>0.67604</b>	1.50E-01
WFG9	<b>0.47164</b>	4.71E-01	0.45408	4.49E-01	WFG9	0.47164	4.71E-01	<b>0.51543</b>	4.79E-01
IDTLZ1	<b>0.93320</b>	2.49E-01	0.93085	2.49E-01	IDTLZ1	0.93325	2.49E-01	<b>0.93326</b>	2.49E-01
IDTLZ2	<b>0.89975</b>	3.00E-01	0.89422	2.98E-01	IDTLZ2	0.89975	3.00E-01	<b>0.89998</b>	3.00E-01
IDTLZ3	<b>0.95714</b>	1.59E-03	0.02772	7.92E-03	IDTLZ3	0.95714	1.59E-03	<b>0.99624</b>	1.32E-03
IDTLZ4	<b>0.93309</b>	2.49E-01	0.92736	2.48E-01	IDTLZ4	0.93309	2.49E-01	<b>0.93333</b>	2.49E-01
IDTLZ5	<b>0.89975</b>	3.00E-01	0.89422	2.98E-01	IDTLZ5	0.89975	3.00E-01	<b>0.89998</b>	3.00E-01
IDTLZ6	<b>0.96640</b>	1.79E-01	0.96046	1.78E-01	IDTLZ6	0.9664	1.79E-01	<b>0.96666</b>	1.79E-01
IDTLZ7	0.96648	1.79E-01	<b>0.96654</b>	1.77E-01	IDTLZ7	0.96648	1.79E-01	<b>0.97006</b>	1.55E-01

**Table 3.** Comparison of results in test problems with 3 objectives using the hypervolume.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.78115</b>	6.67E-02	0.20439	1.36E-01	DTLZ1	0.78115	6.67E-02	<b>0.79505</b>	6.61E-02
DTLZ2	0.36573	8.54E-02	<b>0.51208</b>	1.53E-01	DTLZ2	0.36573	8.54E-02	<b>0.52165</b>	1.57E-01
DTLZ3	0.45864	1.46E-01	<b>0.57649</b>	1.60E-01	DTLZ3	0.45864	1.46E-01	<b>0.57568</b>	1.69E-01
DTLZ4	<b>0.94960</b>	1.66E-01	0.92533	2.39E-01	DTLZ4	0.9496	1.66E-01	<b>0.98814</b>	3.01E-03
DTLZ5	0.10259	7.18E-02	<b>0.82681</b>	3.98E-02	DTLZ5	0.10259	7.18E-02	<b>0.57040</b>	1.24E-01
DTLZ6	0.49945	2.04E-01	<b>0.50294</b>	2.02E-01	DTLZ6	0.49945	2.04E-01	<b>0.51185</b>	2.48E-01
DTLZ7	<b>0.51209</b>	6.11E-02	0.16349	1.29E-01	DTLZ7	0.51209	6.11E-02	<b>0.79868</b>	8.35E-02
WFG1	<b>0.72011</b>	2.76E-01	0.61325	2.75E-01	WFG1	<b>0.72011</b>	2.76E-01	0.63706	2.96E-01
WFG2	0.66584	4.19E-01	<b>0.71352</b>	3.45E-01	WFG2	0.66584	4.19E-01	<b>0.67507</b>	4.22E-01
WFG3	<b>0.60044</b>	1.76E-01	0.3563	1.42E-01	WFG3	0.60044	1.76E-01	<b>0.65851</b>	2.11E-01
WFG4	0.31721	1.56E-01	<b>0.45204</b>	1.86E-01	WFG4	0.31721	1.56E-01	<b>0.46655</b>	1.62E-01
WFG5	<b>0.48179</b>	1.80E-01	0.46342	1.62E-01	WFG5	0.48179	1.80E-01	<b>0.59633</b>	1.84E-01
WFG6	<b>0.46938</b>	2.00E-01	0.46405	1.65E-01	WFG6	0.46938	2.00E-01	<b>0.47637</b>	1.91E-01
WFG7	0.37713	1.12E-01	<b>0.42341</b>	1.78E-01	WFG7	0.37713	1.12E-01	<b>0.54767</b>	1.36E-01
WFG8	0.50481	8.66E-02	<b>0.64052</b>	2.04E-01	WFG8	0.50481	8.66E-02	<b>0.63066</b>	1.08E-01
WFG9	0.40139	4.27E-01	<b>0.54778</b>	4.09E-01	WFG9	0.40139	4.27E-01	<b>0.58829</b>	4.23E-01
IDTLZ1	0.83644	2.24E-01	<b>0.93314</b>	2.49E-01	IDTLZ1	0.83644	2.24E-01	<b>0.90195</b>	2.41E-01
IDTLZ2	0.92708	2.48E-01	<b>0.93223</b>	2.49E-01	IDTLZ2	0.92708	2.48E-01	<b>0.93242</b>	2.49E-01
IDTLZ3	0.89394	2.98E-01	<b>0.89884</b>	3.00E-01	IDTLZ3	0.89394	2.98E-01	<b>0.89892</b>	3.00E-01
IDTLZ4	0.96078	1.78E-01	<b>0.96613</b>	1.79E-01	IDTLZ4	0.96078	1.78E-01	<b>0.96635</b>	1.79E-01
IDTLZ5	0.93026	2.48E-01	<b>0.93283</b>	2.49E-01	IDTLZ5	0.93026	2.48E-01	<b>0.93365</b>	2.47E-01
IDTLZ6	0.8629	3.38E-01	<b>0.86650</b>	3.40E-01	IDTLZ6	0.8629	3.38E-01	<b>0.86641</b>	3.40E-01
IDTLZ7	<b>0.89981</b>	3.00E-01	0.89971	3.00E-01	IDTLZ7	<b>0.89981</b>	3.00E-01	0.88463	2.95E-01

**Table 4.** Comparison of results in test problems with 4 objectives using the hypervolume.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.79463</b>	8.72E-02	0.42196	1.59E-01	DTLZ1	<b>0.79463</b>	8.72E-02	0.79315	8.68E-02
DTLZ2	0.40867	1.62E-01	<b>0.71332</b>	1.37E-01	DTLZ2	0.40867	1.62E-01	<b>0.66507</b>	1.28E-01
DTLZ3	<b>0.82567</b>	4.75E-02	0.44948	1.68E-01	DTLZ3	<b>0.82567</b>	4.75E-02	0.80508	6.61E-02
DTLZ4	0.27359	1.27E-01	<b>0.75469</b>	1.41E-01	DTLZ4	0.27359	1.27E-01	<b>0.52208</b>	1.16E-01
DTLZ5	0.49497	2.86E-02	<b>0.89203</b>	4.60E-02	DTLZ5	0.49497	2.86E-02	<b>0.75741</b>	7.17E-02
DTLZ6	<b>0.60878</b>	8.49E-02	0.25492	1.17E-01	DTLZ6	<b>0.60878</b>	8.49E-02	0.5973	9.86E-02
DTLZ7	<b>0.36283</b>	1.20E-01	0.18684	1.26E-01	DTLZ7	0.36283	1.20E-01	<b>0.75108</b>	1.04E-01
WFG1	0.41544	2.94E-01	<b>0.41699</b>	2.60E-01	WFG1	<b>0.41544</b>	2.94E-01	0.40384	2.41E-01
WFG2	<b>0.85445</b>	3.04E-01	0.69125	3.94E-01	WFG2	0.85445	3.04E-01	<b>0.92206</b>	2.25E-01
WFG3	<b>0.61315</b>	1.02E-01	0.44094	1.57E-01	WFG3	<b>0.61315</b>	1.02E-01	0.59961	1.15E-01
WFG4	<b>0.50368</b>	1.85E-01	0.50028	2.03E-01	WFG4	0.50368	1.85E-01	<b>0.59369</b>	1.83E-01
WFG5	0.50659	1.84E-01	<b>0.59780</b>	1.44E-01	WFG5	0.50659	1.84E-01	<b>0.67177</b>	1.57E-01
WFG6	0.49412	1.65E-01	<b>0.54457</b>	2.17E-01	WFG6	<b>0.49412</b>	1.65E-01	0.48852	2.21E-01
WFG7	<b>0.33741</b>	1.37E-01	0.33588	1.51E-01	WFG7	0.33741	1.37E-01	<b>0.44829</b>	1.63E-01
WFG8	0.37295	1.50E-01	<b>0.66888</b>	2.29E-01	WFG8	0.37295	1.50E-01	<b>0.75818</b>	1.69E-01
WFG9	0.11596	2.08E-01	<b>0.39254</b>	3.68E-01	WFG9	0.11596	2.08E-01	<b>0.16571</b>	2.26E-01
IDTLZ1	0.65308	1.22E-01	<b>0.70751</b>	1.58E-01	IDTLZ1	0.65308	1.22E-01	<b>0.68065</b>	1.33E-01
IDTLZ2	0.88085	1.64E-01	<b>0.94438</b>	1.76E-01	IDTLZ2	0.88085	1.64E-01	<b>0.96325</b>	1.09E-02
IDTLZ3	0.88272	1.64E-01	<b>0.91043</b>	2.43E-01	IDTLZ3	0.88272	1.64E-01	<b>0.89657</b>	2.40E-01
IDTLZ4	0.8524	2.28E-01	<b>0.91060</b>	2.43E-01	IDTLZ4	0.8524	2.28E-01	<b>0.90002</b>	2.41E-01
IDTLZ5	0.92851	1.72E-01	<b>0.98888</b>	4.59E-03	IDTLZ5	0.92851	1.72E-01	<b>0.95289</b>	1.77E-01
IDTLZ6	0.77558	3.47E-01	<b>0.82205</b>	3.68E-01	IDTLZ6	0.77558	3.47E-01	<b>0.90936</b>	2.43E-01
IDTLZ7	0.86633	3.40E-01	<b>0.86660</b>	3.40E-01	IDTLZ7	<b>0.86633</b>	3.40E-01	0.84093	3.30E-01

**Table 5.** Comparison of results in test problems with 5 objectives using the hypervolume.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.86779</b>	5.45E-02	0.46538	1.78E-01	DTLZ1	<b>0.86779</b>	5.45E-02	0.84491	8.39E-02
DTLZ2	0.44663	1.55E-01	<b>0.76285</b>	1.49E-01	DTLZ2	0.44663	1.55E-01	<b>0.76262</b>	1.08E-01
DTLZ3	<b>0.47940</b>	1.18E-01	0.39361	1.76E-01	DTLZ3	0.4794	1.18E-01	<b>0.54441</b>	1.52E-01
DTLZ4	0.90638	1.29E-02	<b>0.98160</b>	1.30E-02	DTLZ4	0.90638	1.29E-02	<b>0.90696</b>	1.69E-01
DTLZ5	<b>0.53888</b>	3.07E-01	0.43925	3.32E-01	DTLZ5	0.53888	3.07E-01	<b>0.78200</b>	2.39E-01
DTLZ6	<b>0.78701</b>	1.06E-01	0.5533	2.12E-01	DTLZ6	0.78701	1.06E-01	<b>0.81507</b>	9.60E-02
DTLZ7	0.158	7.64E-02	<b>0.40469</b>	1.14E-01	DTLZ7	0.158	7.64E-02	<b>0.85507</b>	7.38E-02
WFG1	<b>0.45271</b>	1.74E-01	0.282	1.72E-01	WFG1	0.45271	1.74E-01	<b>0.52692</b>	2.05E-01
WFG2	<b>0.94906</b>	1.63E-01	0.81943	3.11E-01	WFG2	0.94906	1.63E-01	<b>0.95750</b>	1.65E-01
WFG3	<b>0.46443</b>	2.08E-01	0.4261	1.82E-01	WFG3	0.46443	2.08E-01	<b>0.51382</b>	2.04E-01
WFG4	<b>0.73234</b>	1.19E-01	0.4263	1.61E-01	WFG4	<b>0.73234</b>	1.19E-01	0.67221	1.31E-01
WFG5	0.55916	2.16E-01	<b>0.60429</b>	1.93E-01	WFG5	0.55916	2.16E-01	<b>0.63777</b>	1.68E-01
WFG6	0.50864	2.21E-01	<b>0.51723</b>	1.59E-01	WFG6	0.50864	2.21E-01	<b>0.58072</b>	1.74E-01
WFG7	<b>0.39129</b>	1.57E-01	0.21911	1.46E-01	WFG7	0.39129	1.57E-01	<b>0.46373</b>	1.68E-01
WFG8	<b>0.48954</b>	1.50E-01	0.43107	1.23E-01	WFG8	0.48954	1.50E-01	<b>0.52675</b>	1.55E-01
WFG9	0.46108	1.49E-01	<b>0.84287</b>	1.14E-01	WFG9	0.46108	1.49E-01	<b>0.61813</b>	1.43E-01
IDTLZ1	<b>0.07390</b>	1.77E-01	0.03128	3.32E-02	IDTLZ1	<b>0.07390</b>	1.77E-01	0.04525	4.84E-02
IDTLZ2	0.58141	1.94E-01	<b>0.78339</b>	3.92E-01	IDTLZ2	0.58141	1.94E-01	<b>0.69048</b>	3.81E-01
IDTLZ3	0.56714	2.23E-01	<b>0.84305</b>	3.31E-01	IDTLZ3	0.56714	2.23E-01	<b>0.77850</b>	3.05E-01
IDTLZ4	0.57595	1.92E-01	<b>0.94902</b>	1.76E-01	IDTLZ4	0.57595	1.92E-01	<b>0.81559</b>	2.72E-01
IDTLZ5	<b>0.81135</b>	2.17E-01	0.75983	4.19E-01	IDTLZ5	0.81135	2.17E-01	<b>0.89972</b>	2.40E-01
IDTLZ6	0.63062	2.47E-01	<b>0.95735</b>	1.78E-01	IDTLZ6	0.63062	2.47E-01	<b>0.80709</b>	3.17E-01
IDTLZ7	0.89739	2.99E-01	<b>0.89966</b>	3.00E-01	IDTLZ7	<b>0.89739</b>	2.99E-01	0.8315	3.24E-01

**Table 6.** Comparison of results in test problems with 6 objectives using the hypervolume.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.77527</b>	1.01E-01	0.22056	1.14E-01	DTLZ1	<b>0.77527</b>	1.01E-01	0.73828	1.05E-01
DTLZ2	0.34124	1.47E-01	<b>0.75911</b>	1.23E-01	DTLZ2	0.34124	1.47E-01	<b>0.66103</b>	1.09E-01
DTLZ3	<b>0.69014</b>	1.01E-01	0.33642	1.68E-01	DTLZ3	0.69014	1.01E-01	<b>0.70567</b>	8.26E-02
DTLZ4	0.29561	1.09E-01	<b>0.89387</b>	7.15E-02	DTLZ4	0.29561	1.09E-01	<b>0.61672</b>	9.31E-02
DTLZ5	<b>0.38145</b>	2.03E-01	0.2731	2.08E-01	DTLZ5	<b>0.38145</b>	2.03E-01	<b>0.66591</b>	1.83E-01
DTLZ6	<b>0.70644</b>	1.10E-01	0.67136	2.26E-01	DTLZ6	0.70644	1.10E-01	<b>0.77577</b>	1.03E-01
DTLZ7	<b>0.63776</b>	9.70E-02	0.26638	1.10E-01	DTLZ7	<b>0.63776</b>	9.70E-02	0.53999	9.49E-02
WFG1	<b>0.52651</b>	2.12E-01	0.47232	2.51E-01	WFG1	0.52651	2.12E-01	<b>0.53469</b>	2.15E-01
WFG2	<b>0.92164</b>	2.32E-01	0.85487	2.80E-01	WFG2	0.92164	2.32E-01	<b>0.92320</b>	2.30E-01
WFG3	0.48992	2.51E-01	<b>0.49084</b>	2.08E-01	WFG3	<b>0.48992</b>	2.51E-01	0.4749	2.54E-01
WFG4	<b>0.67214</b>	1.75E-01	0.52001	2.11E-01	WFG4	0.67214	1.75E-01	<b>0.69414</b>	2.12E-01
WFG5	0.42035	2.09E-01	<b>0.50445</b>	2.26E-01	WFG5	0.42035	2.09E-01	<b>0.60843</b>	1.65E-01
WFG6	0.47871	2.56E-01	<b>0.51014</b>	2.64E-01	WFG6	0.47871	2.56E-01	<b>0.54992</b>	2.18E-01
WFG7	<b>0.77093</b>	1.47E-01	0.61142	2.41E-01	WFG7	0.77093	1.47E-01	<b>0.77884</b>	1.65E-01
WFG8	0.42047	1.98E-01	<b>0.65890</b>	1.78E-01	WFG8	0.42047	1.98E-01	<b>0.61073</b>	1.89E-01
WFG9	0.44598	1.43E-01	<b>0.70867</b>	1.70E-01	WFG9	0.44598	1.43E-01	<b>0.66831</b>	1.33E-01
IDTLZ1	0.92238	1.72E-01	<b>0.92276</b>	1.72E-01	IDTLZ1	0.92238	1.72E-01	<b>0.92356</b>	1.72E-01
IDTLZ2	0.65039	1.75E-01	<b>0.66423</b>	1.79E-01	IDTLZ2	0.65039	1.75E-01	<b>0.69323</b>	1.41E-01
IDTLZ3	0.78181	2.12E-01	<b>0.79616</b>	2.18E-01	IDTLZ3	0.78181	2.12E-01	<b>0.80746</b>	2.18E-01
IDTLZ4	<b>0.89270</b>	1.68E-01	0.82396	2.75E-01	IDTLZ4	<b>0.89270</b>	1.68E-01	0.8922	1.67E-01
IDTLZ5	0.90936	2.43E-01	<b>0.91405</b>	2.44E-01	IDTLZ5	0.90936	2.43E-01	<b>0.91376</b>	2.40E-01
IDTLZ6	0.91392	1.70E-01	<b>0.92086</b>	1.70E-01	IDTLZ6	0.91392	1.70E-01	<b>0.92140</b>	1.69E-01
IDTLZ7	0.82823	3.70E-01	<b>0.83264</b>	3.72E-01	IDTLZ7	<b>0.82823</b>	3.70E-01	0.79901	3.55E-01

**Table 7.** Comparison of results in test problems with 10 objectives using the hypervolume.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.81294</b>	1.10E-01	0.58344	1.34E-01	DTLZ1	<b>0.81294</b>	1.10E-01	0.76191	1.63E-01
DTLZ2	<b>0.65066</b>	1.41E-01	0.65001	1.53E-01	DTLZ2	0.65066	1.41E-01	<b>0.75830</b>	1.02E-01
DTLZ3	<b>0.66770</b>	8.45E-02	0.43772	1.57E-01	DTLZ3	0.66770	8.45E-02	<b>0.70229</b>	1.37E-01
DTLZ4	0.65936	7.43E-02	<b>0.84957</b>	1.40E-01	DTLZ4	0.65936	7.43E-02	<b>0.70648</b>	2.05E-01
DTLZ5	<b>0.30730</b>	2.18E-01	0.25256	2.74E-01	DTLZ5	0.3073	2.18E-01	<b>0.36810</b>	2.60E-01
DTLZ6	0.44166	2.43E-01	<b>0.57816</b>	2.36E-01	DTLZ6	<b>0.44166</b>	2.43E-01	0.43486	2.15E-01
DTLZ7	<b>0.56313</b>	1.12E-01	0.29824	1.36E-01	DTLZ7	<b>0.56313</b>	1.12E-01	0.46371	1.65E-01
WFG1	<b>0.54699</b>	1.98E-01	0.52031	2.20E-01	WFG1	<b>0.54699</b>	1.98E-01	0.5019	1.85E-01
WFG2	<b>0.93733</b>	1.69E-01	0.92103	1.73E-01	WFG2	0.93733	1.69E-01	<b>0.94825</b>	1.72E-01
WFG3	0.91771	1.72E-01	<b>0.95130</b>	2.43E-02	WFG3	0.91771	1.72E-01	<b>0.95036</b>	1.77E-02
WFG4	0.59836	1.91E-01	<b>0.61382</b>	2.03E-01	WFG4	0.59836	1.91E-01	<b>0.67055</b>	1.68E-01
WFG5	<b>0.59335</b>	1.85E-01	0.56807	1.96E-01	WFG5	<b>0.59335</b>	1.85E-01	0.54048	1.85E-01
WFG6	<b>0.59188</b>	1.79E-01	0.51932	1.74E-01	WFG6	0.59188	1.79E-01	<b>0.61110</b>	2.19E-01
WFG7	<b>0.56534</b>	1.29E-01	0.55873	1.62E-01	WFG7	0.56534	1.29E-01	<b>0.58656</b>	1.14E-01
WFG8	0.71276	1.43E-01	<b>0.73562</b>	1.81E-01	WFG8	0.71276	1.43E-01	<b>0.72063</b>	2.06E-01
WFG9	0.46988	1.73E-01	<b>0.62335</b>	1.81E-01	WFG9	0.46988	1.73E-01	<b>0.57258</b>	1.58E-01
IDTLZ1	<b>0.60202</b>	2.27E-01	0.57452	2.12E-01	IDTLZ1	<b>0.60202</b>	2.27E-01	0.54422	2.37E-01
IDTLZ2	0.63038	2.07E-01	<b>0.65898</b>	2.07E-01	IDTLZ2	<b>0.63038</b>	2.07E-01	0.61508	2.04E-01
IDTLZ3	<b>0.70399</b>	1.86E-01	0.69497	1.77E-01	IDTLZ3	0.70399	1.86E-01	<b>0.72820</b>	1.87E-01
IDTLZ4	0.52129	2.71E-01	<b>0.63378</b>	2.38E-01	IDTLZ4	0.52129	2.71E-01	0.52129	2.71E-01
IDTLZ5	<b>0.86257</b>	2.32E-01	0.86257	2.32E-01	IDTLZ5	<b>0.86257</b>	2.32E-01	0.84874	2.29E-01
IDTLZ6	<b>0.78459</b>	2.17E-01	0.77397	2.16E-01	IDTLZ6	<b>0.78459</b>	2.17E-01	0.7841	2.19E-01
IDTLZ7	<b>0.71568</b>	4.32E-01	0.61684	4.69E-01	IDTLZ7	<b>0.71568</b>	4.32E-01	0.71295	4.30E-01

**Table 8.** Comparison of results in test problems with 2 objectives using *s*-energy.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.03396</b>	5.34E-02	0.04773	6.30E-02	DTLZ1	0.03396	5.34E-02	<b>0.02294</b>	4.12E-02
DTLZ2	0.0425	1.80E-01	<b>0.00436</b>	4.57E-03	DTLZ2	0.0425	1.80E-01	<b>0.02872</b>	1.31E-01
DTLZ3	0.10072	2.22E-01	<b>0.09638</b>	1.77E-01	DTLZ3	0.10072	2.22E-01	<b>0.04910</b>	4.88E-02
DTLZ4	1.38E-05	5.83E-05	<b>1.19E-05</b>	5.78E-05	DTLZ4	1E-05	5.83E-05	<b>2.45E-06</b>	7.05E-06
DTLZ5	0.0425	1.80E-01	<b>0.00436</b>	4.57E-03	DTLZ5	0.0425	1.80E-01	<b>0.02872</b>	1.31E-01
DTLZ6	0.08795	1.86E-01	<b>0.07449</b>	1.55E-01	DTLZ6	0.08795	1.86E-01	<b>0.06997</b>	7.05E-02
DTLZ7	<b>0.00444</b>	6.50E-03	0.03115	3.50E-02	DTLZ7	<b>0.00444</b>	6.50E-03	0.02013	7.77E-02
WFG1	0.07402	1.16E-01	<b>0.05840</b>	5.61E-02	WFG1	0.07402	1.16E-01	<b>0.06070</b>	7.74E-02
WFG2	0.16328	1.88E-01	<b>0.11361</b>	1.66E-01	WFG2	0.16328	1.88E-01	<b>0.13370</b>	1.51E-01
WFG3	<b>0.05228</b>	5.60E-02	0.0613	5.80E-02	WFG3	<b>0.05228</b>	5.60E-02	0.08831	1.32E-01
WFG4	0.07741	1.53E-01	<b>0.02509</b>	4.12E-02	WFG4	<b>0.07741</b>	1.53E-01	0.10605	2.06E-01
WFG5	0.13371	1.13E-01	<b>0.10734</b>	1.12E-01	WFG5	<b>0.13371</b>	1.13E-01	0.18122	1.68E-01
WFG6	0.07397	1.51E-01	<b>0.01898</b>	1.80E-02	WFG6	0.07397	1.51E-01	<b>0.02855</b>	3.87E-02
WFG7	0.04333	6.10E-02	<b>0.01828</b>	1.86E-02	WFG7	0.04333	6.10E-02	<b>0.02975</b>	2.49E-02
WFG8	<b>0.00512</b>	1.48E-02	0.03646	1.79E-01	WFG8	0.00512	1.48E-02	<b>0.00295</b>	2.15E-03
WFG9	0.14668	1.50E-01	<b>0.09273</b>	5.06E-02	WFG9	<b>0.14668</b>	1.50E-01	0.15609	1.66E-01
IDTLZ1	0.0347	1.79E-01	<b>0.00231</b>	9.86E-03	IDTLZ1	0.0347	1.79E-01	<b>0.01802</b>	9.57E-02
IDTLZ2	<b>0.00010</b>	1.71E-04	0.05152	2.01E-01	IDTLZ2	<b>0.00010</b>	1.71E-04	0.01968	5.99E-02
IDTLZ3	<b>0.00004</b>	1.27E-04	5E-05	1.89E-04	IDTLZ3	<b>0.00004</b>	1.27E-04	0.0002	9.49E-04
IDTLZ4	<b>0.00016</b>	8.28E-05	0.00026	7.96E-05	IDTLZ4	<b>0.00016</b>	8.28E-05	0.03446	1.79E-01
IDTLZ5	<b>0.00010</b>	1.71E-04	0.05152	2.01E-01	IDTLZ5	<b>0.00010</b>	1.71E-04	0.01968	5.99E-02
IDTLZ6	<b>0.00001</b>	3.54E-05	0.03335	1.79E-01	IDTLZ6	<b>0.00001</b>	3.54E-05	0.00042	2.11E-03
IDTLZ7	<b>0.02281</b>	1.52E-02	0.09989	2.19E-01	IDTLZ7	<b>0.02281</b>	1.52E-02	0.03939	9.28E-02

**Table 9.** Comparison of results in test problems with 3 objectives using *s*-energy.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.02023</b>	1.09E-01	0.03334	1.79E-01	DTLZ1	0.02023	1.09E-01	<b>1.96E-06</b>	2.95E-06
DTLZ2	<b>0.01456</b>	2.15E-02	0.05342	1.82E-01	DTLZ2	0.01456	2.15E-02	<b>8.00E-03</b>	5.06E-03
DTLZ3	<b>6.50E-07</b>	2.04E-06	0.00174	9.37E-03	DTLZ3	6.50E-07	2.04E-06	<b>1.51E-07</b>	1.39E-07
DTLZ4	1.70E-02	9.12E-02	<b>0.00198</b>	7.25E-03	DTLZ4	0.01702	9.12E-02	<b>0.00047</b>	2.35E-03
DTLZ5	0.13761	1.82E-01	<b>0.12105</b>	1.74E-01	DTLZ5	<b>0.13761</b>	1.82E-01	0.24996	2.50E-01
DTLZ6	0.30194	2.65E-01	<b>0.10278</b>	1.58E-01	DTLZ6	0.30194	2.65E-01	<b>0.10464</b>	1.88E-01
DTLZ7	0.22586	3.13E-01	<b>0.21113</b>	2.96E-01	DTLZ7	<b>0.22586</b>	3.13E-01	0.26026	3.15E-01
WFG1	0.03947	1.55E-01	<b>0.00124</b>	3.93E-03	WFG1	0.03947	1.55E-01	<b>0.02587</b>	1.11E-01
WFG2	0.05319	2.01E-01	<b>0.00678</b>	1.50E-02	WFG2	0.05319	2.01E-01	<b>0.00968</b>	3.16E-02
WFG3	0.11941	2.48E-01	<b>0.04173</b>	1.33E-01	WFG3	0.11941	2.48E-01	<b>0.05415</b>	1.79E-01
WFG4	0.00771	4.31E-03	<b>0.00630</b>	9.99E-04	WFG4	0.00771	4.31E-03	<b>0.00670</b>	1.29E-03
WFG5	<b>2.06E-06</b>	3.44E-06	0.03333	1.79E-01	WFG5	2.06E-06	3.44E-06	<b>1.66E-06</b>	3.74E-06
WFG6	0.09337	1.69E-01	<b>0.05733</b>	1.39E-02	WFG6	0.09337	1.69E-01	<b>0.05543</b>	8.35E-03
WFG7	<b>0.00103</b>	2.49E-04	0.00137	1.81E-03	WFG7	<b>0.00103</b>	2.49E-04	0.0352	1.79E-01
WFG8	0.06543	2.05E-01	<b>0.04181</b>	1.80E-01	WFG8	0.06543	2.05E-01	<b>0.00137</b>	2.44E-03
WFG9	0.03334	1.79E-01	<b>2.04E-06</b>	5.73E-06	WFG9	0.03334	1.79E-01	<b>1.31E-06</b>	3.61E-06
IDTLZ1	0.23695	3.04E-01	<b>0.05408</b>	1.43E-01	IDTLZ1	0.23695	3.04E-01	<b>0.00233</b>	5.04E-03
IDTLZ2	0.04852	1.37E-01	<b>0.02185</b>	1.17E-01	IDTLZ2	0.04852	1.37E-01	<b>0.00023</b>	1.14E-03
IDTLZ3	0.1014	2.99E-01	<b>0.00520</b>	2.79E-02	IDTLZ3	0.1014	2.99E-01	<b>0.00004</b>	5.93E-05
IDTLZ4	0.07753	2.01E-01	<b>0.07705</b>	2.19E-01	IDTLZ4	<b>0.07753</b>	2.01E-01	0.011	5.71E-02
IDTLZ5	0.06258	1.97E-01	<b>0.00042</b>	2.24E-03	IDTLZ5	0.06258	1.97E-01	<b>0.02224</b>	5.09E-02
IDTLZ6	0.04757	1.60E-01	<b>0.03524</b>	1.51E-01	IDTLZ6	<b>0.04757</b>	1.60E-01	0.06766	2.14E-01
IDTLZ7	0.04302	1.80E-01	<b>0.00234</b>	2.76E-03	IDTLZ7	0.04302	1.80E-01	<b>0.01335</b>	5.57E-02

**Table 10.** Comparison of results in test problems with 4 objectives using *s*-energy.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.01848</b>	4.51E-02	0.08178	2.25E-01	DTLZ1	<b>0.01848</b>	4.51E-02	0.02254	8.34E-02
DTLZ2	<b>0.00002</b>	6.24E-05	0.00031	1.02E-03	DTLZ2	<b>0.00002</b>	6.24E-05	7E-05	2.39E-04
DTLZ3	0.00901	4.84E-02	<b>0.00155</b>	8.12E-03	DTLZ3	<b>0.00901</b>	4.84E-02	0.03406	1.45E-01
DTLZ4	<b>0.00289</b>	9.64E-03	0.00679	3.53E-02	DTLZ4	0.00289	9.64E-03	<b>0.00185</b>	5.63E-03
DTLZ5	0.30118	2.32E-01	<b>0.29328</b>	2.34E-01	DTLZ5	0.30118	2.32E-01	<b>0.24575</b>	1.73E-01
DTLZ6	0.26745	1.39E-01	<b>0.23423</b>	1.51E-01	DTLZ6	0.26745	1.39E-01	<b>0.25200</b>	1.46E-01
DTLZ7	<b>0.06111</b>	7.91E-02	0.10539	1.24E-01	DTLZ7	0.06111	7.91E-02	<b>0.06099</b>	7.26E-02
WFG1	0.15004	2.32E-01	<b>0.10807</b>	2.24E-01	WFG1	<b>0.15004</b>	2.32E-01	0.17604	2.58E-01
WFG2	<b>0.00208</b>	8.78E-03	0.07611	2.45E-01	WFG2	0.00208	8.78E-03	<b>0.00072</b>	2.74E-03
WFG3	0.35852	1.94E-01	<b>0.22826</b>	1.74E-01	WFG3	0.35852	1.94E-01	<b>0.35003</b>	2.04E-01
WFG4	1.72E-06	5.49E-06	<b>1.44E-07</b>	3.37E-07	WFG4	<b>1.72E-06</b>	5.49E-06	1E-05	2.52E-05
WFG5	<b>0.00145</b>	7.81E-03	0.03484	1.79E-01	WFG5	0.00145	7.81E-03	<b>1.52E-06</b>	4.53E-06
WFG6	0.03196	7.74E-02	<b>0.03058</b>	7.08E-02	WFG6	<b>0.03196</b>	7.74E-02	0.05567	1.94E-01
WFG7	<b>0.00061</b>	1.36E-03	0.0035	1.16E-02	WFG7	<b>0.00061</b>	1.36E-03	0.03797	1.30E-01
WFG8	0.09748	2.05E-01	<b>0.01868</b>	5.73E-02	WFG8	0.09748	2.05E-01	<b>0.02312</b>	6.83E-02
WFG9	<b>7.11E-06</b>	3.71E-05	0.03355	1.79E-01	WFG9	<b>7.11E-06</b>	3.71E-05	1.25E-05	6.07E-05
IDTLZ1	<b>0.20303</b>	1.62E-01	0.20585	1.16E-01	IDTLZ1	0.20303	1.62E-01	<b>0.19638</b>	1.26E-01
IDTLZ2	0.22802	2.40E-01	<b>0.20800</b>	1.67E-01	IDTLZ2	0.22802	2.40E-01	<b>0.16250</b>	1.57E-01
IDTLZ3	<b>0.20221</b>	2.00E-01	0.21904	1.66E-01	IDTLZ3	<b>0.20221</b>	2.00E-01	0.21936	2.07E-01
IDTLZ4	0.04939	6.27E-02	<b>0.04081</b>	4.84E-02	IDTLZ4	0.04939	6.27E-02	<b>0.04817</b>	4.72E-02
IDTLZ5	0.03609	1.79E-01	<b>0.00245</b>	2.16E-03	IDTLZ5	0.03609	1.79E-01	<b>0.00498</b>	1.38E-02
IDTLZ6	0.06322	2.10E-01	<b>0.05788</b>	1.99E-01	IDTLZ6	0.06322	2.10E-01	<b>0.01442</b>	6.83E-02
IDTLZ7	<b>0.00485</b>	1.22E-02	0.03573	1.79E-01	IDTLZ7	0.00485	1.22E-02	<b>0.00308</b>	7.18E-03

**Table 11.** Comparison of results in test problems with 5 objectives using *s*-energy.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.09211</b>	9.95E-02	0.11242	1.52E-01	DTLZ1	<b>0.09211</b>	9.95E-02	0.09653	1.32E-01
DTLZ2	<b>0.02271</b>	4.83E-02	0.03855	1.39E-01	DTLZ2	<b>0.02271</b>	4.83E-02	0.02979	6.78E-02
DTLZ3	0.04598	1.79E-01	<b>0.01271</b>	2.40E-02	DTLZ3	<b>0.04598</b>	1.79E-01	0.06622	1.51E-01
DTLZ4	<b>0.02071</b>	8.48E-02	0.1125	2.81E-01	DTLZ4	<b>0.02071</b>	8.48E-02	0.05334	1.16E-01
DTLZ5	<b>0.47548</b>	1.11E-01	0.58109	1.64E-01	DTLZ5	<b>0.47548</b>	1.11E-01	0.48196	1.40E-01
DTLZ6	<b>0.35530</b>	2.14E-01	0.46193	1.94E-01	DTLZ6	<b>0.35530</b>	2.14E-01	0.39761	1.90E-01
DTLZ7	<b>0.35466</b>	1.95E-01	0.36159	2.07E-01	DTLZ7	0.35466	1.95E-01	<b>0.22952</b>	1.45E-01
WFG1	0.23667	1.87E-01	<b>0.14168</b>	1.15E-01	WFG1	0.23667	1.87E-01	<b>0.21233</b>	2.60E-01
WFG2	<b>0.02767</b>	6.89E-02	0.18631	2.19E-01	WFG2	0.02767	6.89E-02	<b>0.01192</b>	2.51E-02
WFG3	0.45411	2.05E-01	<b>0.33911</b>	1.54E-01	WFG3	<b>0.45411</b>	2.05E-01	0.46662	1.79E-01
WFG4	0.03362	1.79E-01	<b>0.00001</b>	3.28E-05	WFG4	0.03362	1.79E-01	<b>0.00024</b>	8.47E-04
WFG5	<b>0.00493</b>	1.97E-02	0.0105	5.57E-02	WFG5	<b>0.00493</b>	1.97E-02	0.04316	1.85E-01
WFG6	<b>0.00718</b>	3.01E-02	0.01238	6.40E-02	WFG6	0.00718	3.01E-02	<b>0.00305</b>	9.02E-03
WFG7	<b>0.00969</b>	3.69E-02	0.01125	5.81E-02	WFG7	<b>0.00969</b>	3.69E-02	0.04716	1.86E-01
WFG8	<b>0.00264</b>	1.21E-02	0.00435	1.54E-02	WFG8	<b>0.00264</b>	1.21E-02	0.02136	5.34E-02
WFG9	<b>0.00004</b>	7.81E-05	0.00044	2.04E-03	WFG9	<b>0.00004</b>	7.81E-05	0.03835	1.79E-01
IDTLZ1	0.42105	2.27E-01	<b>0.36516</b>	1.35E-01	IDTLZ1	0.42105	2.27E-01	<b>0.37583</b>	1.87E-01
IDTLZ2	0.42158	1.76E-01	<b>0.39433</b>	1.65E-01	IDTLZ2	<b>0.42158</b>	1.76E-01	0.44151	1.86E-01
IDTLZ3	0.0042	7.40E-03	<b>0.00271</b>	2.01E-03	IDTLZ3	0.0042	7.40E-03	<b>0.00340</b>	2.17E-03
IDTLZ4	<b>0.39638</b>	1.50E-01	0.41305	1.42E-01	IDTLZ4	<b>0.39638</b>	1.50E-01	0.42975	2.17E-01
IDTLZ5	<b>0.00404</b>	1.32E-03	0.08645	2.50E-01	IDTLZ5	0.00404	1.32E-03	<b>0.00388</b>	1.43E-03
IDTLZ6	<b>0.00106</b>	4.92E-04	0.03432	1.79E-01	IDTLZ6	<b>0.00106</b>	4.92E-04	0.00319	1.23E-02
IDTLZ7	0.00634	3.55E-03	<b>0.00541</b>	8.90E-03	IDTLZ7	<b>0.00634</b>	3.55E-03	0.06161	1.90E-01

**Table 12.** Comparison of results in test problems with 6 objectives using  $s$ -energy.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	0.21791	1.32E-01	<b>0.18048</b>	8.55E-02	DTLZ1	0.21791	1.32E-01	<b>0.20438</b>	1.19E-01
DTLZ2	0.23022	2.21E-01	<b>0.20171</b>	2.32E-01	DTLZ2	0.23022	2.21E-01	<b>0.17225</b>	1.85E-01
DTLZ3	<b>0.14514</b>	1.76E-01	0.15298	2.21E-01	DTLZ3	0.14514	1.76E-01	<b>0.10612</b>	1.93E-01
DTLZ4	0.08846	1.72E-01	<b>0.06157</b>	9.75E-02	DTLZ4	0.08846	1.72E-01	<b>0.05743</b>	1.30E-01
DTLZ5	0.32194	1.56E-01	<b>0.25564</b>	1.86E-01	DTLZ5	0.32194	1.56E-01	<b>0.21800</b>	9.58E-02
DTLZ6	<b>0.39549</b>	1.86E-01	0.49549	2.07E-01	DTLZ6	<b>0.39549</b>	1.86E-01	0.41407	1.42E-01
DTLZ7	<b>0.24238</b>	1.54E-01	0.29417	1.34E-01	DTLZ7	<b>0.24238</b>	1.54E-01	0.2849	2.07E-01
WFG1	<b>0.23283</b>	1.60E-01	0.30558	2.35E-01	WFG1	<b>0.23283</b>	1.60E-01	0.24275	1.47E-01
WFG2	<b>0.06166</b>	1.11E-01	0.26726	2.55E-01	WFG2	0.06166	1.11E-01	<b>0.06107</b>	8.89E-02
WFG3	<b>0.40026</b>	1.38E-01	0.46579	2.02E-01	WFG3	<b>0.40026</b>	1.38E-01	0.4131	1.36E-01
WFG4	0.03638	1.25E-01	<b>0.00205</b>	5.01E-03	WFG4	0.03638	1.25E-01	<b>0.02977</b>	9.42E-02
WFG5	<b>0.00806</b>	2.09E-02	0.03415	1.20E-01	WFG5	<b>0.00806</b>	2.09E-02	0.06099	1.90E-01
WFG6	0.05204	1.89E-01	<b>0.00049</b>	1.05E-03	WFG6	0.05204	1.89E-01	<b>0.02432</b>	1.28E-01
WFG7	<b>0.06101</b>	1.95E-01	0.07192	1.94E-01	WFG7	<b>0.06101</b>	1.95E-01	0.06444	1.98E-01
WFG8	0.4212	1.56E-01	<b>0.30553</b>	1.54E-01	WFG8	0.4212	1.56E-01	<b>0.40457</b>	1.96E-01
WFG9	<b>0.05270</b>	1.19E-01	0.0866	2.44E-01	WFG9	<b>0.05270</b>	1.19E-01	0.10652	2.08E-01
IDTLZ1	<b>0.27139</b>	1.24E-01	0.30806	1.41E-01	IDTLZ1	<b>0.27139</b>	1.24E-01	0.31907	1.91E-01
IDTLZ2	0.28669	1.59E-01	<b>0.25054</b>	7.92E-02	IDTLZ2	0.28669	1.59E-01	<b>0.25578</b>	1.16E-01
IDTLZ3	<b>0.00005</b>	1.98E-04	9E-05	3.34E-04	IDTLZ3	<b>0.00005</b>	1.98E-04	6E-05	2.13E-04
IDTLZ4	0.5303	1.89E-01	<b>0.46159</b>	1.41E-01	IDTLZ4	0.5303	1.89E-01	<b>0.51955</b>	2.37E-01
IDTLZ5	<b>0.06674</b>	1.14E-01	0.08063	1.72E-01	IDTLZ5	0.06674	1.14E-01	<b>0.06600</b>	9.65E-02
IDTLZ6	0.41202	1.81E-01	<b>0.41163</b>	1.63E-01	IDTLZ6	<b>0.41202</b>	1.81E-01	0.42941	2.05E-01
IDTLZ7	0.01611	5.50E-02	<b>0.00762</b>	2.75E-02	IDTLZ7	<b>0.01611</b>	5.50E-02	0.18035	2.40E-01

**Table 13.** Comparison of results in test problems with 10 objectives using  $s$ -energy.

Problem	ASF		AGSF1		Problem	ASF		AGSF2	
	MED	STD	MED	STD		MED	STD	MED	STD
DTLZ1	<b>0.33773</b>	1.72E-01	0.53746	1.79E-01	DTLZ1	<b>0.33773</b>	1.72E-01	0.36237	1.56E-01
DTLZ2	0.44721	1.97E-01	<b>0.35538</b>	1.49E-01	DTLZ2	0.44721	1.97E-01	<b>0.37708</b>	1.83E-01
DTLZ3	0.39859	1.97E-01	<b>0.34639</b>	1.46E-01	DTLZ3	<b>0.39859</b>	1.97E-01	0.45988	2.24E-01
DTLZ4	<b>0.16349</b>	1.51E-01	0.19282	1.27E-01	DTLZ4	<b>0.16349</b>	1.51E-01	0.24254	2.50E-01
DTLZ5	<b>0.18951</b>	1.65E-01	0.22913	8.11E-02	DTLZ5	<b>0.18951</b>	1.65E-01	0.21848	1.01E-01
DTLZ6	<b>0.23592</b>	1.03E-01	0.2532	1.01E-01	DTLZ6	<b>0.23592</b>	1.03E-01	0.57038	2.38E-01
DTLZ7	<b>0.11122</b>	1.07E-01	0.18736	1.95E-01	DTLZ7	<b>0.11122</b>	1.07E-01	0.13158	1.28E-01
WFG1	<b>0.05130</b>	2.01E-02	0.05535	2.49E-02	WFG1	<b>0.05130</b>	2.01E-02	0.05754	3.70E-02
WFG2	<b>0.26651</b>	1.58E-01	0.39618	2.16E-01	WFG2	<b>0.26651</b>	1.58E-01	0.28387	1.53E-01
WFG3	<b>0.18477</b>	1.93E-01	0.20172	1.34E-01	WFG3	0.18477	1.93E-01	<b>0.12599</b>	5.67E-02
WFG4	0.10415	1.86E-01	<b>0.08143</b>	8.58E-02	WFG4	0.10415	1.86E-01	<b>0.08544</b>	9.81E-02
WFG5	0.10478	1.60E-01	<b>0.08112</b>	1.77E-01	WFG5	<b>0.10478</b>	1.60E-01	0.1169	2.17E-01
WFG6	0.21392	2.81E-01	<b>0.08966</b>	1.73E-01	WFG6	0.21392	2.81E-01	<b>0.15141</b>	2.29E-01
WFG7	<b>0.14863</b>	1.76E-01	0.14944	2.00E-01	WFG7	0.14863	1.76E-01	<b>0.12281</b>	2.07E-01
WFG8	0.31137	1.97E-01	<b>0.23653</b>	1.78E-01	WFG8	0.31137	1.97E-01	<b>0.21146</b>	2.22E-01
WFG9	0.21529	2.31E-01	<b>0.12855</b>	2.01E-01	WFG9	0.21529	2.31E-01	<b>0.14267</b>	1.42E-01
IDTLZ1	2.23E-07	1.20E-06	<b>4.29E-10</b>	2.30E-09	IDTLZ1	<b>2.23E-07</b>	1.20E-06	3.33E-07	1.29E-06
IDTLZ2	0.15901	1.52E-01	<b>0.11532</b>	1.32E-01	IDTLZ2	<b>0.15901</b>	1.52E-01	0.18295	2.08E-01
IDTLZ3	<b>2.53E-23</b>	1.36E-22	1.30E-16	6.98E-16	IDTLZ3	<b>2.53E-23</b>	1.36E-22	2.86E-17	1.54E-16
IDTLZ4	0.15096	2.51E-01	<b>0.13664</b>	2.10E-01	IDTLZ4	0.15096	2.51E-01	<b>0.14840</b>	2.43E-01
IDTLZ5	0.03375	1.79E-01	<b>0.01551</b>	5.64E-02	IDTLZ5	0.03375	1.79E-01	<b>0.01226</b>	6.28E-02
IDTLZ6	<b>0.00493</b>	2.60E-02	0.03346	1.79E-01	IDTLZ6	<b>0.00493</b>	2.60E-02	<b>0.00011</b>	2.04E-04
IDTLZ7	<b>0.00088</b>	4.21E-03	0.0517	2.01E-01	IDTLZ7	<b>0.00088</b>	4.21E-03	0.00364	1.36E-02

test problems. And in the worst cases, it performs similarly to **ASF**. Again, the improvement observed decreases as the number of objectives increases. Regarding the  $s$ -energy indicator, **AGSF1** outperformed **ASF** in 21.74% of the problems, while **ASF** outperformed **AGSF1** in 12.32% of the problems. Regarding **AGSF2**, it outperformed **ASF** in 18.84% of the problems, while **ASF** outperformed **AGSF2** in only 4.35% of the problems.

From this data, we can observe that, on average, **AGSF1** performs similarly or marginally better than **ASF**. However, **AGSF2** performs better than **ASF** in more than half of the cases when comparing hypervolume values and gets a similar performance when comparing  $s$ -energy values.

## 4 Conclusions and Future Work

We have proposed a strategy to generate new scalarizing functions using a combination of genetic programming and an MOEA. Using this strategy we were able to develop several scalarizing functions, from which we chose two (**AGSF1** and **AGSF2**) to perform an experimental evaluation of their performance. We used MOMBII-II to evaluate both of them against **ASF**, which is the scalarizing function used by default in MOMBII-II. Results obtained using a set of test problems and two performance indicators (namely hypervolume and  $s$ -energy) showed that **AGSF1** has a similar performance to that of **ASF**, while **AGSF2** outperforms **ASF** in more than half of the test problems adopted. It is interesting to note that the two new scalarizing functions were obtained trying to solve a specific MOP (DTLZ4 with 2 objectives), in which both of them outperformed **ASF**. However, these two new scalarizing functions were able to generalize their good performance in problems with a completely different Pareto Front geometry, or even with an increasing number of objectives.

There are several possible paths for future research. To the authors' best knowledge, this is the first proposal for the automatic generation of scalarizing functions, but there are obviously many other modifications that are worth exploring. For example, our proposed strategy can be run for a longer number of generations, aiming to produce better scalarizing functions. In the experiments reported in this paper, we adopted a maximum number of generations of 50 due to the high computational cost of our proposed approach, but if more computational power is available, a more thorough exploration of the search space could be conducted. Additionally, other operators can be considered in the functions set (e.g., trigonometric functions). Furthermore, the fitness evaluation procedure can be modified to either use another MOP (or even a combination of MOPs), or to use another performance indicator different from the hypervolume (or in addition to it). It is worth noting that the two scalarizing functions generated by our system share the same term ( $\frac{f'_i}{w_i}$ ) found in **ASF**. This suggests that this term could be a good starting seed for future executions of the algorithm. Finally, it would also be interesting to modify our proposed approach so that it can be used, for example, to generate performance indicators to be adopted in the selection mechanism of an (indicator-based) MOEA.






## References

1. Banzhaf, W., Francone, F.D., Keller, R.E., Nordin, P.: Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers, San Francisco (1998)
2. Brockhoff, D., Wagner, T., Trautmann, H.: On the properties of the  $R2$  indicator. In: 2012 Genetic and Evolutionary Computation Conference (GECCO2012), pp. 465–472. ACM Press (2012). ISBN: 978-1-4503-1177-9
3. Coello Coello, C.A.: A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl. Inf. Syst. **1**(3), 269–308 (1999)
4. 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)
5. Hardin, D.P., Saff, E.B.: Minimal Riesz energy point configurations for rectifiable  $d$ -dimensional manifolds. Adv. Math. **193**(1), 174–204 (2005)
6. Hernández Gómez, R.: Parallel Hyper-Heuristics for Multi-Objective Optimization. Ph.D. thesis, Department of Computer Science, CINVESTAV-IPN, Mexico City, México (2018)
7. 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). pp. 679–686. ACM Press (2015), ISBN 978-1-4503-3472-3
8. 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)
9. 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)
10. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Stat. Comput. **4**(2), 87–112 (1994)
11. La Cava, W., Helmuth, T., Spector, L., Danai, K.: Genetic programming with epigenetic local search. In: 2015 Genetic and Evolutionary Computation Conference (GECCO). pp. 1055–1062. ACM Press (2015), ISBN 978-1-4503-3472-3
12. Pescador-Rojas, M., Hernández Gómez, R., Montero, E., Rojas-Morales, N., Riff, M.-C., Coello Coello, C.A.: An overview of weighted and unconstrained scalarizing functions. In: Trautmann, H., et al. (eds.) EMO 2017. LNCS, vol. 10173, pp. 499–513. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-54157-0\\_34](https://doi.org/10.1007/978-3-319-54157-0_34)
13. Trivedi, A., Srinivasan, D., Sanyal, K., Ghosh, A.: A survey of multiobjective evolutionary algorithms based on decomposition. IEEE Trans. Evol. Comput. **21**(3), 440–462 (2017)
14. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm Evol. Comput. **1**(1), 32–49 (2011)
15. 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)



# The Usability Argument for Refinement Typed Genetic Programming

Alcides Fonseca<sup>(✉)</sup> , Paulo Santos , and Sara Silva 

LASIGE, Faculdade de Ciências da Universidade de Lisboa, Lisbon, Portugal  
{alcides,sara}@fc.ul.pt, psantos@lasige.di.fc.ul.pt

**Abstract.** The performance of Evolutionary Algorithms is frequently hindered by arbitrarily large search spaces. In order to overcome this challenge, domain-specific knowledge is often used to restrict the representation or evaluation of candidate solutions to the problem at hand. Due to the diversity of problems and the unpredictable performance impact, the encoding of domain-specific knowledge is a frequent problem in the implementation of evolutionary algorithms.

We propose the use of Refinement Typed Genetic Programming, an enhanced hybrid of Strongly Typed Genetic Programming (STGP) and Grammar-Guided Genetic Programming (GGGP) that features an advanced type system with polymorphism and dependent and refined types.

We argue that this approach is more usable for describing common problems in machine learning, optimisation and program synthesis, due to the familiarity of the language (when compared to GGGP) and the use of a unifying language to express the representation, the phenotype translation, the evaluation function and the context in which programs are executed.

**Keywords:** Genetic Programming · Refined types · Search-based software engineering

## 1 Introduction

Genetic Programming (GP) [28] has been successfully applied in different areas, including bioinformatics [13], quantum computing [35], and supervised machine learning [19]. One of the main challenges of applying GP to real-world problems, such as program synthesis, is the efficient exploration of the vast search space. Frequently, domain knowledge can be used to restrict the search space, making the exploration more efficient. Strongly Typed Genetic Programming (STGP) [24] restricts the search space by ignoring candidates that do not type check. To improve its expressive power, STGP has been extended with type

---

This work was supported by LASIGE (UIDB/00408/2020) and the CMU—Portugal project CAMELOT (POCI-01-0247-FEDER-045915).

inheritance [10], polymorphism [37] and a Hindley-Milner inspired type system [22], the basis for those in Haskell, SML, OCaml or F $\sharp$ .

Grammar-Guided Genetic Programming (GGGP) [21] also restricts the search space, sometimes enforcing the same rules as STGP, only allowing the generation of individuals that follow a given grammar. Grammar-based approaches have also been developing towards restricting the search space. The initial proposal [21] used context-free grammars (CFG) in the Backus Normal Form. The GAUGE [31] system relies on attribute grammars to restrict the phenotype translation from a sequence of integers. Christiansen grammars [6, 33] can express more restrictions than CFGs, but still have limitations, such as variable scoping, polymorphism or recursive declarations.

We propose Refinement Typed Genetic Programming (RTGP) as a more robust version of STGP through the use of a type system with refinements and dependent types. Languages with these features have gained focus in the Programming Languages (PL) research area: LiquidHaskell [36] is an extension of Haskell that supports refinements; Agda [2] and Idris [3] are dependently-typed languages that are frequently used as theorem provers. These languages support the encoding of specifications within the type system. Previously, special constructs were required to add specification verification within the source code. This idea was introduced in Eiffel [23] and applied later to Java with the JML specification language [18].

In particular, our major contributions are:

- A GP approach that relies on a simple grammar combined with a dependent refined type system, with the argument that this approach is more expressive than existing approaches;
- Concretisation of this approach in the  $\mathcal{A}EON$  Programming Language.

These contributions advance GP through a new interface in which to define representations and assess their success. One particular field where our approach might have a direct impact is general program synthesis. We identify two difficulties in the literature [15]: a) the large search space that results from the combination of language operators, grammar and available functions, and b) the lack of a continuous fitness function. We address both aspects within the same programming language.

In the remainder of the current paper we present: the  $\mathcal{A}EON$  language for expressing GP problems (Sect. 2); a method for extracting fitness functions from  $\mathcal{A}EON$  programs (Sect. 3); the Refined Typed Genetic Programming approach (Sect. 4); examples of RTGP (Sect. 5); a comparison with other approaches, from a usability point of view (Sect. 6); and concluding remarks (Sect. 7).

## 2 The $\mathcal{A}EON$ Programming Language

We introduce the  $\mathcal{A}EON$  programming language as an example of a language with polymorphism and non-liquid refinements. This language can be used as the basis for RTGP due to its support of static verification of polymorphism and

```

type Array<T> { size:Int } // size is a ghost variable

range : (mi:Int, ma:Int) → arr:Array<Int> where (ma > mi and arr.size == ma -
mi) = native;
append : (a:Array<T>, e:T) → n:Array<T> where (a.size + 1 == n.size) =
native;
listWith10Elements : (i:Int) → n:Array<Int> where (n.size == 10) {
  append(range(0,i), 42) // Type error
}
fib : (n:Int) → f:Int where (n >= 0 and f >= n) {
  if n < 2 then 1 else fib(n-1) + fib(n-2)
}
incomplete : (n:Int) → r:Int where (r > n && fib(r) % 100 == 0) {
  ■
}

```

**Listing 1.1.** An example of the  $\mathcal{A}EON$  language

a subset of the refinements. However, RTGP is not restricted to this language and could be applied to other languages that have similar type systems.

Listing 1.1 presents a simple example in  $\mathcal{A}EON$ . To keep  $\mathcal{A}EON$  a pure language, several low-level details are implemented in a host language, which  $\mathcal{A}EON$  can interact with using the **native** construct. The **range** function is an example of a function whose definition is done in the native language of the interpreter<sup>1</sup>.

What distinguishes  $\mathcal{A}EON$  from strongly typed mainstream languages like C or Java is that types can have refinements that express restrictions over the types. For instance, the refinements on **range** specify that the second argument must be greater than the first, and the output array has size equal to their different.

The **range** call in the **listWith10Elements** function throws a compile error because  $i$  is an Integer, and there are integers that are not greater than 0 (the first argument). The  $i$  argument should have been of type  $i:Int \text{ — } i \geq 0$ . However, there is another refinement being violated because if  $i = 1$ , the size of the output will be 2 and not 10 as expected. The correct input type should have been  $\{i:Int \text{ where } i \geq 9\}$  for the function to compile.

It should now be clear how a language like  $\mathcal{A}EON$  can be used to express domain-knowledge in GP problems. A traditional STGP solution would accept any integer value as the argument for **range** and would result in a runtime-error that would be penalized in the fitness function. Individual repair is not trivial to implement without resorting to symbolic execution, which is more computationally intensive than the static verification applied here.

The **incomplete** function, while very basic, is a simple example of the definition of a search problem. The function receives any integer (as there are no restrictions)

<sup>1</sup> We have developed a compiler from  $\mathcal{A}EON$  to Java and an  $\mathcal{A}EON$  interpreter in Python. In each case, the **range** function would have to be defined in Java and Python.

and returns an integer greater than the one received and whose Fibonacci number is divisible by 100. A placeholder hole (■) is left as the implementation of this function (inspired by Haskell’s and Agda’s `??name` holes [8]). The placeholder allows the program to parse, typecheck, but not execute<sup>2</sup>. Typechecking is required to describe the search problem: Acceptable solutions are those that inhabit the type of the hole, `{r:Int where r > n and fib(r)% 100 == 0}`. This is an example of a dependent refined type as the type of `r` depends on the value of `n` in the context. This approach of allowing the user to define a structure and let the search fill in the details has been used with success in sketch and SMT-based approaches [34].

While the `ÆON` language does not make any distinction, there are two classes of refinements for the purpose of RTGP: liquid and non-liquid refinements. Liquid refinements are those whose satisfiability can be statically verified, usually through the means of an SMT solver. One such example is `{x:Integer where x.size % 2 == 0}` SMT solvers can solve this kind of linear arithmetic problems. Another example is `{x:Array<Integer> where x.size > 0}` because `x.size` is the same as `size(x)` where `size` is an uninterpreted function in SMT solving.

Non-liquid refinements are those that SMT solvers are not able to reason about. These are typically not allowed in languages like LiquidHaskell [36]. One example is the second refinement of incomplete function, `fib(r)% 100 == 0` because the verification of correctness requires the execution of the `fib` function, which can only be called during runtime [7], typically for runtime verification. Another example of a non-liquid refinement would be the use of any natively defined function because the SMT cannot be sure of its behaviour other than the liquid refinement expressed in its type. For instance, when considering a native function that makes an HTTP request, an SMT solver cannot guess statically what kind of reply the server would send.

### 3 Refinements in GP

Now that we have addressed the difference between liquid and non-liquid refined types, we will see how both are used in the RTGP process. Figure 1 presents an overview of the data flow of the evolutionary process, starting from the problem formulation in `ÆON` and ending in the solution found, also in `ÆON`. The architecture identifies compiler components and the `ÆON` code that is either generated or manipulated by those components.

#### 3.1 Liquid Refinements for Constraining the Search Space

Liquid Refinements, the ones supported by Liquid Types [30], are conjunctions of statically verifiable logical predicates of data. We define all other refinements

---

<sup>2</sup> Replacing the hole by a crash-inducing expression allows the program to compile or be interpreted. While this is out of scope, the reader may find more in [25].

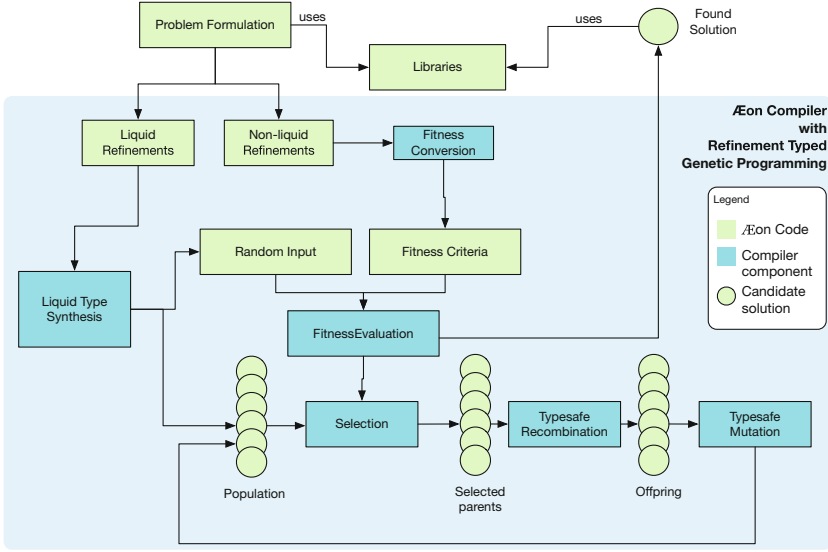


Fig. 1. Architecture of the proposed approach.

as non-liquid refinements. An example of a liquid type is  $\{x:\text{Int} \text{ where } x > 3 \text{ and } x < 7\}$ , where  $x > 3$  and  $x < 7$  are liquid refinements.

In our approach, liquid refinements are used to constraint the generation of candidate programs. Through the usage of a type-checker that supports Liquid Types, we are preventing candidates that are known to be invalid from being generated in the first place. However, the use of a type-checker is not ideal, as several invalid candidates might be generated before one that meets the liquid refinement is found. Synquid [29] is a first step in improving the performance of liquid type synthesis. Synquid uses an enumerative approach and an SMT-solver to synthesize programs from a Liquid Type. However, Synquid has several limitations for this purpose: it is unable to synthesize simple refinements related to numerical values (such as the previous example), it is deterministic since it uses an enumerative approach, and while it is presented as complete with regards to the semantic values (can generate all programs that can be expressed in the subset of supported Liquid Types), it is not complete with regards to the syntactic expression. As an example, Synquid is able to synthesize 2 as an integer, but not  $1+1$ , since the semantic values are equivalent. For the purposes of GP, not being able to synthesize more complex representations of the same program prevents recombination and mutation from exploring small alternatives. We are in the process of formalizing an algorithm that is more complete than Synquid for this purpose.

The RTGP algorithm (Sect. 4) uses the liquid type synthesis algorithm for:

- Generating random input arguments in fitness evaluation (Sect. 3.2);
- Generating a random individual in the initial population;

**Table 1.** Conversion function  $f$  between boolean expressions and continuous values.

Boolean	Continuous		<i>(continued)</i>
<b>true, false</b>	0.0, 1.0	$a \rightarrow b$	$f(\neg a \vee b)$
$x = y$	$\text{norm}( x - y )$	$\neg a$	$1 - f(a)$
$x \neq y$	$1 - f(x == y)$	$x \leq y$	$\text{norm}((x - y))$
$a \wedge b$	$(f(a) + f(b))/2$	$x < y$	$\text{norm}((x - y + \delta))$
$a \vee b$	$\min(f(a), f(b))$		

- Generating a subtree in the mutation operator;
- Generating a subtree in the recombination operator, when the other parent does not have a compatible node.

### 3.2 Non-liquid Refinements to Express Fitness Functions

A good fitness function is ideally continuous and should be able to measure how close to the real solution a potential solution is. However, fulfilling a given specification is a boolean criterion: it either is fulfilled or not. While previous work [15] has used the number of passed tests as the fitness function, we aim to have a more fine-grained measurement of how far each test is from passing. In particular, we consider the overall error as the fitness value, and the search as a minimization problem.

We propose the use of non-liquid refinement types to synthesize a continuous fitness criteria from the specification (depicted in Fig. 1) that, together with randomly generated input values, is used to obtain the fitness function.

```
f : (n:{Int | n > 0 }, a:{Array<String> | a.size == 3}) → r:Int where (r > n &&
  fib(r) % 100 == 0 and (n > 4 → serverCheck(r) == 3) ) { ■ }
```

**Listing 1.2.** An example of a specification that corresponds to a bi-objective problem

Listing 1.2 shows an example with a liquid refinement ( $r > n$ ) and two non-liquid clauses in the refinement. Each of these two clauses is handled individually as in a multi-objective problem. Each clause is first reduced to the conjunctive normal form (CNF) and then converted from a predicate into a continuous function that, given the input and expected output, returns a floating point number between 0.0 and 1.0, where the value represents the error. For instance, the example in Listing 1.2 is converted to two functions:

$$\text{norm}(\text{fib}(r)\% 100 - 0) \text{ and } \min(1 - \text{norm}(4 - n + \delta), \text{norm}(\text{serverCheck}(r) - 3))$$

Table 1 shows the conversion rules between boolean expressions and corresponding continuous values. The function  $f$ , which is defined using these rules, is applied recursively until a final continuous expression is generated. This approach is an extension of that presented in [12].

Since the output of  $f$  is an error, the value **true** is converted to 0.0, stating that condition holds, otherwise 1.0, this being the maximum value of not complying with the condition. Variables and function calls are also converted to 0.0 and 1.0 on whether the condition holds or not. Equalities of numeric values are converted into the normalized absolute difference between the arguments. The normalization is required as it allows different clauses to have the same importance on the given specification. Inequalities are converted to equalities and its difference with 1, negating the fitness result from equality. Conjunctions are converted to the average of the sum of the fitness extraction of both operands. Disjunctions value is obtained by extracting the minimum fitness value of both clauses. The minimum value indicates what clause is the closest to no error. Conditional statements fitness is recursively extracted by using the material implication rule. Similarly to inequalities, the negation of conditions denies the value returned by the truth of the condition. Numeric value comparisons represented a harder challenge as there are intervals where the condition holds. We use the difference of values to represent the error. In the  $<$  and  $>$  rules, the  $\delta$  constant depends on the type of the numerical value, 1.0 for integers and 0.00001 for doubles, and is essential for the extra step required for the condition to hold its truth value. A rectifier linear unit was used to ensure that if the condition holds, it is set to the maximum between the negative number and 0, otherwise, if the value is greater than 0, the positive fitness value is normalized.

The fitness function is the result of applying each  $f_i$  for each non-liquid refinement to a set of randomly generated (using the liquid synthesis algorithm in Sect. 3.1) input values. The fitness of an individual is the combination of all  $f_i$  for all random input values.

## 4 The RTGP Algorithm

The proposed RTGP algorithm follows the classical STGP [24] in its structure but differs in the details. Just like in all GP approaches, multiple variants can be obtained by changing or swapping some of the components presented here.

### 4.1 Representation

RTGP can have either a bitstream representation (e.g., [31]) or a direct representation (e.g., [24]). For the sake of simplicity, let us consider the direct representation in the remainder of the paper.

### 4.2 Initialization Procedure

To generate random individuals, the algorithm mentioned in Sect. 3.1 is used with the context and type of the  $\blacksquare$  as arguments. This is repeated until the population has the desired size. Koza proposed the combination of full and grow as ramped-half-and-half [14], which is used in classical STGP. In RTGP, the full method is not always possible, since no valid expression with that given depth



may exist in the language. If, for instance, we want an expression of type  $X$  and the only function that returns  $X$  is the constructor without any parameters. In this case, it is impossible to have any expression of type  $X$  with  $d$  greater than 1. Unlike in the STGP full method, a tree is used in the initial population, even if it does not have the predetermined depth.

### 4.3 Evaluation

The goal of the search problem is the minimization of the error between the observed and the expressed specification. Non-liquid refinements are translated to multi-objective criteria (following the approach explained in Sect. 3.2). The input values are randomly generated at each generation to prevent overfitting [9]. A fitness of 0.0 for one clause represents that all sets of inputs have passed that condition successfully. The overall objective of the candidate is to obtain a 0.0 fitness in all clauses.

### 4.4 Selection and Genetic Operators

Recent work has provided significant insights on parent selection in program synthesis [11]. A variant of lexicase selection, dynamic  $\epsilon$ -Lexicase [17] selection, has been used to allow near-elite individuals to be chosen in continuous search spaces.

The mutation operator chooses a random node from the candidate tree. A replacement is randomly generated by providing the node type to the expression synthesis algorithm along with the current node depth, fulfilling the maximum tree depth requirement. The valid subtrees of the replaced node are provided as genetic material to the synthesizer, allowing partial mutations on the candidate.

The crossover operator selects two random parents using the dynamic  $\epsilon$ -lexicase selection algorithm. A random node is chosen from the first parent, and nodes with the same type from the second parent are selected for transplantation into the first parent. If no compatible nodes are found, the expression synthesizer is invoked using the second parent valid subtrees, and the remaining first parent subtrees as genetic material. This is similar to how STGP operates, with the distinction that subtyping in Liquid Types refers to the implication of semantic properties. Thus, unsafe recombinations and mutations will never occur.

### 4.5 Stopping Criteria

The algorithm iterates over generations of the population until one or multiple of the following criteria are met: a) there is an individual of fitness 0.0; b) a predefined number of generations have been iterated; c) a predefined time duration has passed.

## 5 Examples of RTGP

This section introduces three examples from the literature implemented in  $\mathcal{A}EON$ .

### 5.1 Santa Fe Ant Trail

The Santa Fe Ant Trail problem is frequently used as a benchmark for GP. In [26], the authors propose a grammar-based approach to solve this problem. In RTGP, if-then-else conditions and auxiliary functions (via lambda abstraction) are embedded in the language, making this a very readable program.

```

type Map;
food_present : (m:Map) → Int = native;
food_ahead : (m:Map) → Boolean = native;
left : (m:Map) → Map = native;
right : (m:Map) → Map = native;
move : (m:Map) → Map = native;
program : (m:Map) → m2:Map where ( food_present(m2) == 0 ) { ■ }

```

Listing 1.3. Santa Fe Ant Trail

### 5.2 Super Mario Bros Level Design

The second example defines the search for an interesting design for a Super Mario Bros level that maximizes the engagement, minimizes frustration and maximizes challenge. These functions are defined according to a model that can easily be implemented in  $\mathcal{A}EON$  (Listing 1.4). We present this as a more usable alternative to the one that uses GGGP [32].

```

type X as {x:Integer | 5 <= x && x <= 95 }
type Y as {x:Integer | 3 <= x && x <= 5 }
type Wg as {x:Integer | 2 <= x && x <= 5 }
type W as {x:Integer | 2 <= x && x <= 7 }
type Wb as {x:Integer | 2 <= x && x <= 6 }
type Wa as Wb
type Wc as W
type Level as Pair<List<Chunk>, {enemies:List<Enemy> | 2 <= enemies.size &&
enemies.size <= 10}>;
type BoxType;
block_coin() → BoxType = native;
rock_coin() → BoxType = native;
block_powerup() → BoxType = native;
rock_empty() → BoxType = native;
type Chunk;
gap(x:X, y:Y, wg:Wg, wb:Wb, wa:Wa) → Level = native;
platform(x:X, y:Y, w:W) → Level = native;
hill(x:X, y:Y, w:W) → Level = native;

```

```

cannon_hill(x:X, y:Y, wg:Wg, wb:Wb, wa:Wa) → Level = native;
tube_hill(x:X, y:Y, wg:Wg, wb:Wb, wa:Wa) → Level = native;
coin(x:X, y:Y, w:Wc) → Level = native;
cannon(x:X, y:Y, wg:Wg, wb:Wb, wa:Wa) → Level = native;
tube(x:X, y:Y, wg:Wg, wb:Wb, wa:Wa) → Level = native;
boxes(t:BoxType, b:{List<Pair<X,Y>| 2 <= b.size && b.size <= 6 }) → Level =
native;
type Enemy;
koopaa(x:X) → Enemy = native;
goompa(x:X) → Enemy = native;

generateLevel() → l:Level where ( @maximize(engagement(l)) and
    @minimize(frustration(l)) and @maximize(challenge(l)) { ■ }

```

**Listing 1.4.** Super Mario Bros Level Design

Compared with the proposed grammar [32], the complexity is similar and productions in either version are directly correspondent. The  $\mathcal{A}EON$  version is arguably more expressive because the combinations of repetitions of objects with minimum and maximum number of repetitions can be bounded using types (enemies and boxes).

### 5.3 Logical Gates

The third example is taken from [27], where the goal is to “given any logical function, find a logically equivalent symbolic expression that uses only the operators in one of the three following complete sets: and, or, not, nand, nor”. The authors propose a Christiansen grammar, which is context-sensitive, to express this problem. Listing 1.5 presents a more simple implementation of the problem using  $\mathcal{A}EON$ . It can be argued that the implementation using refinements comes more directly from the problem statement than the complex dynamic grammar used in [27]. Furthermore, the implementation of the operations can be done directly in the same language.

```

set(x:Boolean) → y:Boolean = uninterpreted;
andG(x:Boolean, y:Boolean) → z:Boolean where ( set(x) == 1 and set(y) == 1
    and set(z) == 1 ) = { x && y }
or(x:Boolean, y:Boolean) → z:Boolean where ( set(x) == 1 and set(y) == 1 and
    set(z) == 1 ) = { x || y }
not(x:Boolean) → z:Boolean where ( set(x) == 1 and set(z) == 1 ) = { !x }
nand(x:Boolean, y:Boolean) → z:Boolean where ( set(x) == 2 and set(y) == 2
    and set(z) == 2 ) = { !(x && y) }
nor(x:Boolean, y:Boolean) → z:Boolean where ( set(x) == 3 and set(y) == 3 and
    set(z) == 3 ) = { !(x || y) }
target : (x:Boolean, ..., z:Boolean) → e:Boolean where ( e == f(x,...z) ) { ■ }

```

**Listing 1.5.** Equivalent Logical Gates to a given function f.

## 6 Discussion

This section compares RTGP with GGGP and presents arguments why RTGP could be used instead of GGGP. Because Dependent Types can encode grammars [5], the performance of both approaches is equivalent.

### 6.1 A Direct Comparison with GGGP

A survey on GGGP [21] identified the advantages and disadvantages of GGGP. We compare with RTGP on the advantages:

- **Ability to declaratively restrict the search space**—A type system is used instead of a grammar to express the restriction.
- **Problem Structure**—Problem domains that already follow a grammar structure can be easily encoded in RTGP. RTGP can more directly encode several problems than a grammar. Two examples are General-purpose programming and the Logical Gates problem (Sect. 5.3).
- **Homologous Operators**—Both GGGP and RTGP restrict the replacement of one component by another of similar close values.
- **Flexible Extension**—Extensions to GP can be encoded both in grammars and dependent types. Both approaches can be used as engines to test other GP concepts.

And disadvantages:

- **Feasibility Constraints**—Both GGGP and RTGP make the design of new operators a more significant challenge than in STGP, given that operators should follow the constraints imposed by the system. All RTGP operators are shared among any problem and rely solely on two algorithms: the type checker and expression synthesis.
- **Repair Mechanisms**—Implementing repairing in GGGP often depends on the grammar. RTGP relies on an expression synthesis algorithm (Sect. 3.1) that generates individuals in a way that constraints are never violated. The same has been done for the mutation and crossover operators. However, a repair mechanism is straightforward: the type-checker identifies the malign node, and expression synthesis generates a replacement.
- **Limited Flexibility**—GGGP is flexible when the program can be directly encoded in a context-free grammar. Some GGGP approaches use context-sensitive grammars (CSGP) [27], but readability can become a problem (explained in Sect. 6.2).
- **Turing Incompleteness**—GGGP supports grammars with semantics that allow the encoding of Turing-complete and incomplete problems. As such, GGGP does not offer any additional support for computation paradigms such as recursion and iteration, like other GP systems. RTGP supports both recursion and iteration directly, unless otherwise specified.

## 6.2 Usability

Instead, the main argument for RTGP over GGGP is one that concerns with usability. First, RTGP provides an integrated environment for describing the context, the problem, the search space and the solutions. Taking  $\mathcal{A}EON$  as an example of RTGP, the environment in which the final program will execute can be defined, relying on native functions to use software written in other programming languages. The problem is defined using refined types for the goal of the system, and a hole marker (■) is left as a placeholder for the program we are looking for. The search space is defined by the types used in the problem definition. Finally, the solution is a program in the same language as everything else, so it is ready to execute (and be evaluated).

On the other hand, if one were to use GGGP, one would have to create each of these components individually. The lack of a de-facto standard framework for GGGP helps this argument, in which interfacing with the context can be more complicated than implementing GGGP itself. GGGP concerns only with the description of the search space, while RTGP provides an integrated view of using GP.

The strongest point for RTGP is that it does not require the user to define a grammar. Just by placing holes in a program, users can use RTGP without even knowing how to define a grammar. Instead, they need to know how to use a familiar programming language (which to implement GGGP is already required) and to know how to express desired properties in refined types. While refined types have not yet become mainstream, several languages have feature subsets of its features for a long time. Eiffel [23] supported pre- and post-conditions since 1986. Ada is another language that supports design by contract [4], and it is very popular for critical embedded development, being used for large projects in air traffic control [16] with more than 1 million lines of code. Advanced type systems have become more popular to prevent bugs from existing in codebases. Mozilla created Rust to avoid concurrency issues in the Firefox browser [20], and Microsoft is using PL and SMT-based techniques to verify low-level critical components of the kernel and drivers [1].

## 7 Conclusions and Future Work

We have presented Refinement Typed Genetic Programming (RTGP) as an approach to describe search problems in an integrated programming language. We have introduced a language,  $\mathcal{A}EON$ , capable of expressing the environment, the fitness function, the search space, and the solution. The language features an advanced type system with liquid and non-liquid types. We have provided a methodology to generate the fitness function from non-liquid refined types, and we have introduced an algorithm that generates expressions from any inhabitable type in this language.

In Sect. 6 we have compared RTGP against GGGP, concluding that they are equivalent in expressiveness. However, we argue that RTGP provides better usability for end-users than GGGP, in which all aspects of the evolution have

to be implemented. Furthermore, expressing restrictions in types allows more modular programs and better readability inside an integrated experience for defining and using RTGP.

There are still some aspects to explore with regards to RTGP: identifying the most efficient representation; improving the liquid type synthesis; finding the best representation for non-functional properties of programs; how to integrate this synthesis in an integrated editor; and to perform an exhaustive benchmark performance analysis.

## References

1. Ball, T., Cook, B., Levin, V., Rajamani, S.K.: SLAM and static driver verifier: technology transfer of formal methods inside microsoft. In: Boiten, E.A., Derrick, J., Smith, G. (eds.) IFM 2004. LNCS, vol. 2999, pp. 1–20. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24756-2\\_1](https://doi.org/10.1007/978-3-540-24756-2_1)
2. Bove, A., Dybjer, P., Norell, U.: A brief overview of agda – a functional language with dependent types. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 73–78. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03359-9\\_6](https://doi.org/10.1007/978-3-642-03359-9_6)
3. Brady, E.: Idris, a general-purpose dependently typed programming language: design and implementation. *J. Funct. Program.* **23**(05), 552–593 (2013)
4. Brink, K., van Katwijk, J., Toetenel, W.: Ada 95 as implementation vehicle for formal specifications. In: Proceedings of 3rd International Workshop on Real-Time Computing Systems and Applications, pp. 98–105. IEEE (1996)
5. Brink, K., Holdermans, S., Löh, A.: Dependently typed grammars. In: Bolduc, C., Desharnais, J., Ktari, B. (eds.) MPC 2010. LNCS, vol. 6120, pp. 58–79. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-13321-3\\_6](https://doi.org/10.1007/978-3-642-13321-3_6)
6. Christiansen, H.: A survey of adaptable grammars. *ACM SIGPLAN Notices* **25**(11), 35–44 (1990)
7. Elmas, T., Tasiran, S., Qadeer, S.: Vyrd: verifying concurrent programs by runtime refinement-violation detection. *ACM SIGPLAN Notices* **40**(6), 27–37 (2005)
8. Gissurarson, M.P.: Suggesting Valid Hole Fits for Typed-Holes in Haskell. Master’s thesis (2018)
9. Gonçalves, I., Silva, S., Melo, J.B., Carreiras, J.M.B.: Random Sampling Technique for Overfitting Control in Genetic Programming. In: Moraglio, A., Silva, S., Krawiec, K., Machado, P., Cotta, C. (eds.) EuroGP 2012. LNCS, vol. 7244, pp. 218–229. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29139-5\\_19](https://doi.org/10.1007/978-3-642-29139-5_19)
10. Haynes, T.D., Schoenfeld, D.A., Wainwright, R.L.: Type inheritance in strongly typed genetic programming. *Adv. Genetic Program.* **2**(2), 359–376 (1996)
11. Helmuth, T., McPhee, N., Spector, L.: Lexicase selection for program synthesis: a diversity analysis, pp. 151–167 (2016)
12. Korel, B.: Automated software test data generation. *IEEE Trans. Software Eng.* **16**(8), 870–879 (1990)
13. Kosakovsky Pond, S.L., Posada, D., Gravenor, M.B., Woelk, C.H., Frost, S.D.: Gard: a genetic algorithm for recombination detection. *Bioinformatics* **22**(24), 3096–3098 (2006)
14. Koza, J.R.: Genetic Programming: On The Programming of Computers by Means Of Natural Selection, vol. 1. MIT press (1992)

15. Krawiec, K.: Program synthesis. Behavioral Program Synthesis with Genetic Programming. SCI, vol. 618, pp. 1–19. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-27565-9\\_1](https://doi.org/10.1007/978-3-319-27565-9_1)
16. Kruchten, P., Thompson, C.J.: An object-oriented, distributed architecture for large-scale ADA systems. In: Proceedings of the Conference on TRI-ADA 1994, pp. 262–271 (1994)
17. La Cava, W., Helmuth, T., Spector, L., Moore, J.H.: A probabilistic and multi-objective analysis of lexicase selection and e-lexicase selection. *Evol. Comput.* **27**(3), 377–402 (2019). <https://doi.org/10.1162/evco.a.00224>
18. Leavens, G.T., Baker, A.L., Ruby, C.: JML: a Java modeling language. In: Formal Underpinnings of Java Workshop (at OOPSLA), pp. 404–420 (1998)
19. Loveard, T., Ciesielski, V.: Representing classification problems in genetic programming. In: *Evolutionary Computation*, vol. 2, pp. 1070–1077. IEEE (2001)
20. Matsakis, N.D., Klock, F.S.: The rust language. *ACM SIGAda Ada Letters* **34**(3), 103–104 (2014)
21. McKay, R.I., Hoai, N.X., et al.: Grammar-based genetic programming: a survey. *Genetic Program. Evol. Mach.* **11**(3), 365–396 (2010)
22. McPhee, N.F., Hopper, N.J., Reiersen, M.L.: Impact of types on essentially typeless problems in GP. In: *Genetic Programming*, pp. 232–240 (1998)
23. Meyer, B.: Eiffel: programming for reusability and extendibility. *ACM Sigplan Notices* **22**(2), 85–94 (1987)
24. Montana, D.J.: Strongly typed genetic programming. *Evol. Comput.* **3**(2), 199–230 (1995)
25. Omar, C., Voysey, I., Chugh, R., Hammer, M.A.: Live functional programming with typed holes. *Proc. ACM Program. Lang.* **3**(POPL), 1–32 (2019)
26. O’Neill, M., Ryan, C.: Grammar based function definition in grammatical evolution. In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, pp. 485–490 (2000)
27. Ortega, A., De La Cruz, M., Alfonseca, M.: Christiansen grammar evolution: grammatical evolution with semantics. *IEEE Trans. Evol. Comput.* **11**(1), 77–90 (2007)
28. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk> (2008), (With contributions by J. R. Koza)
29. Polikarpova, N., Kuraj, I., Solar-Lezama, A.: Program synthesis from polymorphic refinement types. In: *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 522–538. ACM (2016)
30. Rondon, P.M., Kawaguci, M., Jhala, R.: Liquid types. In: *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 159–169 (2008)
31. Ryan, C., Nicolau, M., O’Neill, M.: Genetic Algorithms Using Grammatical Evolution. In: Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A. (eds.) *EuroGP 2002*. LNCS, vol. 2278, pp. 278–287. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45984-7\\_27](https://doi.org/10.1007/3-540-45984-7_27)
32. Shaker, N., Nicolau, M., Yannakakis, G.N., Togelius, J., O’neill, M.: Evolving levels for super mario bros using grammatical evolution. In: *2012 IEEE Conference on Computational Intelligence and Games (CIG)*. pp. 304–311. IEEE (2012)
33. Shutt, J.N.: *Recursive adaptable grammars* (1999)
34. Solar-Lezama, A.: *Program synthesis by sketching*. University of California, Berkeley (2008)

35. Spector, L., Barnum, H., Bernstein, H.J., Swamy, N.: Finding a better-than-classical quantum and/or algorithm using genetic programming. In: *Evolutionary Computation*. vol. 3, pp. 2239–2246. IEEE (1999)
36. Vazou, N., Seidel, E.L., Jhala, R., Vytiniotis, D., Peyton-Jones, S.: Refinement types for haskell. In: *ACM SIGPLAN Notices*. vol. 49, pp. 269–282. ACM (2014)
37. Yu, T.: Polymorphism and Genetic Programming. In: Miller, J., Tomassini, M., Lanzi, P.L., Ryan, C., Tettamanzi, A.G.B., Langdon, W.B. (eds.) *EuroGP 2001*. LNCS, vol. 2038, pp. 218–233. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-45355-5\\_17](https://doi.org/10.1007/3-540-45355-5_17)





# Program Synthesis in a Continuous Space Using Grammars and Variational Autoencoders

David Lynch<sup>1(✉)</sup>, James McDermott<sup>2(✉)</sup>, and Michael O'Neill<sup>1(✉)</sup>

<sup>1</sup> Natural Computing Research and Applications Group, UCD, Dublin, Ireland  
{david.lynch,m.oneill}@ucd.ie

<sup>2</sup> School of Computer Science, National University of Ireland, Galway, Ireland  
james.mcdermott@nuigalway.ie

**Abstract.** An important but elusive goal of computer scientists is the automatic creation of computer programs given only input and output examples. We present a novel approach to program synthesis based on the combination of grammars, generative neural models, and evolutionary algorithms. Programs are described by sequences of productions sampled from a Backus-Naur form grammar. A sequence-to-sequence Variational Autoencoder (VAE) is trained to embed randomly sampled programs in a continuous space – the VAE’s encoder maps a sequence of productions (a program) to a point  $z$  in the latent space, and the VAE’s decoder reconstructs the program given  $z$ . After the VAE has converged, we can engage the decoder as a generative model that maps locations in the latent space to executable programs. Hence, an Evolutionary Algorithm can be employed to search for a vector  $z$  (and its corresponding program) that solves the synthesis task. Experiments on the program synthesis benchmark suite suggest that the proposed approach is competitive with tree-based GP and PushGP. Crucially, code can be synthesised in any programming language.

## 1 Introduction

The automatic generation of computer programs has been a goal of researchers in the field of computer science since the origins of the discipline [34]. There are reports of primitive program synthesis in the literature dating back to the 1950’s [11,12] with many examples since [13,30,36,37]. The arrival of Genetic Programming and its variants in the late 1980’s renewed hopes that programs could be automatically generated by computers. In recent years, researchers in the wider machine learning community have also started to focus on program synthesis [1,14,16,22]. This interest is driven by the expectation that real-world applications of automated program synthesis will have enormous economic and social impact, and will also have important implications for artificial general intelligence [4].

The ability to automatically synthesise programs that solve challenging real-world problems remains an elusive goal. Reasons include the discrete and variable-length nature of computer programs, the non-local mapping between

syntax and semantics, the “all or nothing” aspect of program correctness, and the vast search space.

Genetic Programming (GP) [2, 25, 35], and its grammar-based variants [29, 33], is a form of evolutionary computation [5, 21] which can be used for program synthesis. GP routinely achieves human-competitive performance [24] in diverse domains including symbolic regression, architecture, and network optimisation. However, GP has yet to realise its full potential as an engine for automatic programming.

Recently, GP researchers have been calling for an increased focus on program synthesis that embraces techniques drawn from the wider fields of analytics and machine learning [32, 34]. One promising research direction looks to combine grammars with autoencoders [20, 23]. For instance, Kusner et al. [28] used a combination of grammars and a Variational Autoencoder (VAE) [23] to learn representations for two domains: symbolic regression and drug discovery. The VAE discovers a latent-space encoding of the neighbourhood of sentences in the language expressed by the grammar. The learned representation has appealing properties: it is continuous, approximately normally-distributed, and relatively syntactically and semantically smooth. Thus, powerful numerical optimisation algorithms can be employed to search for new arithmetic expressions or drug molecules.

In this study, we adopt a VAE with a sequence-to-sequence structure, in conjunction with grammars that represent a subset of the Python programming language. The VAE discovers a latent-space encoding of Python programs. An Evolutionary Algorithm [5] is used to successfully search this representation for novel programs. We examine a subset of problems with a range of difficulty drawn from the program synthesis benchmark suite [18].

The remainder of this paper is organised as follows. The grammars, VAE, and evolutionary algorithms are developed in Sect. 2. Our experimental set-up is described in Sect. 3. The proposed approach is benchmarked against canonical grammar-based GP and PushGP [39] in Sect. 4. Finally, we draw conclusions and outline how the algorithms could be improved in Sect. 5.

## 2 Methods

The main components of our approach are described in this section. Grammars that enable the creation of Python code to solve arbitrary program synthesis tasks are outlined. Our goal is to learn a latent representation of programs using a Variational Autoencoder (VAE). The VAE architecture is presented, and an Evolutionary Algorithm (EA) is developed to search its latent space.

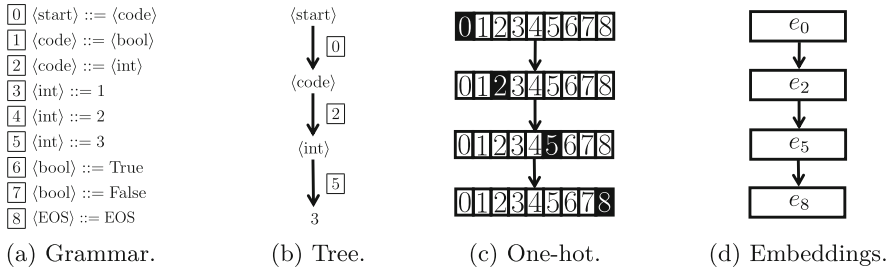
### 2.1 Grammar Design Pattern

Grammars have commonly been used to represent program spaces in GP [29, 33]. However, bespoke grammars had to be written for different program synthesis tasks. Forstenlechner et al. introduced a grammar design pattern to address this inflexibility [9, 10]. Their idea is to create a separate grammar for the Boolean,

float, integer, and string data types (see [8]). These sub-grammars are combined depending on what data types are required to solve a problem. An additional grammar constrains the control flow, such as the arrangement of conditionals and loops. Grammars guarantee type safety and they ensure that all individuals are syntactically correct. Runtime exceptions are further reduced via protected methods. Crucially, code can be created in any programming language including Python, Java, C, etc. We synthesise Python code in this paper.

### 2.2 Variational Autoencoder

VAEs are generative neural models consisting of an encoder and a decoder. In this section, we present an encoder that embeds discrete programs as vectors  $z$  in a continuous latent space. The decoder maps points in this latent space back to programs. Our goal is to search for fit programs by performing numerical optimisation in the latent space.

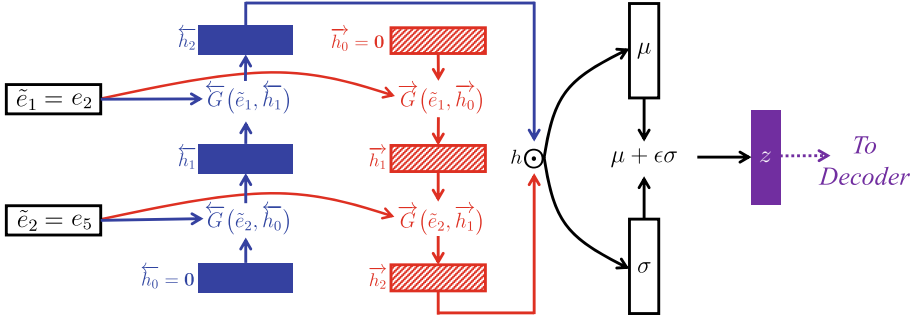


**Fig. 1.** A toy grammar is displayed in Plot (a). The derivation tree in Plot (b) is realised by expanding production rules 0, 2, and 5. In Plot (c), the program is given by a sequence of one-hot vectors (including an ‘EOS’ token). Finally, each one-hot vector is associated with a learned embedding vector in Plot (d).

Initially, a grammar for the program synthesis task is formed as outlined in Sect. 2.1. The VAE is trained on a corpus of programs sampled from this grammar. For example, consider the toy grammar and the sampled program displayed in Fig. 1. The production rules used to generate the program are represented by a sequence of one-hot vectors  $\tilde{o} = (o_0, o_2, o_5, o_8)$ . The associated embeddings  $\tilde{e} = (e_0, e_2, e_5, e_8)$  are then passed as inputs to the encoder.

We adopt a bidirectional [38] gated recurrent unit network [3] (BiGRU) as the encoder. A recurrent model lends itself naturally to modelling sequences of production rules. Furthermore, the BiGRU can deal with input sequences of an arbitrary length (that is, programs of different sizes). The flow of information through the encoder is illustrated in Fig. 2. A sequence of embeddings<sup>1</sup>  $\tilde{e}$  is provided as an input to the BiGRU. The function  $\vec{G}(\cdot)$  is a GRU cell [3] that

<sup>1</sup> Only  $(e_2, e_5)$  are displayed for clarity, but in practice  $(e_0, e_2, e_5, e_8)$  would be used.

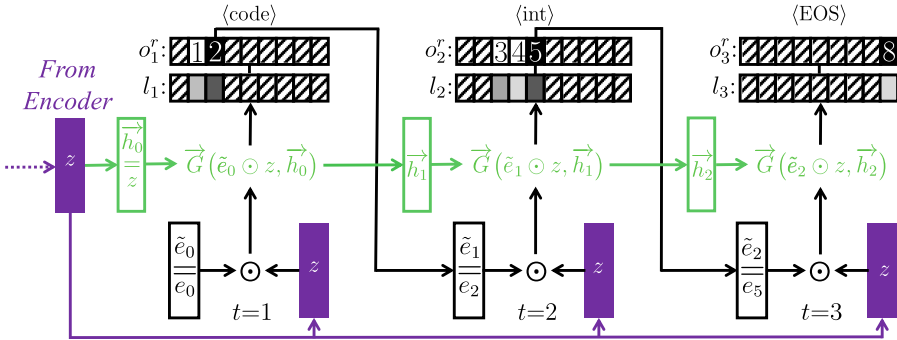


**Fig. 2.** The encoder maps embeddings  $\tilde{e}$  of the production rules used to generate a program to a latent representation  $z$ .

integrates the current embedding vector  $\tilde{e}_t$  with the previous forward hidden state  $\vec{h}_{t-1}$  to give  $\vec{h}_t$ . Similarly,  $\overleftarrow{G}(\cdot)$  updates the previous backward hidden state. The hidden states emerging from the BiGRU are concatenated to yield a summary of the program  $h$ . Hence, the latent code  $z$  is given by:

$$z = (W_{h\mu}h + b_\mu) + \epsilon(W_{h\sigma}h + b_\sigma) = \mu + \epsilon\sigma, \quad (1)$$

where the weight matrices  $W$  and bias vectors  $b$  are learned parameters of the model. Variables  $\mu$  and  $\sigma$  are interpreted as the mean and standard deviation of a multivariate normal distribution  $\mathcal{N}(\mu, \sigma)$ , and  $\epsilon$  is sampled from the multivariate standard normal distribution  $\mathcal{N}(0, 1)$ . The auxiliary variable  $\epsilon$  allows gradients to flow backwards through the network [23].



**Fig. 3.** The decoder reconstructs the sequence of one-hot vectors  $\tilde{o}$  that encode a program given its latent representation  $z$ .

The decoder in Fig. 3 reconstructs the program given  $z$ . We implement the decoder as a forward GRU network. The outputs are one-hot vectors  $o^r$ , which indicate the predicted production rule at a given timestep. Two inputs are

provided to the GRU cell  $\vec{G}(\cdot)$  at each timestep. The first input is the previous hidden state (initialised to  $\vec{h}_0 = z$ ). The second input is the vector  $\tilde{e}_{t-1}$  concatenated with  $z$ , where  $\tilde{e}_{t-1}$  is the embedding of the production rule from the previous timestep (initialised to  $\tilde{e}_0 = e_0$ ). Utilising an autoregressive input helps the decoder keep track of previously selected production rules.

Recall that the program in Fig. 2b is described by one-hot vectors  $\tilde{o} = (o_0, o_2, o_5, o_8)$ , mediating the expansion  $\langle \text{start} \rangle \rightarrow \langle \text{code} \rangle \rightarrow \langle \text{int} \rangle \rightarrow 3$  plus a final ‘EOS’ token. A sequence of reconstructed one-hot vectors  $o^r$  are computed by the decoder as follows:

- $t = 0$ : we can set  $o_0^r = [1, 0, 0, 0, 0, 0, 0, 0]$  since all derivation trees have  $\langle \text{start} \rangle$  at their root node.
- $t = 1$ : the non-terminal to be expanded at  $t = 1$  is  $\langle \text{code} \rangle$ . In order to select a production rule, the GRU cell emits a hidden state  $h_1$  from which the logit  $l_1 = \text{softmax}(\text{mask}(W_{hl}h_1 + b_l))$  is computed. The logit defines a probability distribution over production rules – a mask is applied because  $\langle \text{code} \rangle$  can only be expanded using rules 1 or 2 (see Fig. 2a). Since the maximum value of  $l_1$  occurs at index 2, it follows that rule 2 is selected and  $o_1^r = [0, 0, 1, 0, 0, 0, 0, 0]$ .
- $t = 2$ : the  $\langle \text{int} \rangle$  non-terminal can be expanded using rules 3, 4, or 5. Rule 5 is selected giving  $o_2^r = [0, 0, 0, 0, 0, 1, 0, 0]$ .
- $t = 3$ : finally the ‘EOS’ token is reached indicating that the decoding process has terminated, and  $o_3^r = [0, 0, 0, 0, 0, 0, 0, 1]$ .

At test time, we disregard the encoder and engage the decoder as a generative model to search for fit programs. A program is constructed by passing a point  $z$  in the latent space to the decoder. Non-terminals are expanded in a depth-first manner using the one-hot vectors (that is, production rule choices) produced by the decoder.

**VAE Loss Function:** In summary, a program is described by a sequence of one-hot vectors (production rules)  $\tilde{o}$  and their corresponding embeddings  $\tilde{e}$ . We propose to learn a continuous latent representation of programs using a VAE. The encoder  $q_\phi(z|\tilde{e})$  maps  $\tilde{e}$  to a latent code  $z$ . The decoder  $p_\theta(\tilde{o}|z)$  is a generative model that reconstructs  $\tilde{o}$  given  $z$ . Parameters  $\phi$  and  $\theta$  are jointly optimised via gradient descent on the loss function:

$$\mathcal{L}(\phi, \theta; \tilde{o}, \tilde{l}, z) = \mathcal{L}_{\text{AE}}(\tilde{o}, \tilde{l}) + \mathcal{L}_{\text{REG}}(z), \quad (2)$$

where  $\mathcal{L}_{\text{AE}}(\tilde{o}, \tilde{l})$  denotes the reconstruction loss, and  $\mathcal{L}_{\text{REG}}(z)$  is a regularisation loss encouraging latent codes to be normally distributed.

The reconstruction loss is defined as the cross entropy between  $\tilde{o}$  and the logits  $\tilde{l}$  (see Fig. 3) emitted by the decoder:

$$\mathcal{L}_{\text{AE}}(\tilde{o}, \tilde{l}) = -\frac{1}{|\tilde{o}|} \sum_{t=1}^{|\tilde{o}|} \tilde{o}_t \cdot \log_e(\tilde{l}_t).$$

To shape the latent space, we adopt the maximum-mean discrepancy (MMD) regularisation loss proposed by Zhao et al. [41]:

$$\mathcal{L}_{\text{REG}}(z) = \text{MMD}(z, z'),$$

where  $z \sim q_\phi(z|\bar{o})$  is the latent vector produced by the encoder, and  $z'$  is sampled from a multivariate standard normal distribution. The latent space realised by minimising Eq. 2 should exhibit two properties that enable effective search. Firstly, programs should be densely distributed near the origin. Secondly, nearby points should decode to syntactically similar programs (high syntactic locality).

### 2.3 Evolutionary Algorithms

**Evolutionary Algorithm:** An Evolutionary Algorithm (EA) is implemented to search the real-valued space discovered by the VAE. Programs are represented by real vectors considered as locations in the learned representation. Initialisation, mutation, and crossover are defined on real vectors. An initial population is obtained by sampling 1000 individuals  $z$  from a standard normal distribution. This initial population is evolved over 300 generations as follows.

In every generation, individuals are assigned a fitness by decoding  $z$  to give a derivation tree (program), which is then evaluated on the program synthesis task. Fit programs are selected using tournament selection or lexicase selection [19]. Selected individuals undergo mutation and crossover. An individual  $z$  is mutated by adding to it a vector  $\Delta z$  sampled from a standard normal. Crossover is applied to every pair of selected individuals. Elements are marked for crossover with probability 0.1. Hence, marked elements  $m$  in parents  $p^1$  and  $p^2$  are interpolated to yield children  $c^1$  and  $c^2$  such that:

$$\begin{aligned} c_m^1 &= p_m^1 + i_1 \times (p_m^2 - p_m^1), \\ c_m^2 &= p_m^2 + i_2 \times (p_m^1 - p_m^2), \end{aligned}$$

where  $i_1$  and  $i_2$  are drawn from a uniform distribution  $\mathcal{U}(0, 1)$ . Seven elites enter the next generation without undergoing crossover or mutation.

**Hill Climbing:** The EA is benchmarked against a hill climbing algorithm in order to assess the need for population-based search. Here, a single individual  $z_{\text{best}}$  is initialised and evaluated. A new hypothesis  $z_{\text{hyp}}$  is generated by adding a sample from a standard normal  $\Delta z$  to  $z_{\text{best}}$ . Hypotheses are evaluated over  $\text{pop size} \times \text{gens} = 1000 \times 300$  iterations. After every iteration,  $z_{\text{hyp}}$  replaces  $z_{\text{best}}$  if the corresponding program attains a better fitness.

**Genetic Programming:** The EA and hill climbing algorithm perform search in a continuous latent space. By contrast, Grammar-based Genetic Programming (GP) [7, 29] explores the discrete space of derivation trees directly. An initial population of 1000 randomly generated derivation trees is formed using the ramped

half-and-half method. In every generation, individuals which are selected using tournament selection undergo subtree mutation and crossover. Subtree mutation replaces a randomly selected subtree with a new randomly generated subtree. Subtree crossover swaps randomly selected subtrees (with the same root node) between two parents. Every individual undergoes mutation, and the crossover probability is 0.9. We use generational replacement with elitism (the elite size is 7). Derivation trees are allowed to grow to a maximum depth of 16.

### 3 Experimental Setup

Within the GP community, a program synthesis benchmark suite has been proposed [18], composed of 29 problems which might typically be assigned as exercises to beginner programming students. The problems are all specified as word problems, with recommendations for generating correct input/output pairs and a train/test split. They require the use of multiple data types and control structures including loops. Recent work on program synthesis has made good progress on this suite [9, 10, 17, 19]. In this proof of concept study we examined six problems of varying difficulty drawn from the suite: `grade`, `last_index_of_zero`, `median`, `negative_to_zero`, `smallest`, and `vectors_summed`.

For each problem, VAEs were trained using training and development sets containing 49000 and 1000 programs respectively. Programs were sequences of production rules (encoded as one-hot vectors) sampled from a grammar. The grammars (one per problem) were assembled based on the data types required to solve a problem (see Sect. 2.1 and in [8]). The best VAE from ten independent runs was combined with the EA from Sect. 2.3 to enable program discovery. The hyperparameters displayed in Table 1 were determined by trial-and-error.

**Table 1.** VAE hyperparameter settings.

Epochs	100
Initial learning rate	0.01
Learning rate decay rate (per epoch)	0.95
Batch size	128
Dimensionality of the latent space	50
Dimensionality of the hidden states	50
Dimensionality of the embeddings	50
Optimisation algorithm	RMSprop
Gradient clipping	Norm of gradients $\leq 0.00001$
Model selection based on	Development set loss

The evolutionary algorithms outlined in Sect. 2.3 were deployed on six problems drawn from the benchmark suite. For a given problem, 100 independent runs of the EA-VAE and GP algorithms were carried out. Tournament selection was adopted in one set of runs, and lexicase selection was used in another set. Similarly, 100 runs of the hill climbing algorithm were executed for each problem.

## 4 Results and Discussion

We compare the proposed algorithms for automatically synthesising Python code in this section. Success rates are reported on the training and test sets of six problems drawn from the benchmark suite. We illustrate how transitions between neighbouring points in the latent space map to smooth syntactic transitions in program space. Finally, analysis of the fitness landscape reveals why some problems are harder for the EA-VAE to solve than others.

### 4.1 Success Rates

The EA-VAE and GP success rates are displayed in Table 2. Both algorithms solve more problems when lexicase selection is used to select parents; GP solves all six problems, while the EA-VAE discovers solutions for every problem except `grade`. Neither algorithm solves `vectors_summed` under tournament selection, but they both find solutions under lexicase selection.

**Table 2.** The reported results include: the success rates (out of 100 runs) on training and test sets, median number of production rules consumed when generating programs, and the median generation at which solutions were discovered. Results are given under tournament selection and lexicase selection.

Problem	EA-VAE				GP			
	Train	Test	Rules	Gen	Train	Test	Rules	Gen
<code>grade</code>	0	0	89	NA	10	<b>4</b>	404	204
<code>median</code>	97	<b>97</b>	22	50	77	27	329	102
<code>last_index_of_zero</code>	5	5	28	155	16	<b>14</b>	299	104
<code>negative_to_zero</code>	83	<b>83</b>	18	33	50	47	314	13
<code>smallest</code>	100	<b>100</b>	22	18	100	86	149	12
<code>vectors_summed</code>	0	0	23	NA	0	0	200	NA

(a) Tournament Selection.

Problem	EA-VAE				GP			
	Train	Test	Rules	Gen	Train	Test	Rules	Gen
<code>grade</code>	0	0	141	NA	85	<b>31</b>	362	97
<code>median</code>	100	<b>100</b>	22	24	100	49	214	14
<code>last_index_of_zero</code>	2	1	46	147	33	<b>30</b>	267	91
<code>negative_to_zero</code>	64	64	43	69	72	<b>68</b>	223	21
<code>smallest</code>	100	<b>100</b>	23	12	100	89	86	4
<code>vectors_summed</code>	7	7	69	99	20	<b>14</b>	270	120

(b) Lexicase Selection.



**Table 3.** The proposed approach is benchmarked against hill climbing (HC-VAE), tree-based GP, and PushGP. Success rates are reported on the test sets of each problem. Lexicase selection was used in the EA-VAE, GP, and PushGP runs. The results for PushGP are taken from [18].

Problem	EA-VAE	HC-VAE	GP	PushGP
grade	0	0	<b>31</b>	4
median	<b>100</b>	1	49	45
last_index_of_zero	1	0	<b>30</b>	21
negative_to_zero	64	0	<b>68</b>	45
smallest	<b>100</b>	24	89	81
vectors_summed	7	0	<b>14</b>	1

Programs evolved by the EA-VAE algorithm typically generalise perfectly from train to test cases. The EA-VAE generalises well because it gives rise to near minimal programs. Comparing the columns labelled “Rules” in Tables 2a and 2b, we see that GP consumes many more production rules than the EA-VAE. That is, GP is more susceptible to bloat. Introns may be beneficial to GP during evolution [31], but their presence impacts generalisation to the test sets.

The success rates displayed in Table 3 confirm that the EA outperforms greedy hill climbing. A fitness landscape analysis will reveal why the latent space is not amenable to greedy search. The EA-VAE achieves the highest success rates on two problems. However, GP is the most consistent algorithm overall, finding multiple solutions to every problem. Unlike PushGP, the grammar-based techniques generate interpretable Python programs, such as those in Fig. 4.

```

in0.insert(i1,i0)
in1.insert(i1,max(i0,i2))
in1.insert(i1,max(i0,i2))
in1.insert(max(i1,i0),getIndexIntList(in0,i2))
in1.insert(getIndexIntList(in0,int(3.0)),getIndexIntList(list(saveRange(i1,i0)),getIndexIntList(res0,i0)))
its = 0nfor i1 in in1: {:min0.insert(max((i0-i0),getIndexIntList(res0,i0)),min(divInt(i1,i0),abs(i0)))nif its > 100: {:\nbreak\n:}its += 1n;}
its = 0nfor i1 in in1: {:nres0.append(max((i0-i0),divInt(i0,i1)))nif its > 100: {:\nbreak\n:}its += 1n;}
its = 0nfor i0 in in1: {:nres0.append(max((i0-i1),min(i0,i0)))nif its > 100: {:\nbreak\n:}its += 1n;}
its = 0nfor i0 in in1: {:nres0.append(max((divInt(i0,i1)+min(i0,i0)),getIndexIntList(in0,len(in0))))nif its > 100: {:\nbreak\n:}its += 1n;}
its = 0nfor i0 in in1: {:nres0.append((max(divInt(i0,i1),min(i0,i0))+getIndexIntList(in0,len(res0))))nif its > 100: {:\nbreak\n:}its += 1n;}
    
```

(a) `vectors_summed`

```

b0 = -in0 > i0
b0 = in0 != min(in2,i0)
res0 = min(in0,in2)
res0 = max(in0,min(in2,i0))
res0 = max(min(in0,in2),i0)
res0 = min(max(in0,in2),i0)
res0 = min(max(in0,in1),in2)
res0 = min(max(in0,in1),min(in2,in0))
res0 = min(max(in0,in1),in2)
res0 = min(max(in0,in1),max(in2,min(in0,in1)))
    
```

(b) `median`

```

i0 = in1
res0 = in1
res0 = in1
res0 = min(in1,in2)
res0 = min(in1,in2)
res0 = min(in2,in1)
res0 = min(in2,mod(in1,i1))
res0 = min(mod(in2,in1),min(i1,res0))
res0 = min(mod(in2,in1),min(i1,in1))
res0 = min(mod(abs(in2),min(in1,i1)),min(min(in1,in0),min(in2,in3)))
    
```

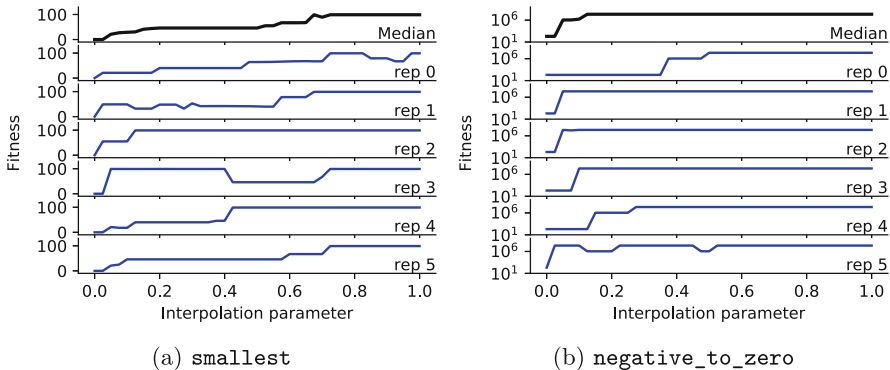
(c) `smallest`

**Fig. 4.** We interpolate between a random point in the latent space  $\mathcal{Z}_r$ , and one of the solutions found the EA  $\mathcal{Z}_{EA}$ . Programs are displayed for points  $\mathcal{Z}_r + \delta(\mathcal{Z}_{EA} - \mathcal{Z}_r)$ , where  $\delta \in [0.0, 0.1, \dots, 1.0]$ . Note that the “\ n” symbols indicate line breaks.

The interpolations in Fig. 4 suggest that the VAE learns a relatively smooth and coherent latent space. For example, consider the interpolations for `vectors_summed`. The concept of a for-loop appears, and is retained, as we move closer to the solution  $z_{EA}$ . Further refinements of the loop body yield a program (red text) that achieves the desired semantics: it returns a vector ‘res0’, which is the summation of input vectors ‘in0’ and ‘in1’. Evidence of gradual syntactic transitions implies that the VAE packs programs densely around the origin. This property of the latent space arises due the regularisation term in Eq. 2.

## 4.2 Landscape Analysis

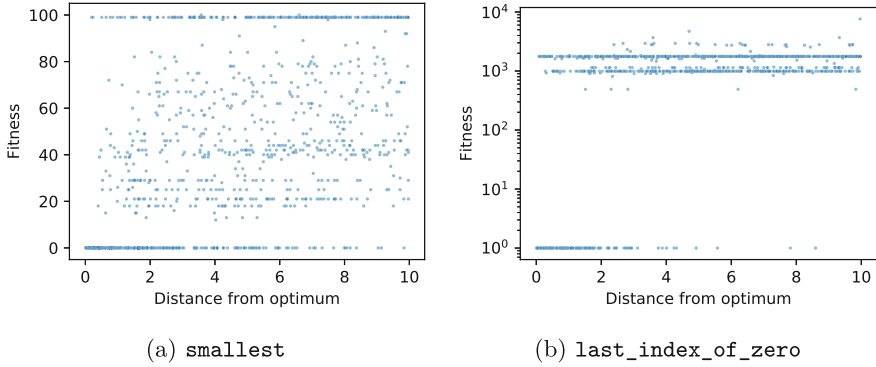
The Cartesian space allows natural methods of landscape analysis. Figure 5 shows how fitness changes over interpolations between solutions (found by EA) and random points (sampled from a standard normal in the VAE latent space). The fitness landscape is characterised by neutrality and discrete steps in fitness. Nonetheless, there is evidence of a positive *fitness-distance correlation* (FDC).



**Fig. 5.** Fitness over interpolations. We show 6 repeats, and the median over 30.

To expand on this evidence, we also present FDC results where points are sampled rather than created by interpolation. In particular, for each trial we randomly choose a solution  $z$  from among those found by the EA, and then sample a random vector  $y$  from a standard normal. Because of the high dimension (50), this gives a strong bias for Euclidean distance  $5 \leq d(z, y) \leq 8$ . A solution is to then scale  $y$  to a desired length, and we have chosen to scale  $y$  so that  $d(z, y)$  is distributed uniformly on  $[0, 10]$ . The scatter plots in Fig. 6 allow us to see the fitness-distance relationship over the whole space, and also focus on the relationship for small distances.

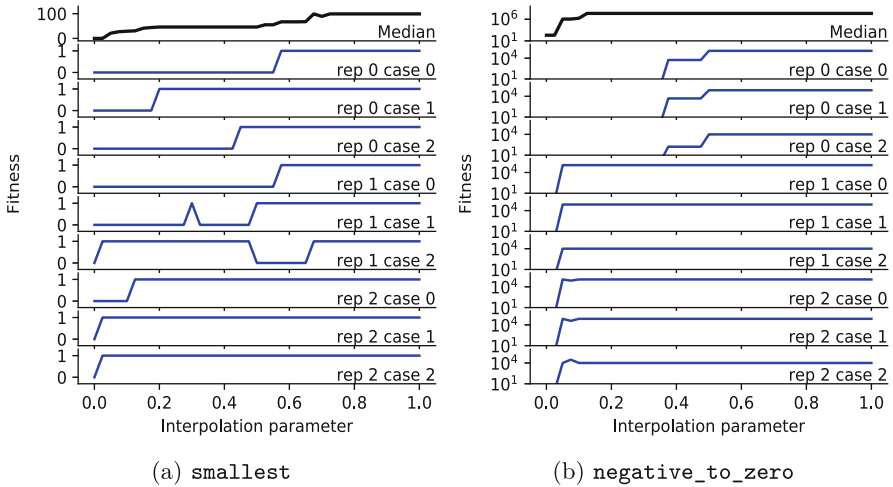
Figure 6 indicates that several solutions exist in the region around a given solution (where a ‘solution’ has fitness 0). Therefore, the EA is not confronted with a needle-in-a-haystack fitness landscape. As expected, increasingly fewer solutions are observed as we move further away from a known solution in the



**Fig. 6.** Fitness against distance. On the right hand side, we have plotted  $fitness + 1$  to allow a log-plot, so  $10^0$  indicates a solution. A few outliers are excluded.

**Table 4.** FDC values, where  $R$  is Pearson’s correlation (excluding outliers), and  $\tau$  is Kendall’s. The `grade` problem is excluded because we found no solutions, and hence cannot compute an FDC value.

Problem	$R$	$\tau$
<code>median</code>	0.30	0.20
<code>smallest</code>	0.23	0.16
<code>negative_to_zero</code>	0.22	0.15
<code>vectors_summed</code>	0.19	0.17
<code>last_index_of_zero</code>	0.07	0.06



**Fig. 7.** Fitness over interpolation: as in Fig. 5, but showing the error on 3 repeats and 3 individual training cases as indexed on the right.

latent space (lower right). Table 4 shows that `last_index_of_zero`, a hard problem for VAE-based search, has FDC near 0, while easier problems show increasingly larger positive FDC values. Thus, FDC partly explains performance.

Because lexibase selection considers errors on individual training cases, it is interesting to consider them separately as in Fig. 7. As expected we see some evidence of correlation among cases.

## 5 Conclusions and Future Work

Variational Autoencoders (VAEs) are effective at learning a coherent continuous representation of discrete programs. Solutions to non-trivial synthesis problems are discovered by searching the VAE’s latent space using an Evolutionary Algorithm (EA). The EA-VAE approach to program synthesis is competitive with tree-based GP and PushGP on problems drawn from the benchmark suite. However, some problems present a neutral and discretised fitness landscape, resulting in lower success rates for the EA-VAE versus the benchmarks.

The algorithm could be improved in a variety of ways. Firstly, it will be interesting to explore techniques for better organising the latent space. One possibility, inspired by Gómez-Bombarelli et al. [15], is to jointly train a multi-layer perceptron (MLP) with the VAE. The MLP could be trained to predict program semantics or the program’s fitness on test cases, given the VAE’s latent layer  $z$  as input. This would encourage program semantics information to be present, and well-structured, in the latent layer. Secondly, a more informative fitness function could be used to guide the search algorithm. We used the raw errors on input/output training pairs. However, program synthesis is not truly a black-box problem. There is a wealth of additional information that can be made available to the search algorithm, such as the program execution trace and the semantics on individual inputs [26, 27]. Finally, state of the art natural language models, such as the transformer [40] or BERT [6], could be easily incorporated into the VAE’s architecture. These ideas can be assessed on the full benchmark suite, and on more recently proposed benchmarks such as the ARC problems [4].

Our approach to program synthesis combines the two dominant paradigms in artificial intelligence: symbolic AI and connectionism. On the one hand, we evolve symbolic programs that can express abstract concepts, generalise perfectly, and that can be interpreted by humans. On the other hand, programs are embedded in the latent space using a neural network. This class of models are adept at pattern recognition, data compression, and representation learning. Discrete search in the space of symbolic programs will be a cornerstone of artificial intelligence research in the coming decades. We believe that hybridising the symbolic and connectionist paradigms is a promising research direction.

**Acknowledgements.** This research is based upon works supported by the Science Foundation Ireland under grant 13/IA/1850.

## References





1. Balog, M., Gaunt, A.L., Brockschmidt, M., Nowozin, S., Tarlow, D.: Deepcoder: learning to write programs. In: Proceedings International Conference on Learning Representations 2017. OpenReviews.net (2017)
2. Orzechowski, P., Magiera, F., Moore, J.H.: Benchmarking manifold learning methods on a large collection of datasets. In: Hu, T., Lourenço, N., Medvet, E., Divina, F. (eds.) EuroGP 2020. LNCS, vol. 12101, pp. 135–150. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-44094-7\\_9](https://doi.org/10.1007/978-3-030-44094-7_9)
3. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint [arXiv:1406.1078](https://arxiv.org/abs/1406.1078) (2014)
4. Chollet, F.: The measure of intelligence. arXiv preprint [arXiv:1911.01547](https://arxiv.org/abs/1911.01547) (2019)
5. De Jong, K.A.: Evolutionary Computation: A Unified Approach. MIT Press, Cambridge (2006)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805) (2018)
7. 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 (2017)
8. Forstenlechner, S.: Program Synthesis with Grammars and Semantics in Genetic Programming. PhD Thesis pp. 162–175 (2019)
9. 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)
10. Forstenlechner, S., Fagan, D., Nicolau, M., O’Neill, M.: Extending Program Synthesis Grammars for Grammar-Guided Genetic Programming. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11101, pp. 197–208. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99253-2\\_16](https://doi.org/10.1007/978-3-319-99253-2_16)
11. Friedberg, R.M.: A learning machine: part i. IBM J. Res. Dev. **2**(1), 2–13 (1958)
12. Friedberg, R.M., Dunham, B., North, J.H.: A learning machine: part ii. IBM J. Res. Dev. **3**(3), 282–287 (1959)
13. Fujiki, C., Dickinson, J.: Using the genetic algorithm to generate LISP source code to solve the prisoner’s dilemma. In: Proceedings of the 2nd International Conference on Genetic Algorithms, Cambridge, MA, USA, July 1987. pp. 236–240 (1987)
14. Gaunt, A.L., et al.: TerpreT: A probabilistic programming language for program induction. CoRR abs/1608.04428 (2016)
15. Gómez-Bombarelli, R.: Automatic chemical design using a data-driven continuous representation of molecules. ACS central science **4**(2), 268–276 (2018)
16. Gulwani, S.: Automating string processing in spreadsheets using input-output examples. SIGPLAN Notices **46**(1), 317–330 (2011)
17. 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, pp. 937–944 (2017)
18. Helmuth, T., Spector, L.: General program synthesis benchmark suite. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1039–1046 (2015)

19. Helmuth, T., Spector, L., Matheson, J.: Solving uncompromising problems with lexibase selection. *IEEE T. Evolut. Comput.* **19**(5), 630–643 (2014)
20. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)
21. Holland, J.H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology. Control and Artificial Intelligence.* MIT Press, Cambridge (1975)
22. Katayama, S.: Recent Improvements of magichaskeller. In: Schmid, U., Kitzelmann, E., Plasmeijer, 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)
23. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013)
24. Koza, J.R.: Human-competitive results produced by genetic programming. *Genet. Program. Evol. Mach.* **11**(3–4), 251–284 (2010)
25. Koza, J.R., Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, vol. 1. MIT press, Cambridge (1992)
26. Krawiec, K., O’Reilly, U.M.: Behavioral programming: a broader and more detailed take on semantic GP. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 935–942 (2014)
27. Krawiec, K., Swan, J.: Pattern-guided genetic programming. In: *Proceedings of the 15th Annual Conference On Genetic And Evolutionary Computation*, pp. 949–956 (2013)
28. Kusner, M.J., Paige, B., Hernández-Lobato, J.M.: Grammar variational autoencoder. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. pp. 1945–1954. *JMLR. org* (2017)
29. Mckay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O’Neill, M.: Grammar-based genetic programming: a survey. *Genet. Program. Evol. Mach.* **11**(3–4), 365–396 (2010)
30. Muggleton, S.: Inductive logic programming: issues, results and the challenge of learning language in logic. *Artif. Intell.* **114**(1–2), 283–296 (1999)
31. Nordin, P., Francone, F., Banzhaf, W.: Explicitly defined introns and destructive crossover in genetic programming. *Adv. Genetic Program.* **2**, 111–134 (1995)
32. O’Neill, M., Fagan, D.: The Elephant in the Room: Towards the Application of Genetic Programming to Automatic Programming. In: Banzhaf, W., Spector, L., Sheneman, L. (eds.) *Genetic Programming Theory and Practice XVI. GEC*, pp. 179–192. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-04735-1\\_9](https://doi.org/10.1007/978-3-030-04735-1_9)
33. O’Neill, M., Ryan, C.: *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language* (2003)
34. O’Neill, M., Spector, L.: Automatic programming: The open issue? *Genetic Programming and Evolvable Machines* pp. 1–12 (2019)
35. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: *A Field Guide to Genetic Programming.* Lulu.com (2008)
36. Rich, C., Waters, R.C.: Automatic programming: Myths and prospects. *Computer* **21**(8), 40–51 (1988)
37. Samuel, A.L.: Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.* **3**(3), 210–229 (1959)
38. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **45**(11), 2673–2681 (1997)

39. Spector, L., Klein, J., Keijzer, M.: The Push3 execution stack and the evolution of control. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 1689–1696 (2005)
40. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
41. Zhao, S., Song, J., Ermon, S.: InfoVAE: Information maximizing variational autoencoders. arXiv preprint [arXiv:1706.02262](https://arxiv.org/abs/1706.02262) (2017)



# Cooperative Co-Evolutionary Genetic Programming for High Dimensional Problems

Lino Rodriguez-Coayahuitl<sup>1</sup>(✉) , Alicia Morales-Reyes<sup>1</sup> ,  
Hugo Jair Escalante<sup>1,2</sup> , and Carlos A. Coello Coello<sup>2</sup> 

<sup>1</sup> Instituto Nacional de Astrofísica, Óptica y Electrónica, 72840 Tonantzintla, Mexico  
{linobi,a.morales,hugojair}@inaoep.mx

<sup>2</sup> CINEVESTAV-IPN, Departamento de Computación, Mexico City, Mexico  
ccoello@cs.cinvestav.mx

**Abstract.** We propose a framework for Cooperative Co-Evolutionary Genetic Programming (CCGP) that considers co-evolution at three different abstraction levels: genotype, feature and output level. A thorough empirical evaluation is carried out on a real-world high dimensional ML problem (image denoising). Results indicate that GP's performance is enhanced only when cooperation happens at an output level (ensemble-like). The proposed co-evolutionary ensemble approach is compared against a canonical GP implementation and a GP customized for image processing tasks. Preliminary results show that the proposed framework obtains superior average performance in comparison to the other GP models. Our most relevant finding is the empirical evidence showing that the proposed CCGP model is a promising alternative to specialized GP implementations that require knowledge of the problem's domain.

**Keywords:** Genetic Programming · Evolutionary machine learning · Co-evolutionary algorithms · Ensemble methods · Image processing.

## 1 Introduction

High dimensional problems have been traditionally challenging for both Machine Learning [2] (ML) methods and Evolutionary Algorithms [31] (EAs). This issue is critical for Genetic Programming (GP), which in this case is *the evolutionary learning algorithm*, because of its complex structures and its need for large populations to reach acceptable solutions. Therefore, large scale learning problems have been out of the scope of GP-based solutions, hindering its raise as a competitive learning model.

In ML, several techniques have been devised to adapt learning algorithms to high dimensional problems. A successful example are convolutional neural networks that process images at a pixel-level [13]. In contrast, a mechanism commonly

---

The last author gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (*Investigación en Fronteras de la Ciencia 2016*) and from a SEP-Cinvestav grant (application no. 4).



adopted in GP to deal with high dimensional problems (e.g., image processing) uses special (high-level) primitives capable of processing groups of features from the input representation altogether. Thus, nodes in GP can represent sets of features or functions to process them (e.g., a mean function over an input space region) [1, 22]. This approach has resulted in satisfactory performance in some domains [7, 16, 26]. However, it usually requires a form of problem’s domain knowledge, which contravenes the spirit of automated ML systems.

From a pure evolutionary computation (EC) standpoint, an approach that has been used for a long time to tackle large scale problems is Cooperative Co-Evolutionary Algorithms [21] (CCA). Potter and De Jong [21] originally proposed the CCA framework to tackle complex optimization problems through Genetic Algorithms (GA) by splitting the search space into multiple, smaller, sub-problems that are solved by semi-independent GA populations cooperating to solve the original (larger) problem. It is well documented that CCA has enabled several EAs to operate on very high dimensional optimization problems [17, 19, 31]. This evidence motivated us to explore CCA’s suitability in a GP context for high-dimensional learning problems.

We propose alternative mechanisms for implementing a Cooperative Co-evolutionary GP (CCGP) and assess their performance in a high dimensional learning problem composed by more than four hundred feature variables: *image denoising*. Our working hypothesis is that a cooperative co-evolutionary approach will allow GP to scale its performance in ML problems with hundreds of input feature variables, without having to resort on high-level primitives. Therefore, the main contributions of this work are threefold:

- We introduce cooperative co-evolution in GP as a way to tackle high-dimensional ML problems. The proposed GP formulation obtains competitive performance in very high-dimensional and complex problems while directly processing raw data (pixels).
- We propose three different approaches to perform cooperative co-evolution in GP aiming at high dimensional problems. We experimentally compare their performance in a real-world, high-dimensional ML problem (natural image denoising).
- The best performing CCGP approach is evaluated extensively and compared to highly competitive baseline algorithms. Our experimental results show the superiority and competitiveness of our proposed approach.

Results indicate that CCGP is a viable alternative to the standard GP approach for high dimensional problems. This is important because in the proposed CCGP scheme, no special nodes are defined for the problem at hand, whereas in the typical GP model for high dimensional problems, those special nodes represent the weakest link in the design process of a GP-based ML solution, mainly because special nodes may require human expert knowledge of the problem’s domain. In contrast, our proposed CCGP approaches are completely agnostic, therefore posing GP a step towards automation, and closer to modern general purpose ML frameworks, such as Deep Learning.

## 2 Related Work

According to [12], when approaching ML problems with GP, there are different abstraction levels to perform CCGP: (1) *genotypic*, (2) subroutine or *feature*, and (3) output or *ensemble* levels. In genotypic CCGP, co-evolving components fusion takes place at the individual representation level, by merging trees directly (in tree-based GP); for CCGP at feature level, co-evolving GP processes generate intermediate input data representations that can be fed into a ML model in order to enhance its performance (i.e., feature extraction). Finally, in a CCGP ensemble, multiple co-evolved species' outputs vote or average in order to achieve higher accuracy in classification or regression problems. In this paper we propose a novel framework to perform cooperative co-evolution at those abstraction levels, and experimentally compare their performance.

CCGP has been mostly studied at ensemble and feature levels [10, 11, 20, 33]. It should be noted, however, that the originally proposed CCA framework considers fusion at a genotypic level [21]. CCA research at a genotypic level in GP is scarce; one of the few works that covers this subject is presented in [12]. Krawiec and Bhanu presented several works on feature-level CCGP [10, 11], and in [12] they proposed a genotype-level CCGP and compared it to their previous approaches. However, it should be noted that [12] covers CCGP only for linear GP [3], and not for the original tree-based GP. A possible reason for this lack of interest could be the fact that performing genotype-level CCGP with the standard tree individual representation is difficult, since there is no obvious way to fuse genotypes for tree-based individuals other than standard GP-subtree crossover, and this might not yield the desired effect in a CCA scheme. Moreover, while in [10] and [11], cooperative co-evolution happens at feature level, the prediction stage is relegated to a different, simple ML algorithm, instead of using another GP process. In contrast, we propose to perform fusion at a feature level through a co-evolving GP species, thus effectively implementing a multi-layer GP system. This multi-layer GP is another relevant contribution of this paper.

On the other hand, GP-based ensembles have been proposed at least as early as in [9]. Many GP-based ensembles found in the literature follow a standard weighted averaging fusion technique [9, 28, 29]. In contrast, herein we propose to generate ensembles through an explicit co-evolutionary framework. Co-evolution generated GP ensembles have been thoroughly researched by Heywood et al. [6, 14, 15, 18]. However, their problem decomposition technique happens mainly at sample subset level, whereas we propose a feature subset approach.

Regarding GP approaches to tackle high dimensional problems, two are the most widely used: (a) using special primitives to process groups of features altogether [1, 22], and (b) using separate GP processes after applying clustering to input variables [23, 27]. The methods proposed here fall in the second category. Both approaches have disadvantages. In the first, it is required to define special nodes, and this might imply requiring some previous knowledge of the problem's domain, while in the second, division and execution of multiple GP processes may involve an increase in the computational cost. A main contribution in this paper is that, for the first time both GP approaches are directly compared.

### 3 Problem Statement

In supervised learning, ML algorithms search for a function  $f$ , mapping inputs ( $\mathbf{x} \in \mathbb{R}^n$ ) to outputs ( $y \in \mathbb{R}$ ) starting from a dataset of input-output pairs ( $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i \in 1, \dots, d}$ ). For a regression problem with inputs in an  $n$ -dimensional space we have  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ . That is,  $f$  receives as input  $n$  feature variables in order to make a scalar prediction. GP as a non-parametric ML method builds  $f$  from scratch by using primitives and feature variables as building blocks.

In high-dimensional learning problems,  $n$  is large enough so that  $f$  can become difficult to infer because GP needs to search among large tree structures that accommodate enough  $n$  feature variables in order to perform satisfactorily. Hence, this simple problem can be associated to an intractable search space.

We claim that it is less complex to search for multiple, *simpler*, functions  $\tilde{f}_i$ , such that by combining their outputs, they may outperform  $f$ . In our context, by *simpler* we mean that they are represented by smaller GP trees, and are restricted to a limited subset of features, and therefore can be more easily discovered by GP. In formal terms, our hypothesis is that it is computationally more efficient to search for  $p$ , lower dimensional, sub-functions  $\tilde{f}_i : \mathbb{R}^m \rightarrow \mathbb{R}$ , such that  $m \ll n$ , that when combined can yield a  $\tilde{f}$  function equivalent or even with superior performance than  $f$ . Two questions arise: (1) how can the original feature space be split?, and (2) how can we combine  $p$  sub-functions  $f_i$ ?

In this work, we hold that in order to get a highly automated and agnostic ML design process, the feature space should be split in a random way. However, we recognize that some previous knowledge of the problem's domain can be used in order to perform an advantageous partition of the initial input representation. On the other hand, a correct method to merge partial solutions can be more difficult to assert which is the main topic in this research. In the next section we propose and discuss some possible approaches to address such task.

### 4 Cooperative Co-Evolutionary GP

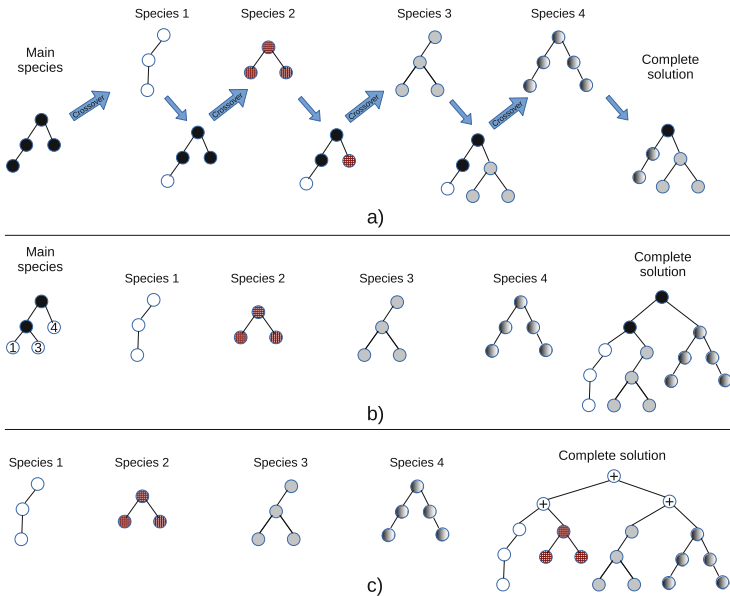
In general terms, CCAs split the search task into multiple, smaller, optimization processes. The main idea is to introduce *modularity* in EAs [21]. In combinatorial and numerical optimization problems, CCAs achieve this by distributing solutions' segments among a number of sub-populations; individuals' evaluation in each sub-population is performed by importing those segments from other sub-populations and assembling complete solutions for evaluation. Thus, individuals take turns to form part of such complete solutions and credit can be assigned to each one of them.

In order to import such problem decomposition strategy into the context of syntax tree-based GP (i.e., not LinearGP), we propose to introduce the concept of *main species*, that represents a partial solution that acts as a *holder* to which the rest of partial solutions attach to, in order to form a complete solution. Next, we detail three proposed approaches developed within the CCGP framework herein introduced. Each method is a CCA with standard tree-based

GP representation at genotype, feature or output level of the ML pipeline. The proposed methods adhere to the following procedure:

1. The input feature set is sampled, with replacement, to form  $p$  subsets with  $m$  randomly picked features variables each;
2.  $p$  species are created; each subset limits valid terminals for each species; all species are confined to one subpopulation (i.e., no inter-species breeding);
3. Additionally, there is a *main* species, that represents the type of individuals to which all other species *attach* to form a complete candidate solution;
4. At each generation, a complete candidate solution is assembled by randomly selecting one individual from each species and attaching them to one main species individual (also selected at random);
5. This complete candidate solution is sent to all subpopulations; each individual is evaluated by attaching it to the complete candidate solution (in order to form a complete chain of execution), for fitness assignment; this also applies to the main species individuals. Details of this procedure are given in Sect. 4.4;
6. The evolutionary process (evaluation, recombination, selection) occurs simultaneously in all subpopulations;

The differences among the proposed CCGP variants rely on the complete solutions assembly process and on the form that the main species take. These variants are described in detail next.



**Fig. 1.** Proposed CCGP models. From top to bottom: genotype-, feature- and ensemble-level models.

### 4.1 Genotype Level

CCGP at genotype level occurs by fusing multiple sub-components while directly mixing syntax trees that represent each co-evolving species. Since the straightforward method to perform such recombination is the standard subtree crossover defined for GP, this is the method we propose to build complete solutions. Thus, subtree crossover is performed sequentially between a main species candidate and each additional co-evolving species. The idea here is that complete candidate solutions have useful subtrees that rely on a wide variety of input variables (because each species is limited to a certain subset of input variables). This is the most straightforward form that CCGP may take, and it is aligned to the original CCA framework proposed by Potter & De Jong. Figure 1a shows the complete solution assembly process under this approach.

### 4.2 Feature Level

We call fusion at *feature level* when species (other than the main one) represent encapsulated sub-components, that is, complete GP trees that merge to form a complete solution by connecting to leaf nodes of a main species individual. This scheme removes inconsistencies of genotype-level/fusion by crossover, and preserves integrity of both species and the main species individuals. Our aim is that co-evolved sub-components may represent *pseudo*-subroutines that act as a sort of feature extractor stages, while the main species act as the predictor stage that operates over those pre-processor stages' outputs, rather than having to work with a raw and large scale input space (hence the name *feature-level*).

In formal terms, this approach fuses  $p$  sub-functions  $f_i$ , by searching for a function (also by means of co-evolution)  $g : \mathbb{R}^p \rightarrow \mathbb{R}$ , that operates over auxiliary sub-functions outputs such that  $\tilde{f} = g(f_1, f_2, \dots, f_p)$ . Figure 1b shows this model.

### 4.3 Ensemble Level

At an ensemble or output CCGP level, each species represents a complete predictor to the problem at hand, and species fusion occurs by only aggregating the output generated by each predictor. Aggregation may take several forms. In this work, we propose to combine each species output by means of a simple sum. This approach bears some resemblance to ML ensemble methods [4] that combine multiple predictors outputs by a weighted sum (hence the name, *ensemble-level*). We proposed this model after observing an undesirable phenomenon in the feature-level CCGP where main individuals' function converged ignoring all but one of the sub-component species, and the search process then happened only in a single population, losing model's co-evolutionary nature and becoming a standard EA (this issue is detailed and discussed in Sect. 5.3). Therefore, through this approach, it becomes more difficult for species to avoid contributing to the global solution. Notice how this approach can be seen as a case where the main function is fixed to a GP tree composed by sum nodes only, and the evolutionary search for a main function is discarded. Figure 1c depicts this concept. Thus, formally, this method proposes that  $\tilde{f} = f_1 + f_2 + \dots + f_p$ .

#### 4.4 Fitness Assignment

Fitness assignment in CCGP (step 5 in the general procedure) is carried out in two different ways. At the Genotype level, attaching each individual to a complete candidate solution that is sent to each species population, occurs by performing crossover between that assessed individual and the complete solution. In CCGP models at Feature and Ensemble level, attachment of individuals for evaluation happens by *replacing* the corresponding individual of that species in the complete candidate solution being used.

## 5 Experimental Results

This section describes the empirical methodology followed for evaluation of proposed CCGP approaches. It also presents and discusses the obtained results.

### 5.1 Datasets

For validation, the proposed CCGP framework tackles image denoising as a high dimensional problem, where a clean image  $\mathbf{x}$  is extracted from a noisy observation  $\mathbf{y}$  such that, for an additive noise model,  $\mathbf{y} = \mathbf{x} + \mathbf{v}$ , where  $\mathbf{v}$  is a contamination process. In this study, Additive White Gaussian Noise (AWN) is targeted, where  $\mathbf{v}$  follows a Gaussian distribution with some given  $\sigma$ .

We used the Berkeley Segmentation Dataset (BSDS) [24] for training and testing purposes. We converted 200 images from BSDS to grayscale and randomly extracted 14,000 patches of  $21 \times 21$  pixels in size. We contaminated images (prior to patch extraction) with AWN noise level  $\sigma = 50$ . We set all GP variants to attack image denoising as a regression problem: the objective function is the minimization of the average mean square error (MSE), from attempting to predict the noise level in the central pixel from all patches in the training set. In a real life scenario, a generated model with this approach can be slid through a full image, in a convolutional fashion, in order to clean it. However, for this set of experiments, we limited ourselves to test generated models in a testing set comprised also by image patches. We used 12,000 patches for training and 2000 for testing. BSDS had been used as testbed for different image denoising methods [5, 25, 30], including deep learning approaches [32], to which we compare later in Sect. 5.4. Figure 2 shows sample images from BSDS.



**Fig. 2.** Sample images from the Berkeley Segmentation Dataset.

## 5.2 Parameters Settings

Table 1 summarizes parameters configuration for all experimental samples. Max, min and mean primitives are 2-arity functions that operate over two single scalars, and division is protected such that any attempt at dividing by zero returns 0. Both crossover and mutation are protected so that the maximum allowable tree depth is never exceeded. Training datasets are split in non-overlapping minibatches of 300 instances and at each generation, populations are evaluated using one minibatch. Minibatch-based evolution (on-line learning) in GP has been found to be successful for this type of ML problem [8,23].

**Table 1.** Parameters configuration for empirical testing.

Parameter	Value
Pop size	400 (per species, inc. main)
Generations	Variable (24 h.)
No. of species	8 + main
Max Tree Depth	6 (for all species, inc. main)
Crossover/Mutation rate	0.5 / 0.5
Pop dynamics	Steady state
Primitives	$+$ , $-$ , $\times$ , $\div$ , $x^2$ , $\sin$ , $\cos$ , $\sqrt{\quad}$ , max, min, mean, ReLU
Terminals	Individual pixels and constants within range $[-1, 1]$
Features per species	30 (from a total of 441)

In order to allow a fair comparison, all setups are run for the same limited amount of time. Subsets of feature variables that are allowed for each species are randomly assembled. However, since the target task predicts central pixel’s noise level within image patches, it is set, as a requirement, that a central pixel appears in at least two random subsets.

For the feature-level approach, another restriction is defined: the main species’ individuals must use, at least, 6 out of 8 total sub-component species. Main species’ trees are parsed and the number of different species used by an individual are counted. If this constraint is not met, the fitness of such individual is set to  $\infty$ . This restriction is set in order to prevent the main species’ individuals from becoming a “wired” function that connects to a single sub-component species, where the whole optimization process is confined to, while the rest of the species do not contribute to the co-evolutionary search.

## 5.3 Analysis of Results

Table 2 shows results obtained for all approaches tested after performing 10 independent runs. Results are shown in decibels Peak Signal to Noise Ratio (dB PSNR), such that higher is better. Figure 3a depicts the fitness evolution in a

feature-level run, in error terms; in this case, lower is better. This behavior is representative of feature-level runs in general. This result shows that the feature-level approach does converge, suggesting that this CCGP variant is capable of evolving multi-layer GP structures with sequential dependencies. However, a closer examination to the best solutions rendered by this approach revealed that this is not the case (discussed below). Moreover, the feature-level approach is no match for ensemble models, which converge faster, and to better solutions, both in average and overall. Meanwhile, genotype-level is left further behind. This result indicates that ensemble CCGP is the best overall method.

Low performance of genotype-level CCGP can be explained by the fusion mechanism used in this approach: subtree crossover is a stochastic operation that even when given the same two parents trees, may render different offspring if performed multiple times. This means that even if a combination of different species individuals that rendered a good complete solution in a previous generation, are chosen again to form a complete solution, this time their merge may result in a bad complete solution; instability in this mechanism is very high for this approach to converge to any acceptable solutions.

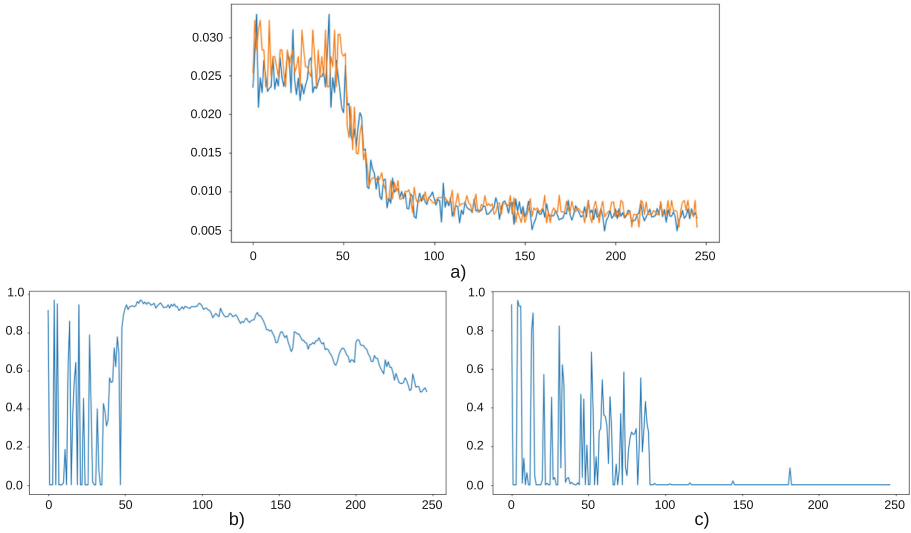
**Table 2.** Results in dB for tested setups with different time frames.

	Ensemble		Feature		Genotype	
	Avg	Best	Avg	Best	Avg	Best
4 h	<b>18.90 ± 0.52</b>	<b>19.46</b>	16.43 ± 1.64	18.73	14.80 ± 0.44	15.41
12 h	<b>20.62 ± 0.36</b>	<b>21.16</b>	17.67 ± 2.30	20.84	15.09 ± 0.25	15.41
24 h	<b>20.96 ± 0.50</b>	<b>21.61</b>	18.55 ± 2.27	21.35	15.09 ± 0.44	15.62

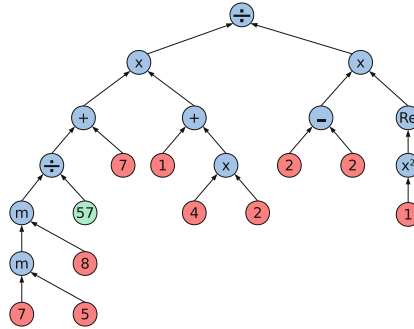
Considerable higher performance at ensemble-level with respect to feature-level can be more arguable. Flexibility at feature-level, which also searches for an optimal main function, could play on its favor against a more restricted ensemble model with its predefined main function - given enough computational time. However, this appears not to be the case. On the contrary, this disadvantage is clearer in the first few generations of feature-level runs, when the evolutionary search appears to stagnate (according to Fig. 3a) apparently searching for a minimally acceptable main function. Even once feature-level CCGP variant escapes initial stagnation, it does not converge to solutions that reach performance of ensemble-level models.

Some dominant solutions are closely examined during this early evolutionary process, and phenotypic diversity is measured in each sub-population. It is observed that feature-level CCGP is surprisingly good at ignoring restriction imposed to the main species' individuals of using co-evolving sub-components: this CCGP variant managed to generate main species' individuals with subtrees that referenced too many sub-components, while at the same time completely ignored such subtrees. Figure 4 depicts an example of such type of individuals.





**Fig. 3.** Fitness and diversity from a feature-level CCGP run as generations elapse. (a) Fitness error (lower is better). In blue (orange) we show the training (testing) error. (b) and (c), phenotypic diversity in two sub-component populations, one (zero) means all solutions yield a different (the same) prediction. (Colour figure online)



**Fig. 4.** Main species' individual example that complies with referencing at least 6 sub-components and not using them. Notice that root's right subtree always returns 0; thus, the whole tree also returns a constant.

Analyzing phenotypic diversity in species' sub-populations confirmed this behavior. Figure 3b shows diversity in a sub-population as generations elapse. It can be observed that at early generations, diversity oscillates between some actual value and zero. When phenotypic diversity abruptly decays to zero, this implies that all individuals in the population generate the same prediction/fitness. This happens because main function individuals are ignoring this sub-component, i.e. regardless of the individual from this population which

connects to the main functions, they all render the same result. This oscillating behavior in the first few generations depicted in Fig. 3b means that the main function individuals do not make a consistent use of this particular sub-component. It is only after a certain number of generations that diversity recovers and the approach begins to behave consistently, because now the main function individuals are working cooperatively with individuals from these species.

Further investigation on diversity of all species in different runs, revealed that by relying on mechanisms such as the one depicted in Fig. 4, the feature-level main individual still converges to “wired” functions that simply connect to one or two sub-component populations where optimization is happening, while the rest of the species do not contribute to the co-evolutionary process. As an example of this behavior, Fig. 3c shows diversity of a species’ population and the moment it begins to be ignored by the main function individuals, which roughly coincides with the time at which overall CCGP run begins to converge. Meanwhile, ensemble-level models exhibited the exact opposite behavior, where in general, only one or two of eight sub-populations decay to zero diversity, suggesting that they may be acting as constant biasing factors in the sum structure, while the rest of the species do alter global solution performance, contributing to the co-evolutionary search. These results show that multi-layer GP architectures remain as very challenging to evolve.

#### 5.4 Other GP Approaches Comparison

In this section, ensemble CCGP performance (the best performing model) is compared against two other GP models: a canonical GP representation, and a modern GP variant that makes use of special nodes aimed at high dimensional problems. The aim is twofold: (1) to collect evidence that supports our hypothesis that cooperative co-evolution can boost GP performance in high dimensional learning problems, as well as (2) to quantify how CCGP compares with respect to GP models tailored for tackling high-dimensional problems.

Two GP models called Low-level GP (LowGP) and Mid-Level GP (MidGP) are implemented. LowGP is a canonical GP that operates at individual pixel level; while MidGP uses special nodes and terminals that allow to process features groups. Table 3 summarizes the configurations used for these approaches. LowGP can only make use of primitives and terminals (the same used by CCGP), while MidGP can use special primitives, special terminals, and regular primitives and terminals. Special primitives are functions that receive as inputs variable-length vectors and whose output is a single scalar that can be processed by regular primitives. For an in-depth analysis of these GP models refer to [22].

Notice in Table 3 that non co-evolutionary GPs are setup with the same population size to that of a single CCGP species, but the max tree depth is extended in this case, to account for main functions on top of sub-component species, as well as with an increased number of subtrees (equivalent to sub-component species). Thus, a fair comparison against the proposed CCGP model is guaranteed: both LowGP and MidGP individuals can accommodate the same maximum number of nodes to that of a CCGP complete solution.

**Table 3.** Specific tested parameters for non CCGP models

Parameter	Value
Pop Size	400
Max Tree Depth	9
Primitives	Same as in Table 1
Special Primitives	mMean, mMax, mMin, mMed
Terminals	Same as in Table 1
Special Terminals	<i>Trimmers</i>

Table 4 shows the results obtained for both standard GP variants. It is observed that CCGP average performance is superior to both LowGP and MidGP. The proposed approach also presents a lower variance, thus providing evidence that ensemble CCGP is a viable method to step up GP’s performance in high-dimensional learning problems, with the added benefit of not requiring a specialized primitives set. It should be noted, however, that MidGP’s best solution outclasses CCGP’s best result by a considerable margin, indicating that MidGP remains as the reference method to outperform within GP. For a more general comparison, consider that a deep network with 17 hidden layers [32], can score 27.20 dB PSNR given similar training and testing sets.

**Table 4.** Comparison to non-coevolutionary GPs. Expressed in dB; higher is better.

Hrs	LowGP		CCGP		MidGP	
	Avg	Best	Avg	Best	Avg	Best
4	17.81 ± 2.31	20.94	18.90 ± 0.52	19.46	<b>20.21 ± 3.74</b>	<b>23.30</b>
12	18.13 ± 2.55	21.68	<b>20.62 ± 0.36</b>	21.16	20.27 ± 3.74	<b>23.35</b>
24	18.23 ± 2.64	21.81	<b>20.96 ± 0.50</b>	21.61	20.41 ± 3.82	<b>23.30</b>

## 6 Conclusions

This paper proposed and contrasted three different approaches to perform cooperative co-evolution within the GP framework at different abstraction levels: *genotype*, *feature* and *ensemble*. A thorough behavior and performance analysis of CCGP at a feature-level showed that synthesizing multi-layer GP architectures is surprisingly difficult, because GP tends to confine all optimization processes within a single sub-population, effectively losing properties of a true co-evolutionary search. This is an important result that shreds light on some future research guidelines.

For full empirical assessment, conventional GP variants were also compared. We can conclude that CCGP’s performance sits in between a completely agnostic canonical GP, that only processes feature variables at an individual level, and

higher-level GP variants that require problem domain knowledge to a lesser or greater extent. This is a very promising result, because with further research, CCGP could boost agnostic GP models to reach the best performances obtained by higher-level GP variants, or maybe to help reducing the amount of designer input knowledge necessary in higher-level GP models.

**Acknowledgements.** This work was partially supported by CONACyT under project grant A1-S-26314, *Integración de Visión y Lenguaje mediante Representaciones Multimodales Aprendidas para Clasificación y Recuperación de Imágenes y Videos*.

## References

1. Al-Sahaf, H., Song, A., Neshatian, K., Zhang, M.: Two-tier genetic programming: Towards raw pixel-based image classification. *Expert Syst. Appl.* **39**(16), 12291–12301 (2012)
2. Alpaydin, E.: *Introduction to Machine Learning*. MIT press, Cambridge (2009)
3. Brameier, M., Banzhaf, W.: A comparison of linear genetic programming and neural networks in medical data mining. *IEEE Trans. Evol. Comput.* **5**(1), 17–26 (2001)
4. Brown, G.: Ensemble learning. *Encyclopedia Mach. Learn.* **312** (2010)
5. Chen, Y., Pock, T.: Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**(6), 1256–1272 (2016)
6. Doucette, J.A., McIntyre, A.R., Lichodziejewski, P., Heywood, M.I.: Symbiotic coevolutionary genetic programming: a benchmarking study under large attribute spaces. *Genet. Program. Evol. Mach.* **13**(1), 71–101 (2012)
7. Esfahanipour, A., Mousavi, S.: A genetic programming model to generate risk-adjusted technical trading rules in stock markets. *Expert Syst. Appl.* **38**(7), 8438–8445 (2011)
8. Gathercole, C., Ross, P.: Dynamic training subset selection for supervised learning in genetic programming. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) *PPSN 1994*. LNCS, vol. 866, pp. 312–321. Springer, Heidelberg (1994). [https://doi.org/10.1007/3-540-58484-6\\_275](https://doi.org/10.1007/3-540-58484-6_275)
9. Iba, H.: Bagging, boosting, and bloating in genetic programming. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) *Genetic and Evolutionary Computing Conference (GECCO'99)*, vol. 2, pp. 1053–1060. Morgan Kaufmann Publishers, San Francisco, California (July (1999)
10. Krawiec, K., Bhanu, B.: Coevolution and Linear Genetic Programming for Visual Learning. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L.D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A.C., Dowsland, K.A., Jonoska, N., Miller, J. (eds.) *GECCO 2003*. LNCS, vol. 2723, pp. 332–343. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-45105-6\\_39](https://doi.org/10.1007/3-540-45105-6_39)
11. Krawiec, K., Bhanu, B.: Visual learning by coevolutionary feature synthesis. *IEEE Trans. Syst. Man. Cyber.* **35**(3), 409–425 (2005)
12. Krawiec, K., Bhanu, B.: Visual learning by evolutionary and coevolutionary feature synthesis. *IEEE T. Evol. Comput.* **11**(5), 635–650 (2007)
13. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. *The Handbook of Brain Theory and Neural Networks* **3361**(10), 1995 (1995)

14. Lemczyk, M., Heywood, M.I.: Training Binary GP Classifiers Efficiently: A Pareto-coevolutionary Approach. In: Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) EuroGP 2007. LNCS, vol. 4445, pp. 229–240. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-71605-1\\_21](https://doi.org/10.1007/978-3-540-71605-1_21)
15. Lichodziejewski, P., Heywood, M.I.: Coevolutionary bid-based genetic programming for problem decomposition in classification. *Genetic Program. Evol. Mach.* **9**(4), 331–365 (2008)
16. Liu, L., Shao, L., Li, X., Lu, K.: Learning spatio-temporal representations for action recognition: A genetic programming approach. *IEEE T. Cybernet.* **46**(1), 158–170 (2015)
17. Liu, Y., Yao, X., Zhao, Q., Higuchi, T.: Scaling up fast evolutionary programming with cooperative coevolution. In: Proceedings of the 2001 Congress on Evolutionary Computation (CEC'2001), vol. 2, pp. 1101–1108. IEEE (2001)
18. McIntyre, A.R., Heywood, M.I.: Cooperative Problem Decomposition in Pareto Competitive Classifier Models of Coevolution. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 289–300. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78671-9\\_25](https://doi.org/10.1007/978-3-540-78671-9_25)
19. Miguel Antonio, L., Coello Coello, C.A.: Use of cooperative coevolution for solving large scale multiobjective optimization problems. In: 2013 IEEE Congress on Evolutionary Computation (CEC 2013). pp. 2758–2765. IEEE Press (2013)
20. Park, J., Mei, Y., Nguyen, S., Chen, G., Johnston, M., Zhang, M.: Genetic programming based hyper-heuristics for dynamic job shop scheduling: cooperative coevolutionary approaches. In: Heywood, M.I., McDermott, J., Castelli, M., Costa, E., Sim, K. (eds.) EuroGP 2016. LNCS, vol. 9594, pp. 115–132. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-30668-1\\_8](https://doi.org/10.1007/978-3-319-30668-1_8)
21. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994). [https://doi.org/10.1007/3-540-58484-6\\_269](https://doi.org/10.1007/3-540-58484-6_269)
22. Rodriguez-Coayahuit, L., Morales-Reyes, A., H.J., E.: A comparison among different levels of abstraction in genetic programming. In: 2019 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), IEEE (Nov 2019)
23. Rodriguez-Coayahuitl, L., Morales-Reyes, A., Escalante, H.J.: Structurally layered representation learning: towards deep learning through genetic programming. In: Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S., García-Sánchez, P. (eds.) EuroGP 2018. LNCS, vol. 10781, pp. 271–288. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77553-1\\_17](https://doi.org/10.1007/978-3-319-77553-1_17)
24. Roth, S., Black, M.J.: Fields of experts: A framework for learning image priors. In: Null, pp. 860–867. IEEE (2005)
25. Schmidt, U., Roth, S.: Shrinkage fields for effective image restoration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2774–2781 (2014)
26. Shao, L., Liu, L., Li, X.: Feature learning for image classification via multiobjective genetic programming. *IEEE Trans. Neural Net. Learn. Syst.* **25**(7), 1359–1371 (2013)
27. Tran, B., Xue, B., Zhang, M.: Using feature clustering for gp-based feature construction on high-dimensional data. In: McDermott, J., Castelli, M., Sekanina, L., Haasdijk, E., García-Sánchez, P. (eds.) EuroGP 2017. LNCS, vol. 10196, pp. 210–226. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55696-3\\_14](https://doi.org/10.1007/978-3-319-55696-3_14)

28. Veeramachaneni, K., Arnaldo, I., Derby, O., O'Reilly, U.M.: FlexGP. *J. Grid Comput.* **13**(3), 391–407 (2015)
29. Veeramachaneni, K., Derby, O., Sherry, D., O'Reilly, U.M.: Learning regression ensembles with genetic programming at scale. In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pp. 1117–1124 (2013)
30. Yan, R., Shao, L., Liu, L., Liu, Y.: Natural image denoising using evolved local adaptive filters. *Sig. Process.* **103**, 36–44 (2014)
31. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. *Inf. Sci.* **178**(15), 2985–2999 (2008)
32. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: residual learning of deep cnn for image denoising. *IEEE Trans. Image Proces.* **26**(7), 3142–3155 (2017)
33. Zou, X., Bhanu, B.: Human activity classification based on gait energy image and coevolutionary genetic programming. In: *18th International Conference on Pattern Recognition (ICPR 2006)*, vol. 3, pp. 556–559. IEEE (2006)



# Image Feature Learning with Genetic Programming

Stefano Ruberto<sup>1</sup>(✉) , Valerio Terragni<sup>2</sup> , and Jason H. Moore<sup>1</sup> 

<sup>1</sup> Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, USA  
stefano.ruberto@pennterapeutics.com, jhmoore@upenn.edu

<sup>2</sup> Faculty of Informatics, Università della Svizzera Italiana USI, Lugano, Switzerland  
valerio.terragni@usi.ch

**Abstract.** Learning features from raw data is an important topic in machine learning. This paper presents Genetic Program Feature Learner (GPFL), a novel generative GP feature learner for 2D images. GPFL executes multiple GP runs, each run generates a model that focuses on a particular high-level feature of the training images. Then, it combines the models generated by each run into a function that reconstructs the observed images. As a sanity check, we evaluated GPFL on the popular MNIST dataset of handwritten digits, and compared it with the convolutional neural network LENET5. Our evaluation results show that when considering smaller training sets, GPFL achieves comparable/slightly-better classification accuracy than LENET5. However, GPFL drastically outperforms LENET5 when considering noisy images as test sets.

**Keywords:** Genetic programming · Semantic GP · Feature learning · Image classification

## 1 Introduction

Feature learning [49] is an important topic in machine learning, as it powers many classification and knowledge discovery techniques. Such techniques need numeric representations of raw data (i.e., features) that are computationally convenient to process [49]. Feature learning becomes a key task when dealing with raw high-dimensional data (e.g., 2D images, videos and sounds) [49], which lack well-defined features [16, 23, 26]. Manually identifying features from high-dimensional data is often infeasible because it requires expensive human-labor and domain knowledge [1, 36]. As such, automatic feature learning techniques have gained much attention [30, 49].

Most recent automatic feature learners are implemented as (multi-layer) neural networks [49]. However, in principle, an automatic feature learner based on GP would entail two important advantages: (i) GP often does not need large training sets to learn competitive models [33]; and (ii) GP is generally robust to noisy data [27]. Recently, we have seen the first GP feature learners for 2D images [9, 25, 38]. These approaches emulate the behavior of a neural-network

*autoencoder* [18], with multiple GP-evolved models that reconstruct (encode and decode) the pixels of an image [38]. Evolving a dedicated model to reconstruct a single pixel has two main issues: (i) it is computationally expensive, especially with high-resolution images; and (ii) it ignores the (important) spatial relations of pixels, as it evolves each model independently from the ones of adjacent pixels.

This paper presents a **GP Feature Learner**, called **GPFL**<sup>1</sup>, to learn high-level features from 2D images. There are two main differences between GPFL and the previous GP feature learners [9, 25, 38]. First, GPFL does not evolve as many models as the pixels in the image, but one model for each high-level feature of the image. Second, GPFL learns high-level features leveraging the spatial relations of pixels, as it uses the pixel coordinates as inputs of the models.

In a nutshell, GPFL takes in input a training dataset of images and outputs a model represented as a function  $f_{gp}$  that, given a 2D coordinate  $(c_1, c_2)$ , returns a pixel value  $p$  (i.e.,  $f_{gp}(c_1, c_2) = p$ ). As such, GPFL is a generative and unsupervised feature learner. Under the hood, GPFL follows the *dynamic target* approach SGP-DT [41, 42] that executes multiple GP runs (called external iterations). Each external iteration evolves a population of models driven by a dynamic “*target*” that changes at each iteration. Each target is defined as the *residual errors* of the reconstructed images between the previous and current iterations. As such, the next iteration will focus on the characteristics of the images that the previous iteration did not approximated well. Each external iteration outputs a model (called partial model) that focuses on a specific high-level feature of the images. When all external iterations terminate, GPFL creates the final model  $f_{gp}$  by combining with *linear scaling* [17, 43] all the partial models.

As a sanity check, we evaluated GPFL on the popular MNIST dataset of 2D images representing handwritten digits [6]. We evaluated how well the models trained by GPFL capture the relevant high-level features of the MNIST images. Towards this goal, we implemented a classifier that uses the reconstruction error of  $f_{gp}$  to classify the MNIST digits. We compared the GPFL-based classifier with L<sup>5</sup>NET5 [19, 21], the well-known DNN specific for MNIST.

When trained with smaller training sets and evaluated with all 10,000 images in the MNIST test set, GPFL-based classifier achieves a median classification accuracy that is comparable or better than L<sup>5</sup>NET5. For example, when trained with a dataset composed of ten images for each digit, the GPFL-based classifier has a median classification accuracy of 82.81%, while L<sup>5</sup>NET5 of 80.67%.

We also evaluated the noise robustness of GPFL by corrupting the 10,000 images of the test set with five noise levels (of salt type). GPFL always outperforms L<sup>5</sup>NET5 for all five levels (with a classification accuracy improvement up to +40.85%). These are important results, considering that GPFL is one of the first genetic programming attempts to learn high-level features from 2D images.

The remainder of this paper is structured as follows. Section 2 discusses the related work. Section 3 describes the GPFL approach. Section 4 presents our experiments. Section 5 concludes the paper.

<sup>1</sup> We presented a preliminary version of this work in a poster paper [40].



## 2 Background and Related Work

GPFL aims to automatically learn high-level features from 2D images. Following previous work, we use low-level features to refer to the pixel values, and high-level features to refer to conglomerations of related low-level features.

Deep Neural Networks (DNNs) [20] are often used to learn high-level features from high-dimensional data with great success [2, 6]. Because developing DNNs requires labor-intensive architecture engineering, researchers have investigated GP approaches (e.g., NEAT techniques [3–5, 8, 10, 11, 34, 35]) to automate the architecture engineering activity of DNNs. These techniques show promising but still limited results, as finding an optimized DNN architecture largely remains a human activity. Instead of leveraging GP to explore the space of possible DNN architectures, GPFL is a GP feature learner detached from DNNs.

Most GP feature learners for 2D images discover high-level features using hand-crafted or domain-specific features as building blocks [1, 16, 23, 26, 36]. For example, Speeded Up Robust Features (SURF) [7], Histogram of Oriented Gradients (HoG) [31], Gist features [32] and Scale-Invariant Feature Transform (SIFT) [50]. Differently, GPFL learns high-level features from low-level ones (i.e., pixel values) without requiring human-crafted or domain-specific features.

At the best of our knowledge, there are only three attempts of GP feature learners for 2D images that discover high-level features directly from low-level ones [25, 29, 38]. Such attempts, following the success of NN-based autoencoders [18, 28, 45], use GP to emulate the classical autoencoder architecture with *encoder*→*code*→*decoder*. The *encoder* component learns a compact representation (called code) of the low-level features in input. The *decoder* component uses the learned high-level features (i.e., the code) to reconstruct an approximation of the input. We now discuss these three attempts.

Rodriguez-Coayahuitl et al. proposed *Structured Layered GP* (SLGP) [37, 38], which evolves two distinct populations. One population encodes the pixels in input and outputs the *code* (i.e, latent space). The other population decodes the *code* into the reconstructed image. SLGP generates as many encoding GP trees as the size of *code* and as many decoding GP trees as the number of pixels.

McDermott proposed an autoencoder GP similar to SLGP [29]. Differently from SLGP, it relies on two *multi-value linear GP* components [9], one for the *encoder* and one for the *decoder*.

Lensen et al. proposed GPMAL [24], GP technique for *manifold learning* [46], which relates to both SLGP and McDermott’s approach. Manifold learning aims to reduce the dimensions of raw data. This is similar, in principle, to the *encoder* component of most autoencoders, which transforms the input into a lower dimensional code (latent space). GPMAL resembles the *encoder* of SLGP, as it also uses as many GP trees as the number of dimensions of the latent space (the *code* size in SLGP). A later version of GPMAL [25] relies on a Pareto front technique to dynamically select the number of dimensions of the latent space.

Similarly to GPFL, these three techniques are generative approaches that reconstruct 2D images. However, GPFL differs substantially. First, they simulate the classical NN autoencoder architecture with two distinct components:

*encoder* and *decoder*. Conversely, GPFL does not follow the NN autoencoder architecture, and thus it avoids altogether the issue of aligning the two components. Second, previous GP feature learners evolve a GP model for each pixel, which is computationally expensive when dealing with high-resolution images, and does not directly consider the spatial relations among pixels. Notably, GPFL is the first GP feature learner that directly relies on spatial information (pixel coordinates). Third, they require that the number of high-level features (i.e., *code* size) is chosen in advance. Instead, one can run GPFL with an arbitrary number of external iterations (i.e., number of high-level features) and stop at the desired reconstructed error. Moreover, one can re-run GPFL to obtain additional high-level features without discarding the previously learned features.

Although the three previous attempts have been evaluated with MNIST, GPFL is the only one known to classify all ten MNIST digits.

### 3 Genetic Programming Feature Learning (GPFL)

Most high-dimensional data found in nature exhibit correlations among low-level features expressed by the extra dimensions. For 2D images, such correlations are the spatial relations among pixels (being the space the extra dimension). The spatial position of pixels can be extremely useful to express relevant high-level features, as it characterizes the intrinsic properties of the image itself: Two images with pixels of identical values but of different spatial positions can represent two radically different concepts. Indeed, humans recognize patterns and objects by relying heavily on spatial relations [22].

This paper presents GPFL, a GP feature learner for 2D images that relies on both the pixels values and their spatial positions. GPFL outputs a function  $f_{gp}$  (a GP-evolved model) that, given a 2D coordinate  $(c_1, c_2)$ , returns a pixel value  $p$  (i.e.,  $f_{gp}(c_1, c_2) = p$ ). As such, given all coordinates,  $f_{gp}$  reconstructs an image.

Each model (GP individual) is a mathematical (tree-like) expression, with

- (i) *non-terminal symbols*: algebraic operations (+, −, ·, the protected division, *Min* and *Max*) and trigonometric operations (*sine* and *cosine*).
- (ii) *terminal symbols*: variables (the coordinates  $c_1$  and  $c_2$ ) and decimal constants (*ERC* between −1 and 1).

This dictionary of symbols allows GPFL to evolve continuous functions with the coordinates  $c_1$  and  $c_2$  as independent variables. As such, the produced models can encode spatial relations among pixels. This is because the continuity property entails relations on adjacent low-level features (i.e., pixels). Because the protected division, Min and Max symbols introduce discontinuity, the models can also encode spatial relations that are difficult to model in a single continuous function. For instance, by combining multiple (continuous) functions.

The key challenge of using spatial information for feature learning is their variability among images that represent the same concept. For example, when classifying handwritten digits, “1” or “l” are two popular styles for writing the

number one. These styles have different, albeit similar, spatial relations. A single non-parametric function cannot output different pixel values for the same coordinate, and thus cannot encode both styles. GPFL addresses the challenge by parameterizing  $f_{gp}$ , so that changing the parameter values reproduces the observed variability. GPFL defines such parameters as the coefficients of a linear combination of multiple GP-evolved models (called partial models), each focusing on a specific high-level feature of the images.

---

**Algorithm 1: GPFL** implements the *dynamic-target framework* (SGP-DT [14]), \*\* marks the lines representing the novel aspects of GPFL

---

```

input  :  $\hat{y}[c_1][c_2] \in \hat{\mathbb{Y}}$ : training 2D images ( $H \times W$  matrices of pixels)
          number of external ( $N_{ext}$ ) and internal ( $N_{int}$ ) iterations
output :  $f_{gp}$  : final regression model

1 Function GPFL
2**   $target \leftarrow \hat{\mathbb{Y}}$ 
3     $partialModels \leftarrow [\dots]$ 
4    for  $ext\text{-iter}$  from 1 to  $N_{ext}$  do
5       $\mathcal{P} \leftarrow \text{GET-RANDOM-INITIAL-POPULATION}$ 
6      for  $int\text{-iter}$  from 1 to  $N_{int}$  do
7        for each  $\mathcal{I} \in \mathcal{P}$  do
8**          $\mathcal{I}_{ls} \leftarrow \text{COMPUTE-FITNESS-AND-LS}(target, \mathcal{I})$            // see Algorithm 2
9          $\mathcal{P}' \leftarrow \emptyset$ 
10        add ELITE( $\mathcal{P}$ ) to  $\mathcal{P}'$ 
11        while  $\mathcal{P}'$  is not full do
12           $\mathcal{I}_{ls} \leftarrow \text{TOURNAMENT-SELECTION}(\mathcal{P})$ 
13          add MUTATE( $\mathcal{I}_{ls}$ ) to  $\mathcal{P}'$ 
14         $\mathcal{P} \leftarrow \mathcal{P}'$ 
15       $\mathcal{I}_{ls}^* \leftarrow \text{GET-BEST-INDIVIDUAL}(\mathcal{P})$            // partial model
16      add  $\mathcal{I}_{ls}^*$  to  $partialModels$ 
17      /* the new target is computed as the residual errors of each image */
18**     for each  $i$  from 1 to  $size(target)$  do
19**       for each  $c_1$  from 1 to  $H$  do
20**         for each  $c_2$  from 1 to  $W$  do
21**            $target[i][c_1][c_2] \leftarrow \mathcal{I}_{ls}^*(c_1, c_2) - target[i][c_1][c_2]$ 
22**
23**  return  $f_{gp} \leftarrow \sum_{\mathcal{I}_{ls}^* \in partialModels} \mathcal{I}_{ls}^*$            // linear combination of partial models

```

---

Algorithm 1 describes GPFL’s approach. It has three inputs: (i) the training images ( $\hat{\mathbb{Y}}$ ); (ii) the number of external iterations ( $N_{ext}$ ) (i.e, the number of partial models); and (iii) the number of internal iterations ( $N_{int}$ ) (i.e, the number of generations that GPFL uses to optimize each partial model). GPFL relies on *linear scaling* [17] to construct  $f_{gp}$  as a linear combination of the partial models.

To generate the partial models, GPFL implements the *dynamic-target* approach SGP-DT [41] that evolves multiple models driven by a *target* that changes at each external iteration. SGP-DT initializes the *target* with the training set (line 2, Algorithm 1). At each external iteration (lines 4–20), SGP-DT evolves a population of models ( $\mathcal{P}$ ) to identify one (partial model  $\mathcal{I}_{ls}^*$  line 15) that best approximates the current *target*. SGP-DT evolves  $\mathcal{P}$  using a variant of the classical GP algorithm (lines 6–13) that does not use any form of crossover [41]. Ruberto et al. have shown that such a variant is effective with the dynamic-target approach [41]. At each new external iteration, SGP-DT computes the

new target as the residuals errors of the current target and the best model  $\mathcal{I}_{ls}^*$  (lines 17–20).

Ruberto et al. defined the dynamic-target framework SGP-DT for the numerical symbolic regression domain [41]. We now describe how GPFL adapts it for learning high-level features from 2D images. We mark with \*\* the lines of Algorithm 1 that correspond to the novel aspects of GPFL. First and foremost, GPFL proposes a novel fitness function, which is specific to our problem at hand (Function COMPUTE-FITNESS-AND-LS, lines 22–23, Algorithm 2). Second, GPFL generates the new target by computing the error residuals by differencing images (lines 17–20, Algorithm 1). Third, GPFL constructs the final model using a linear combination. Differently, SGP-DT uses a validation set, which does not apply in our case. We now describe in details these three novel aspects of GPFL.

**Fitness Function.** GPFL invokes Function COMPUTE-FITNESS-AND-LS (Algorithm 2) for each individual in the current population (line 8, Algorithm 1). The function takes in input (i) the current *target*, which are the 2D residual images at the current iteration; and (ii) the individual  $\mathcal{I}$ . Line 33 of Algorithm 2 returns  $\mathcal{I}_{ls}$ , the individual  $\mathcal{I}$  with its fitness score and linear scaling coefficients ( $a$  and  $b$ ). Note that  $a$  and  $b$  are different for each image in *target*. Intuitively, the fitness score captures how well an individual approximates the current target.

---

#### Algorithm 2: GPFL’s fitness function

---

```

input   : target: set of 2D images and  $\mathcal{I}$ : individual
output  :  $\mathcal{I}_{ls}$  encoded with fitness score and linear scaling coefficients

22 Function COMPUTE-FITNESS-AND-LS
23   scores  $\leftarrow [\dots]$  // vector of score for each image in target
24   for each  $\hat{y}$  in target do
25      $y_p \leftarrow [\dots][\dots]$  // predicted image
26     for each  $c_1$  from 1 to  $H$  do
27       for each  $c_2$  from 1 to  $W$  do
28          $y_p[c_1][c_2] \leftarrow \mathcal{I}(c_1, c_2)$ 
29      $\langle a, b \rangle \leftarrow \text{COMPUTE-AND-STORE-LS-COEFFICIENTS}(y_p, \hat{y})$ 
30      $\mathcal{I}_{ls} \leftarrow a + b \cdot \mathcal{I}$  // linear scaling
31     scores[ $\hat{y}$ ]  $\leftarrow \text{COMPUTE-MEAN-SQUARED-ERROR}(\mathcal{I}_{ls}, \hat{y})$ 
32    $\text{fitness-score}(\mathcal{I}_{ls}) \leftarrow \frac{\sum \text{scores}[i]}{\text{size}(\text{scores})}$  // arithmetic mean of the scores
33   return  $\mathcal{I}_{ls}$ 

```

---

The function starts by initializing at empty the vector of scores (line 23 of Algorithm 2), which will populate with the prediction errors of  $\mathcal{I}_{ls}$ , for each image  $\hat{y}$  in *target*. Given an image  $\hat{y}$ , GPFL generates the predicted image  $y_p$  by computing the function  $f_{gp}(c_1, c_2)$  prescribed by the individual  $\mathcal{I}$  for all the  $H \times W$  coordinates  $c_1$  and  $c_2$  in the image  $\hat{y}$ . GPFL assigns the results of  $f_{gp}(c_1, c_2)$  (predicted value) to  $y_p[c_1, c_2]$  (line 28, Algorithm 2). Given  $y_p$  and  $\hat{y}$ , GPFL computes the coefficient  $a$  and  $b$  with the linear scaling technique [17] (line 29, Algorithm 2). Following Keijzer [17], we compute the linear scaling of an individual  $\mathcal{I}$  as follows (where  $\bar{\hat{y}}$  is the arithmetic mean of  $\hat{y}$ )<sup>2</sup>

<sup>2</sup> The cost of computing the linear scaling coefficients is  $\mathcal{O}(|\hat{\mathcal{Y}}| \cdot |\mathcal{P}|)$ .

$$\mathcal{I}_{ls} = a + b \cdot \mathcal{I} \quad (1)$$

$$\text{where } a = \bar{y} - b \cdot \bar{y}_p \quad \text{and} \quad b = \frac{\sum_{i=1}^n [(\hat{y}_i - \bar{y}) \cdot (y_{pi} - \bar{y}_p)]}{\sum_{i=1}^n [(y_{pi} - \bar{y}_p)^2]} \quad (2)$$

Following classical GP approaches, we rely on the Mean Squared Error (MSE) between  $y_p$  and  $\hat{y}$  to compute the scores (line 31, Algorithm 2). Because  $y_p$  and  $\hat{y}$  are images, MSE measures the average squared difference between the predicted value  $y_p[c_1, c_2]$  and the actual value  $\hat{y}[c_1, c_2]$ , for each coordinate  $(c_1, c_2)$ . Being a quadratic function, MSE gives more weight to the pixels with a greater difference. As such, during the first external iterations, GPFL focuses on those elements of the images that lead to greater errors.

After the function analyzes all residual images in *target*, it computes the fitness score of  $\mathcal{I}_{ls}$  as the arithmetic mean of the scores (line 32, Algorithm 2). The rationale of choosing the arithmetic mean is to consider equally important all the images in *target*. The fitness score is a number from zero to infinite. Because it represents an error, the lower the score the fitter the individual.

**Constructing the New Target.** To construct the new target, GPFL scans all the pixel coordinates and computes the difference between the current pixel value and the one predicted by the best model  $\mathcal{I}_{ls}^*$  (lines 17–20, Algorithm 1). As such, the next iteration will focus on the characteristics of the images that the previous iteration did not approximate well. Note that the linear scaling coefficients are different for every image and were previously computed by the Function COMPUTE-FITNESS-AND-LS.

**Constructing the Final Model.** GPFL constructs the final model with a linear combination of all the partial models (line 21, Algorithm 1). Intuitively, by combining all partial models we are summing all the estimates of the residuals, and thus obtaining a function  $f_{gp}$  that well approximates the training images in input. Notably,  $f_{gp}$  is a parametric function with  $a$  and  $b$  as parameters.

## 4 Experiments

This section describes a series of experiments to evaluate how well the models trained by GPFL capture the most relevant high-level features of 2D images. Because given enough external iterations GPFL can achieve an arbitrary lower reconstruction error, we opted to evaluate GPFL with classification accuracy instead. In fact, linear scaling guarantees that the reconstruction error (i.e., RMSE) monotonically decreases [17]. This happens because GPFL re-computes the linear scaling coefficients when reconstructing each test image.

We built a naïve classifier that relies on GPFL for classifying MNIST digits (Algorithm 3), and compared with the DNN L<sub>EN</sub>ET5 [19, 21]<sup>3</sup>. We experimented

<sup>3</sup> When comparing the classification accuracy of GPFL and L<sub>EN</sub>ET5, we computed the p-values with the non-parametric pairwise *Wilcoxon rank-sum test* [15].

with smaller MNIST training sets and with noisy MNIST test sets to evaluate the generalizability and robustness of the models, respectively. This is a common experimental setup [13] for the *few-shot learning problem* [12].

**Dataset.** The MNIST database of handwritten digits [48] comprises a training set of 60,000 examples, and a test set of 10,000 examples. Each example is a grayscale numerical bitmap image of  $28 \times 28$  pixels representing a handwritten digit from 0 to 9. MNIST is widely-used as a standard benchmark in the ML community [6, 47]. Even now, MNIST is often the first dataset that researchers use to validate their algorithms [2–5, 8, 11]. From the MNIST training set of 60,000 images, we constructed three variants of smaller size, with five (MNIST – 5), ten (MNIST – 10), and one hundred (MNIST – 100) images for each digit. Because there are ten digits (0 to 9), the three variants contain 50, 100 and 1,000 examples, respectively. We constructed such variants by randomly sampling the MNIST training set. To avoid selection biases, we repeated the sampling process 30 times for each of the three variants, obtaining 90 datasets in total. We stored them on disk to train GPFL and LENET5 with exactly the same datasets.

---

**Algorithm 3: GPFL-based MNIST naïve classifier**

---

```

input  :  $\hat{Y}$  MNIST training set,  $S$  ensembles size
output : ensembles for each digit

1 Function TRAINER
2   for each digit from 0 to 9 do
3     for each  $i$  from 1 to  $S$  do
4        $ensembles[digit][i] \leftarrow \text{GPFL}(\hat{Y}[digit], N_{ext}=100, N_{int}=50)$ 
5   return ensembles

input  : ensembles for each digit returned by the trainer,  $\hat{y}$  image to classify
output : predicted digit of  $\hat{y}$ 

6 Function PREDICTOR
7   for each digit from 0 to 9 do
8      $\hat{y}_{rc} \leftarrow [\dots][\dots]$  // reconstructed image initialized at empty
9     for each  $i$  from 1 to  $S$  do
10       $ensembles[digit][i]_{ls} \leftarrow \text{COMPUTE-LS}(ensembles[digit][i], \hat{y})$ 
11       $\hat{y}_{rc} \leftarrow \hat{y}_{rc} + reconstruction(ensembles[digit][i]_{ls})$ 
12       $average-image \leftarrow \hat{y}_{rc}/N$  // average of each pixel
13       $error[digit] \leftarrow \text{MSE}(\hat{y}, average-image)$  // mean square error
14   return  $digit \leftarrow \underset{digit \in \{0 \dots 9\}}{\text{argmin}} \{error[digit]\}$ 

```

---

**A GPFL-Based Classifier.** To evaluate how well GPFL learns relevant high-level features that characterize the images in input, we constructed a *naïve classifier* (Algorithm 3) that classifies unseen MNSIT digits relying on the models that GPFL produces. The classifier splits the training images  $\hat{Y}$  into ten partitions ( $\hat{Y}[digit]$ ) according to their digit label. Then, it uses GPFL to train multiple models (called ensemble) for each of the partitions. We use the *ensemble method* to mitigate the stochasticity of GP, which can lead to models of arbitrary performance. Note that, although GPFL is unsupervised, the classifier is supervised because it splits the training set according to the digit labels.

The naïve classifier has two components: the TRAINER and the PREDICTOR (see Algorithm 3). The predictor takes in input the test image to classify and the list of ensembles precomputed by the trainer. To predict the digit of the test image, the predictor reconstructs the image multiple times, one for each ensemble. Internally, the predictor obtains each reconstructed image by averaging the pixels outputted by the models. Then, it returns the digit corresponding to the ensemble that yielded the lowest reconstruction error. When reconstructing the test image  $\hat{y}$ , we recompute the linear scaling parameters that best approximate  $\hat{y}$ .

We decided to rely on a naïve classifier (as opposed to more sophisticated approaches, e.g., SVM and Random forest) to isolate our main contribution. So that the effectiveness of the classification (Algorithm 3) can be mainly attributed to the quality of the high-level features that GPFL identified.

**GPFL Configuration.** Following the dynamic target framework SGP-DT [41], we configured GPFL as follows. Fifty internal iterations ( $N_{\text{int}}$ ). One hundred the population size. The *ramped-half-and-half* initialization generates trees with a depth that ranges from 1 to 4 (line 5, Algorithm 1). The probability of mutation is 100%, and the maximum depth of the subtrees generated by the mutation operators is five. The probability of a sub-tree mutation happening at the leaf level is 70%. We set no limits on the number of nodes and on the depth of the trees. We set size of the tournament selection to two, and the elitism size to one.

We ran GPFL with different combinations of  $N_{\text{ext}}$  (number of external iterations) and  $S$  (the ensembles size) and choose the best ones. Table 1 gives the median accuracy of GPFL on the 30 datasets of each variant, using the original test set of 10,000 images. The different combinations of  $N_{\text{ext}}$  and  $S$  give similar accuracy results, except for the combination  $N_{\text{ext}} = 40$  and  $S = 1$ , which has the lowest performance. This confirms the importance of an ensembles approach. Table 1 marks in bold the highest median accuracy for each of the datasets. In our experiments, we will use the corresponding values of  $N_{\text{ext}}$  and  $S$ .

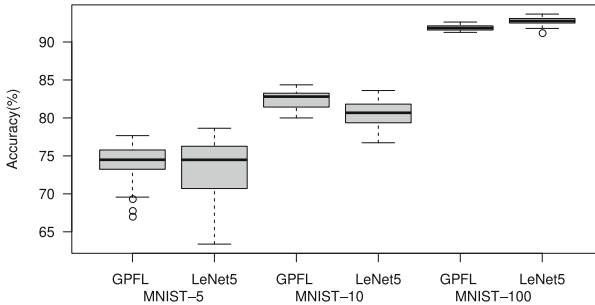
**Table 1.** Classification accuracy of GPFL and L<sub>EN</sub>ET5 on 10,000 test images

# external iter. ( $N_{\text{ext}}$ )	Ensembles size ( $S$ )	GPFL median accuracy %			Learning rate	L <sub>EN</sub> ET5 median accuracy %		
		MNIST-5	MNIST-10	MNIST-100		MNIST-5	MNIST-10	MNIST-100
20	50	73.78%	81.21%	90.16%	0.1	<b>74.48%</b>	<b>80.67%</b>	92.67%
30	50	74.18%	82.53%	91.26%	0.01	73.88%	80.58%	<b>92.76%</b>
40	50	<b>74.49%</b>	<b>82.81%</b>	91.66%	0.001	73.52%	79.92%	92.28%
60	50	73.82%	82.70%	<b>91.84%</b>	0.0001	73.90%	79.48%	91.49%
100	50	73.27%	82.19%	91.26%				
40	1	<b>68.36%</b>	<b>76.32%</b>	85.07%				
40	30	74.33%	82.63%	91.59%				

**L<sub>EN</sub>ET5 Configuration.** We compared GPFL with the Convolutional Neural Network (CNN) L<sub>EN</sub>ET5 [19,21], the most used baseline for MNIST [13,14].

We implemented L<sub>ENET</sub>5 with the T<sub>ENSORFLOW</sub> framework. To be sure that our implementation was correct, we confirmed that when trained with all the 60,000 training images and validated with the 10,000 test images, our implementation achieves the advertised classification accuracy of 99% [19,21]. We released our datasets, models and implementation and we welcome external validation [39].

A key hyper-parameter of CNNs is the *learning rate* that controls how much the weights are adjusted with respect to the loss gradient [20]. The lower the value, the slower the training. The original L<sub>ENET</sub>5 uses 0.1 as learning rate, which might be inadequate in our case since we have smaller training sets. Table 1 shows the median accuracy of L<sub>ENET</sub>5 on our datasets with different learning rates (0.1, 0.01, 0.001, and 0.0001), using the original test set of 10,000 images. Table 1 marks in bold the highest median accuracy for each of the three datasets.



**Fig. 1.** GPFL and L<sub>ENET</sub>5 classification accuracy (best configurations)

**Classification Accuracy on 10,000 Test Images.** Figure 1 shows the box-plot of the classification accuracy distributions of GPFL and L<sub>ENET</sub>5 with their best configurations (see Table 1). On MNIST – 5, GPFL and L<sub>ENET</sub>5 achieve a similar median accuracy of  $\sim 74.50\%$ , but GPFL exhibits less variance. This is the only non-statistically significant result (p-value 0.597). On MNIST – 10, GPFL outperforms L<sub>ENET</sub>5 (p-value  $5.14 \cdot 10^{-6}$ ) with a median accuracy of 82.81% and 80.67%, respectively. On MNIST – 100, GPFL underperforms L<sub>ENET</sub>5 (p-value  $1.17 \cdot 10^{-5}$ ) with a median accuracy of 91.84% and 92.76%, respectively.

As expected, with the increasing of the training size, both techniques attain better classification accuracy. Moreover, because smaller datasets might lack representative training cases, the variance increases when the training size decreases. Despite that the architecture of L<sub>ENET</sub>5 was specifically designed for the MNIST dataset [19,21], GPFL’s results are comparable with L<sub>ENET</sub>5 on MNIST – 5, and better than L<sub>ENET</sub>5 on MNIST – 10. This demonstrates that, given small training sets, GPFL learns the relevant high-level features of MNIST images. This is an important result considering that GPFL’s architecture is agnostic to MNIST.



**Classification Accuracy on 10,000 Noisy Test Images.** We added random noise to the MNIST test set of 10,000 images to compare the noise resilience of GPFL and L<sub>ENET</sub>5. We considered five levels of salt noise  $L$ : 5%, 10%, 20%, 30%, 40%.  $L$  specifies the percentage of randomly selected pixels in each image whose values turn into 255 (white color). We decided to use salt noise because (in our case) is more disruptive than the salt-and-pepper noise. In fact, the majority of pixels in MNIST images are black (background color). The left matrix in Fig. 3 exemplifies the five noise levels (Columns 5%, 10%, 20%, 30%, 40%).

Table 2 shows the median accuracy for each combination of training sets (MNIST – 5, MNIST – 10 and MNIST – 100) and noise levels (5%, 10%, 20%, 30%, 40%). GPFL always outperforms L<sub>ENET</sub>5 for every combination of training sets and noise levels. The comparison is always statistical significant (p-values  $< 10^{-6}$ ), except when comparing GPFL and L<sub>ENET</sub>5 trained on MNIST – 5 and tested with noise level 5% (p-value 0.121). For noise level 5%, the difference between the median accuracy of GPFL and L<sub>ENET</sub>5 ranges from +1.11% to +2.84%. With the increasing of the noise level, the difference constantly grows. For noise level 40%, the difference ranges from +30.51% to +40.85%.

**Table 2.** Median accuracy with 10,000 noisy test images

Salt noise level %	MNIST-5		MNIST-10		MNIST-100	
	GPFL	L <sub>ENET</sub> 5	GPFL	L <sub>ENET</sub> 5	GPFL	L <sub>ENET</sub> 5
5	73.54%	72.43%	82.01%	79.18%	91.01%	89.75%
10	72.79%	68.95%	80.99%	75.56%	89.55%	80.49%
20	69.47%	56.78%	78.02%	63.45%	84.98%	57.37%
30	65.31%	41.10%	73.00%	49.26%	76.42%	37.72%
40	59.18%	28.67%	65.73%	35.99%	65.49%	24.64%

Figure 2 plots the median accuracy trend at the noise level increases. For all the three MNIST variants, L<sub>ENET</sub>5 accuracy degrades much faster than the one of GPFL. Interestingly, GPFL always outperforms L<sub>ENET</sub>5.

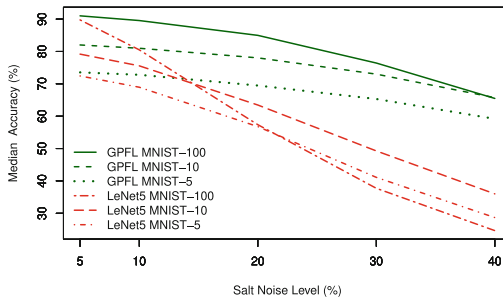
For the lowest noise level (5%), L<sub>ENET</sub>5 trained on MNIST – 100 (denoted by L<sub>ENET</sub>5<sub>100</sub>) outperforms L<sub>ENET</sub>5<sub>5</sub> and L<sub>ENET</sub>5<sub>10</sub>. This is an expected result, because (in principle) the larger the training set the highest the classification accuracy. However, the performance of L<sub>ENET</sub>5<sub>100</sub> drastically decreases when the noise level increases. For noise level 30% and 40%, L<sub>ENET</sub>5<sub>100</sub> performs significantly worse than L<sub>ENET</sub>5<sub>5</sub> and L<sub>ENET</sub>5<sub>10</sub>. This result can be due to L<sub>ENET</sub>5<sub>100</sub> has “*overfitted the clean data*”: MNIST – 100 has the largest size and it requires more epochs to converge. As such, when the noise level increases, the difference between the training and the test sets also increases. Intuitively, the higher this difference, the lower the classification accuracy. Tsipras et al. had similar conclusions when testing recent DNNs with noisy MNIST test sets [44].

The classification accuracy of GPFL<sub>100</sub> degrades at the increasing of the noise, but at a much slower pace. Only at noise level 40%, GPFL<sub>100</sub> achieves a similar classification accuracy with GPFL<sub>10</sub>. Analogously to L<sub>ENET</sub>5, GPFL<sub>100</sub> has “*overfitted the clean data*”: MNIST – 100 has the largest size and the highest number of external iterations ( $N_{\text{ext}} = 60$  vs 40 see Table 1).

**Reconstruction Results.** The right matrix in Fig. 3 shows ten images from the MNIST test set and their GPFL’s reconstructions at various numbers of external iterations (i.e., partial models). These are the results of GPFL trained on MNIST – 100 with ensembles size  $S = 50$ . Column  $N_{\text{ext}}$  shows the images that GPFL reconstructs using the linear combination of the first  $N_{\text{ext}}$  partial models, that is  $f_{gp}(\text{original}) = \sum_{i=1}^{N_{\text{ext}}} \text{partialModels}[i]$ . With a low value of  $N_{\text{ext}}$ , the reconstructed images focus on the macro characteristics of the images. For instance, the images of Column  $N_{\text{ext}} = 2$  show clouds of dust that resemble the shape of the digits. When  $N_{\text{ext}}$  increases, the finer details gradually appear because GPFL focuses on the residual errors of previous iterations. As Fig. 3 exemplifies, the process looks like a progressive cleansing of the images.

The left matrix of Fig. 3 shows ten images of the MNIST test set, their noisy versions ( $L\%$ ) and their reconstructions ( $R$ ) using GPFL<sub>100</sub>. The reconstructed digits are recognizable even at noise level 40%. However, the reconstructions of the digits two and zero show some artifacts originated by an uneven distribution of the noise that GPFL interpreted as high-level features.

**Size of the Final Solutions.** The average size of  $f_{gp}$  with  $N_{\text{int}} = 50$  and  $N_{\text{ext}} = 40$  is 4,587 ( $\pm 4.6\%$ ), which is the number of nodes in the tree-like representation of  $f_{gp}$ . Recall that GPFL constructs  $f_{gp}$  by assembling the partial models with a linear combination. As such, after 50 internal iterations the resulting partial models have an average size of 115 nodes (i.e.,  $4,587/40 = 115$ ).



**Fig. 2.** Median classification accuracy degradation at the increasing of noise level.

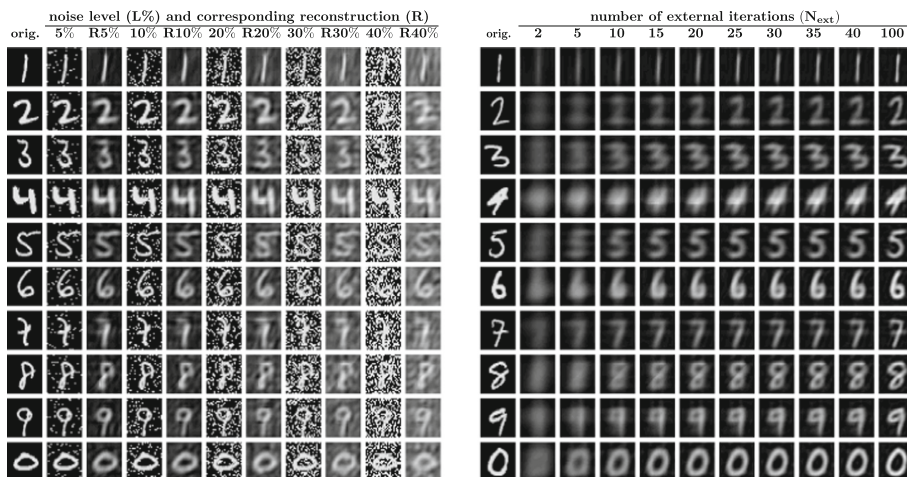


Fig. 3. Examples of reconstructed images of GPFL (with noise on the left).

## 5 Conclusion

This paper presented GPFL, a GP technique to learn high-level features from 2D images. Differently from previous GP feature learner attempts, GPFL does not simulate the behavior of Deep Neural Networks (DNNs) whatsoever. Our novel GP approach can handle more complex classification tasks than previous attempts. Our experiments with MNIST show that GPFL has a competitive edge with L<sub>ENET</sub>5 when considering small training sets and noisy test sets.

*Note that, we are not claiming that GPFL is a valid alternative to DNNs for learning high-level features from 2D images.* In fact, MNIST is the (simple) de-facto standard benchmark for a first sanity check only. Moreover, we compared GPFL with L<sub>ENET</sub>5 that (although being specific to MNIST) is not the most recent DNN-based feature learner. However, GPFL demonstrates that a GP feature learner can lead to interesting results that are worth investigating further.

## References

1. Albukhanajer, W.A., Briffa, J.A., Jin, Y.: Evolutionary multiobjective image feature extraction in the presence of noise. *IEEE Trans. Cybern.* **45**(9), 1757–1768 (2015). <https://doi.org/10.1109/TCYB.2014.2360074>
2. Alvear-Sandoval, R.F., Sancho-Gómez, J.L., Figueiras-Vidal, A.R.: On improving CNNs performance: the case of MNIST. *Inf. Fusion* **52**, 106–109 (2019)
3. Baldominos, A., Saez, Y., Isasi, P.: Evolutionary convolutional neural networks: an application to handwriting recognition. *Neurocomputing* **283**, 38–52 (2018). <https://doi.org/10.1016/j.neucom.2017.12.049>

4. Baldominos, A., Saez, Y., Isasi, P.: Model selection in committees of evolved convolutional neural networks using genetic algorithms. In: Intelligent Data Engineering and Automated Learning, IDEAL 2018, pp. 364–373 (2018). [https://doi.org/10.1007/978-3-030-03493-1\\_39](https://doi.org/10.1007/978-3-030-03493-1_39)
5. Baldominos, A., Saez, Y., Isasi, P.: Hybridizing evolutionary computation and deep neural networks: an approach to handwriting recognition using committees and transfer learning. *Complexity* (2019). <https://doi.org/10.1155/2019/2952304>
6. Baldominos, A., Saez, Y., Isasi, P.: A survey of handwritten character recognition with MNIST and EMNIST. *Appl. Sci.* **9**(15), 3169 (2019)
7. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). [https://doi.org/10.1007/11744023\\_32](https://doi.org/10.1007/11744023_32)
8. Bochinski, E., Senst, T., Sikora, T.: Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. In: Proceedings International Conference on Image Processing, ICIP 2017, pp. 3924–3928 (2017). <https://doi.org/10.1109/ICIP.2017.8297018>
9. Brameier, M.F., Banzhaf, W.: Linear Genetic Programming. Springer, Boston (2007). [https://doi.org/10.1007/978-0-387-31030-5\\_1](https://doi.org/10.1007/978-0-387-31030-5_1)
10. Butterworth, J., Savani, R., Tuyls, K.: Evolving indoor navigational strategies using gated recurrent units in NEAT. In: Proceedings of the Companion of Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 111–112 (2019). <https://doi.org/10.1145/3319619.3321995>
11. Davison, J.: DEvol: automated deep neural network design via genetic programming (2020). <https://github.com/joeddav/devol>
12. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(4), 594–611 (2006)
13. George, D., et al.: A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science* **358**(6368), 2612 (2017)
14. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems, pp. 1135–1143 (2015)
15. Haynes, W.: Wilcoxon rank sum test. In: Encyclopedia of Systems Biology, pp. 2354–2355 (2013)
16. Impedovo, S., Mangini, F.: A novel technique for handwritten digit classification using genetic clustering. In: Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR 2012, pp. 236–240 (2012). <https://doi.org/10.1109/ICFHR.2012.167>
17. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36599-0\\_7](https://doi.org/10.1007/3-540-36599-0_7)
18. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.* **37**(2), 233–243 (1991)
19. LeCun, Y.: Lenet-5, convolutional neural networks (2020). <http://yann.lecun.com/exdb/lenet>
20. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
21. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition, vol. 86, pp. 2278–2324. IEEE (1998)
22. Legge, G.E., Foley, J.M.: Contrast masking in human vision. *Josa* **70**(12), 1458–1471 (1980)

23. Lensen, A., Al-Sahaf, H., Zhang, M., Xue, B.: Genetic programming for region detection, feature extraction, feature construction and classification in image data. In: Heywood, M.I., McDermott, J., Castelli, M., Costa, E., Sim, K. (eds.) EuroGP 2016. LNCS, vol. 9594, pp. 51–67. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-30668-1\\_4](https://doi.org/10.1007/978-3-319-30668-1_4)
24. Lensen, A., Xue, B., Zhang, M.: Can genetic programming do manifold learning too? In: Sekanina, L., Hu, T., Lourenço, N., Richter, H., García-Sánchez, P. (eds.) EuroGP 2019. LNCS, vol. 11451, pp. 114–130. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-16670-0\\_8](https://doi.org/10.1007/978-3-030-16670-0_8)
25. Lensen, A., Zhang, M., Xue, B.: Multi-objective genetic programming for manifold learning: balancing quality and dimensionality. *Genet. Program Evolvable Mach.* **21**(3), 399–431 (2020). <https://doi.org/10.1007/s10710-020-09375-4>
26. Liu, L., Shao, L., Li, X.: Evolutionary compact embedding for large-scale image classification. *Inf. Sci.* **316**, 567–581 (2015). <https://doi.org/10.1016/j.ins.2014.06.030>
27. López, U., Trujillo, L., Martínez, Y., Legrand, P., Naredo, E., Silva, S.: RANSAC-GP: dealing with outliers in symbolic regression with genetic programming. In: Proceedings of the European Conference on Genetic Programming, EuroGP 2017, pp. 114–130 (2017)
28. Makhzani, A., Frey, B.J.: Winner-take-all autoencoders. In: *Advances in Neural Information Processing Systems*, pp. 2791–2799 (2015)
29. McDermott, J.: Why is auto-encoding difficult for genetic programming? In: Sekanina, L., Hu, T., Lourenço, N., Richter, H., García-Sánchez, P. (eds.) EuroGP 2019. LNCS, vol. 11451, pp. 131–145. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-16670-0\\_9](https://doi.org/10.1007/978-3-030-16670-0_9)
30. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.: Reading digits in natural images with unsupervised feature learning. Google technical report (2011)
31. Neumann, L., Matas, J.: Real-time scene text localization and recognition. In: Proceedings of Conference on Computer Vision and Pattern Recognition, CVPR 2012, pp. 3538–3545 (2012)
32. Oliva, A., Torralba, A.: Building the gist of a scene: the role of global image features in recognition. *Prog. Brain Res.* **155**, 23–36 (2006)
33. Orzechowski, P., La Cava, W., Moore, J.H.: Where are we now?: a large benchmark study of recent symbolic regression methods. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 1183–1190 (2018). <https://doi.org/10.1145/3205455.3205539>
34. Papavasileiou, E., Jansen, B.: An investigation of topological choices in FS-NEAT and FD-NEAT on XOR-based problems of increased complexity. In: Proceedings of the Companion of Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 1431–1434 (2017). <https://doi.org/10.1145/3067695.3082497>
35. Peng, Y., Chen, G., Singh, H., Zhang, M.: NEAT for large-scale reinforcement learning through evolutionary feature learning and policy gradient search. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 490–497 (2018). <https://doi.org/10.1145/3205455.3205536>
36. Perez, C.B., Olague, G.: Genetic programming as strategy for learning image descriptor operators. *Intell. Data Anal.* **17**(4), 561–583 (2013). <https://doi.org/10.3233/IDA-130594>

37. Rodriguez-Coayahuitl, L., Morales-Reyes, A., Escalante, H.J.: Structurally layered representation learning: towards deep learning through genetic programming. In: Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S., García-Sánchez, P. (eds.) EuroGP 2018. LNCS, vol. 10781, pp. 271–288. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77553-1\\_17](https://doi.org/10.1007/978-3-319-77553-1_17)
38. Rodriguez-Coayahuitl, L., Morales-Reyes, A., Escalante, H.J.: Evolving autoencoding structures through genetic programming. *Genet. Program Evolvable Mach.* **20**(3), 413–440 (2019). <https://doi.org/10.1007/s10710-019-09354-4>
39. Ruberto, S., Terragni, V., Moore, J.H.: GPFL replication package. experimental data of GPFL and source code of Lenet5, April 2020. <https://doi.org/10.5281/zenodo.3899891>
40. Ruberto, S., Terragni, V., Moore, J.H.: Image feature learning with a genetic programming autoencoder. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2020 (2020)
41. Ruberto, S., Terragni, V., Moore, J.H.: SGP-DT: semantic genetic programming based on dynamic targets. In: Hu, T., Lourenço, N., Medvet, E., Divina, F. (eds.) EuroGP 2020. LNCS, vol. 12101, pp. 167–183. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-44094-7\\_11](https://doi.org/10.1007/978-3-030-44094-7_11)
42. Ruberto, S., Terragni, V., Moore, J.H.: SGP-DT: towards effective symbolic regression with a semantic GP approach based on dynamic targets. In: Proceedings of the Genetic and Evolutionary Computation Conference (Hot Off the Press track), GECCO 2020 (2020)
43. Ruberto, S., Vanneschi, L., Castelli, M.: Genetic programming with semantic equivalence classes. *Swarm Evol. Comput.* **44**, 453–469 (2019). <https://doi.org/10.1016/j.swevo.2018.06.001>
44. Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., Madry, A.: Robustness may be at odds with accuracy (2018)
45. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the International Conference on Machine Learning, ICML 2008, pp. 1096–1103 (2008)
46. Wang, J., Zhang, Z., Zha, H.: Adaptive manifold learning. In: Advances in Neural Information Processing Systems, NIPS 2005 (2005)
47. Yadav, C., Bottou, L.: Cold case: the lost MNIST digits. In: Advances in Neural Information Processing Systems, NIPS 2019, pp. 13443–13452 (2019)
48. Yann LeCun, C.C., Burges, C.: MNIST handwritten digit database (2020)
49. Zheng, A., Casari, A.: Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. O’Reilly Media Inc., Sebastopol (2018)
50. Zhou, H., Yuan, Y., Shi, C.: Object tracking using sift features and mean shift. *Comput. Vis. Image Underst.* **113**(3), 345–352 (2009)



# Learning a Formula of Interpretability to Learn Interpretable Formulas

Marco Virgolin<sup>1</sup>(✉), Andrea De Lorenzo<sup>2</sup>, Eric Medvet<sup>2</sup>,  
and Francesca Randone<sup>3</sup>

<sup>1</sup> Centrum Wiskunde & Informatica, Amsterdam, Netherlands  
marco.virgolin@cwi.nl

<sup>2</sup> Department of Engineering and Architecture, University of Trieste, Trieste, Italy

<sup>3</sup> IMT School for Advanced Studies Lucca, Lucca, Italy

**Abstract.** Many risk-sensitive applications require Machine Learning (ML) models to be interpretable. Attempts to obtain interpretable models typically rely on tuning, by trial-and-error, hyper-parameters of model complexity that are only loosely related to interpretability. We show that it is instead possible to take a meta-learning approach: an ML model of non-trivial Proxies of Human Interpretability (PHIs) can be learned from human feedback, then this model can be incorporated within an ML training process to directly optimize for interpretability. We show this for evolutionary symbolic regression. We first design and distribute a survey finalized at finding a link between features of mathematical formulas and two established PHIs, *simulatability* and *decomposability*. Next, we use the resulting dataset to learn an ML model of interpretability. Lastly, we query this model to estimate the interpretability of evolving solutions within bi-objective genetic programming. We perform experiments on five synthetic and eight real-world symbolic regression problems, comparing to the traditional use of solution size minimization. The results show that the use of our model leads to formulas that are, for a same level of accuracy-interpretability trade-off, either significantly more or equally accurate. Moreover, the formulas are also arguably more interpretable. Given the very positive results, we believe that our approach represents an important stepping stone for the design of next-generation interpretable (evolutionary) ML algorithms.

**Keywords:** Explainable artificial intelligence · Interpretable machine learning · Symbolic regression · Genetic programming · Multi-objective

## 1 Introduction

Artificial Intelligence (AI), especially when intended as Machine Learning (ML), is increasingly pervading society. Although ML brings numerous advantages, it is far from being fault-prone, hence its use comes with risks [1, 16, 24, 36]. In many cases, failures with serious consequences could have been foreseen and prevented,



if the ML models had not been unintelligible, i.e., *black-boxes*. Nowadays, especially for high-stakes applications, practitioners, researchers, and policy makers increasingly ask for ML to be used responsibly, fairly, and ethically [8, 15]. Therefore, decisions taken by ML models need to be explainable [1, 2].

The field of eXplainable AI (XAI) studies techniques to provide explanations for the decisions taken by black-box models (or, more generally, AI systems), metrics that can be used as *Proxies of Human Interpretability* (PHIs), as well as ML algorithms meant for the synthesis of models that are immediately interpretable [1, 36]. In this paper, we consider the latter case.

Several ML algorithms and techniques exist that are considered *capable* of synthesizing interpretable models. Among these, fitting linear models (e.g., by ordinary least squares or elastic net [53]), building decision trees [4], and evolutionary program synthesis [30] are often listed in surveys on XAI (see, e.g., [1, 16]). Unfortunately, in general, it cannot be *guaranteed* that the model obtained by an ML algorithm will turn out to be interpretable. For example, when a decision tree is built, the more the tree grows deep, the less the chances of the tree being interpretable. Therefore, what is normally done is to repeat the ML training process (decision tree construction) with different hyper-parameter settings (tree depth) in a trial-and-error fashion, until a satisfactory model is obtained. Trial-and-error, of course, comes with time costs. Next to this, another important issue is the fact that hyper-parameters are mostly meant to control the bias-variance interplay [3], and are but loosely related to interpretability.

Multi-Objective Genetic Programming (MOGP) is a very interesting approach because, by its very nature, it mitigates the need for trial-and-error [30, 52]. By evolving a population of solutions that encode ML models, MOGP can synthesize, in a single run, a plethora of models with trade-offs between accuracy and a chosen PHI. Obtaining multiple models at once enhances the chance that a model with a satisfying trade-off between accuracy and interpretability will be found quickly. Nonetheless, the problem of finding a suitable PHI remains. So far, the PHI that have been used were quite simplistic. For example, a common approach is to simply minimize the total number of model components (see Sect. 2 for more). In this paper, we propose a way to improve upon the use of simplistic PHIs, and we focus on the case of MOGP for symbolic regression, i.e., where models are sought that can be written as mathematical formulas.

Our proposal is composed of three main parts. We begin by showing how it is possible to learn a model of non-trivial PHIs. This can be seen as a concretization of an idea that was sketched in [11]: a data-driven approach can be taken to discover what features make ML models more or less interpretable. In detail, (1) we design a survey about mathematical formulas, to gather human feedback on formula interpretability according to two established PHIs: *simulatability* and *decomposability* [24] (see Sect. 3.1); (2) we process the survey answers and condense them into to a regression dataset that enables us to discover a non-trivial model of interpretability; (3) we incorporate the so-found model within an MOGP algorithm, to act as an estimator for the second objective (the first being



the mean squared error): in particular, the model takes as input the features of a formula, and outputs an estimate of interpretability.

All of our code, including the data obtained from the survey, is available at: <https://github.com/MaLeLabTs/GPFormulasInterpretability>.

## 2 Related Work

In this paper, we focus on using ML to obtain interpretable ML models, particularly in the form of formulas and by means of (MO)GP. We do not delve into XAI works where explanations are sought for the decisions made by a black-box model (see, e.g., [34, 48]), nor where the black-box model needs to be approximated by an interpretable surrogate (e.g., a recent GP-based work on this is [14]). We refer to [1, 16] as excellent surveys on various aspects of XAI. We describe the PHIs we adopt, and briefly mention works adopting them, in Sect. 3.1.

Regarding GP for the synthesis of ML models, a large amount of literature is focused on controlling complexity, but not primarily as a means to enable interpretability. Rather, the focus is on overfitting prevention, oftentimes (but not exclusively) by limiting bloat, i.e., the excessive growth of solution size [7, 31, 33, 37, 38, 42, 51]. Among these works, [13, 39, 46] share with us the use MOGP, but are different in that they use hand-crafted complexity metrics instead of taking a data-driven approach (respectively solution size, order of non-linearity, and a modification of solution size), and again these metrics are designed to control bloat and overfitting instead of enable interpretability ([46] does however discuss some effects on interpretability).

Among the works that use GP and focus on interpretability, [6] considers the evolution of rule-based classifiers, and evaluates them using a PHI that consists of dividing the number of conditions in the classifier by the number of classes. In [18], GP is used to evolve reinforcement learning policies as symbolic expressions, and complexity in interpretation is measured by accounting for variables, constants, and operations, with different ad-hoc weights. The authors of [43] study whether modern model-based GP can be useful when particularly compact symbolic regression solutions are sought, to allow interpretability. A very different take to enable or improve interpretability is taken in [22, 41, 45], where interpretability is sought by means of feature construction and dimensionality reduction. In [22] in particular, MOGP is used, with solution size as a simple PHI. Importantly, none of these works takes attempts to learn a PHI from data.

Perhaps the most similar work to ours is [27]. Like we do, the authors train an ML model (a deep residual network [17]) from pre-collected human-feedback to drive an evolutionary process, but for a very different aim, i.e., automatic art synthesis (the human-feedback is aesthetic rankings for images).

## 3 The Survey

We prepared an online survey (<http://mathquiz.inginf.units.it>) to assess the simulatability and decomposability of mathematical formulas (we referred to [5] for

<p>Given the formula <math>\frac{5x_1+1}{\cos(x_2-3.14)}</math> and the input value(s) <math>[x_1 = 8.0, x_2 = 6.28]</math>, which option is closest to the output?</p> <p>(a) -410.0 (b) -41.0 (c) 410.0 (d) -20.5</p>	<p>Consider the formula <math>5 \sin^{0.5}(x - 3.14)</math>. Which option best describes the behavior of the function as <math>x</math> varies in <math>[-1.0, 1.0]</math>?</p> <p>(a) The function is bounded but not always defined (b) The function is not bounded nor always defined (c) The function is not bounded but always defined (d) The function is bounded and always defined</p>
---	--

**Fig. 1.** Examples of questions on simulatability (left) and decomposability (right).

survey-preparation guidelines). We begin by describing the two PHIs, and proceed with an overview of the content of the survey and the generation process. We provide full details on online supplementary material at: <https://github.com/MaLeLabTs/GPFormulasInterpretability>.

### 3.1 Simulatability and Decomposability

Simulatability and decomposability are two established PHIs, introduced in a seminal work on XAI [24]. Simulatability represents a measurable proxy for the capability of a person to contemplate an entire ML model, and is measured by assessing whether a human, given some input data, can reproduce the model’s output within a reasonable error margin and time [24]. No specific protocol exists to perform the measurement. In [32], this PHI was measured as the absolute deviation between the human estimate for the output of a (linear) model and the actual output, given a set of inputs. With our survey, we measured the rate of correct answers to multiple choices questions on the output of a formula.

Decomposability represents the possibility that a model can be interpreted by parts: inputs, parameters, and (partial) calculations of a model need to admit an intuitive explanation [24]. For example, the coefficients of a linear model can be interpreted as the strengths of association between features and output. Decomposability is similar to the concept of *intelligibility* of [25]. As for simulatability, there exists no prescription on how to measure decomposability. We considered variables as important components to represent this PHI, and gathered answers (correct/wrong) on properties of the behavior of a formula when one of its variables varies within an interval.

### 3.2 Overview on the Survey and Results

We implemented the survey as a webpage, consisting of an introductory section to assess the respondents’ level of familiarity with formulas, followed by eight questions, four about simulatability, and four about decomposability. The eight questions are randomly selected when the webpage is loaded, from a pre-generated database that contains 1000 simulatability and 1000 decomposability questions, each linked to one of 1000 automatically generated formulas. Figure 1

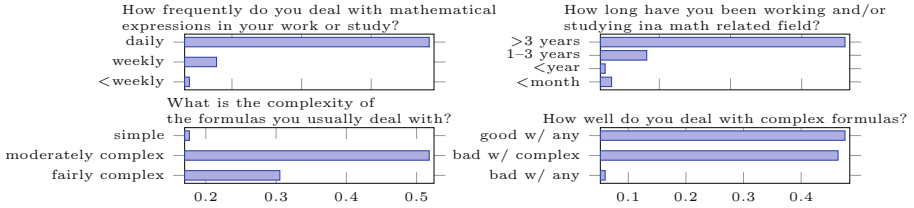


Fig. 2. Distribution of answers about user expertise.

shows examples of these questions. Each and every question presents four possible answers, out of which only one is correct. Alongside each question, the user is asked to state how confident (s)he is about the answer, on a scale from 1 to 4.

The 1000 formulas were encoded with trees, and randomly generated with a half-and-half initialization of GP [30] (max depth 4). The set of leaf nodes for the trees included 4 different variables, and constants that were either integers (from 0 to 10) or multiples of  $\pi$  (3.14 or 6.28). The possible operations were  $+$ ,  $-$ ,  $\times$ ,  $\div$ ,  $\wedge$ ,  $\sqrt{\cdot}$ ,  $\sin$ , and  $\cos$ . We performed rejection sampling and automatic simplifications to avoid presenting fundamentally uninteresting functions (e.g., constant ones), or functions with exploding output (e.g., due to  $\wedge$ ).

For simulatability questions, the user was either asked to pick the correct 2D graph representing the behavior of the (one variable) formula, or to choose the best estimate of the output of the (multi-variable) formula, given values for the variables. Decomposability questions asked whether the formula was (not) bounded, (not) always defined, (not) null in some points, (not) negative in some points, for one variable changing in a given interval and the others being fixed.

We distributed the survey by emailing research groups and departments within the institutes of the authors, targeting both students and faculty members. We further shared the survey on Reddit (subreddit CasualMath) and Twitter. We obtained 334 responses in  $\approx 35$  days, corresponding to 2672 answers. Figure 2 shows the distribution of answers to the introductory part of the survey.

## 4 Learning a Formula of Interpretability

We now describe how we condense the survey answers into a regression dataset, and use this dataset to learn an ML model (as a formula) of interpretability.

We begin with replacing each question with a set of feature values that represents the formula contained in the question (explained in detail below). We obtain multiple identical sets of feature values with different outcomes in terms of correctness and confidence. We merge equal sets of feature values into a single sample, taking the ratio of correct answers and the mean confidence. In doing so, we do not distinguish between answers belonging to simulatability or decomposability, assuming they are equally good PHIs. We also remark that we did not

make expertise-based partitions because of the limited number of respondents and the skew in expertise distribution (Fig. 2).

As label to regress, we take the product between correctness ratio and confidence (the latter normalized to have values  $\frac{0}{3}, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}$ ). We choose to weight by confidence because, arguably, the less a user is confident about the answer, the less (s)he feels (s)he interprets the formula correctly. Essentially, this is a new PHI synthesized out of simulatability and decomposability, that takes confidence into account. From now on, we refer to this PHI as  $\phi$ .

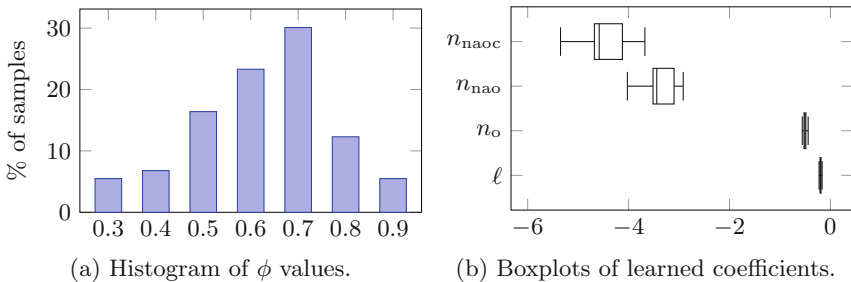
The choice of what formula features are considered is of crucial importance as it determines the way the answers are merged. We ultimately consider the following features: the size  $\ell$  of the formula (counting variables, constants, and operations), the number  $n_o$  of operations, the number  $n_{nao}$  of non-arithmetic operations, the number  $n_{naoc}$  of consecutive compositions of non-arithmetic operations. Note that the number of variables or constants is  $\ell - n_o$ , and the number of arithmetic operations is  $n_o - n_{nao}$ .

By merging answers sharing the same values for the aforementioned four features, and excluding merged samples composed by less than 10 answers for robustness, we obtain a small regression dataset with 73 samples.

#### 4.1 Learning the Model

Figure 3a shows the distribution of  $\phi$ . Since this distribution is not uniform, similarly to what is done for classification with imbalanced class frequency, we weight samples by the inverse frequency of the bin they belong to.

To obtain a readable ML model and due to the small number of samples, we choose to fit a elastic net linear model [53] of the four features with stochastic gradient descent, and validate it with leave-one-out cross-validation. We refer the reader interested in the details of this process (which includes, e.g., hyper-parameter tuning) to <https://github.com/MaLeLabTs/GPFormulasInterpretability>. The leave-one-out cross-validation scores a (weighted) training  $R^2 = 0.506$ , and (weighted) test  $R^2 = 0.545$  (mean weighted absolute error of 26%). The distribution of the model coefficients optimized across the folds is shown in Fig. 3b.



**Fig. 3.** Salient information about the learning data and the linear model.

We take the average coefficients found during the cross-validation to obtain the final model of  $\phi$  (from now on, considered as a percentage):

$$\mathcal{M}^\phi(\ell, n_o, n_{nao}, n_{naoc}) = 79.1 - 0.2\ell - 0.5n_o - 3.4n_{nao} - 4.5n_{naoc}. \quad (1)$$

By observing  $\mathcal{M}^\phi$ , it can be seen that each feature plays a role in lowering interpretability, yet by different magnitudes;  $n_{naoc}$  is the most important factor.

## 5 Exploiting the Model of Interpretability in MOGP

The experimental setup adopted for the use of the model  $\mathcal{M}^\phi$  within an MOGP algorithm for symbolic regression is presented next. We describe the algorithm we use, its objectives, the datasets we consider, and the evaluation process.

**MOGP by NSGA-II.** We use a GP version of NSGA-II [9], the most popular multi-objective evolutionary algorithm, and refer to it as *NSGP* (the same name has been used in different works, e.g., [23, 47, 49]). We use traditional settings (all described in [30]): tree-based encoding; ramped half-and-half initialization (min and max depth of 1 and 6 respectively); and tournament selection (size 2, default in NSGA-II). The crossover operator is subtree crossover (probability of 0.9, default in NSGA-II). The mutation operator is one-point mutation (probability of  $1/\ell$  for each tree node, with  $\ell$  the number of nodes).

We set the population size to 1000 and perform 100 generations across all experiments. The possible tree leaves are the problem variables and an ephemeral random constant [30], with random values from  $\mathcal{U}(-5, +5)$ . The operations are  $+$ ,  $-$ ,  $\times$ ,  $\div_p$ ,  $\sin$ ,  $\cos$ ,  $\exp$ , and  $\log_p$ . Protection of division by zero is implemented by  $\div_p(i_1, i_2) := \text{sign}(i_2) \frac{i_1}{|i_2|+\epsilon}$ . Similarly, the logarithm is protected by taking as argument the absolute value of the input plus  $\epsilon$ . We use  $\epsilon = 10^{-6}$ . Trees are not allowed to grow past 100 nodes, as they would definitely be not interpretable.

Our (Python 3) implementation of NSGP (including an interface to scikit-learn [29]) is available at: <https://github.com/marcovirgolin/pyNSGP>.

**Objectives.** We consider two competing objectives: error vs. interpretability. For the first objective, we consider the Mean Squared Error (MSE) with *linear scaling* [20, 21], i.e.,  $\text{MSE}^{\text{lin.scal.}}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N (y_i - a - b\hat{y}_i)^2$ . The use of the optimal (on training data) affine transformation coefficients  $a, b$  corresponds to computing an absolute correlation. From now on, we simply refer to this as MSE.

For the second objective, we consider two possibilities. The first one is realized by using our model  $\mathcal{M}^\phi$ : we extract the features from the tree to be evaluated, feed them to  $\mathcal{M}^\phi$ , and take the resulting estimate of  $\phi$ . To conform with objective minimization, we actually seek to minimize the opposite of this estimate (we also ignore the intercept term of Eq. (1)). We call *NSGP $^\phi$*  the version of NSGP using this second objective.

The second possibility is to solely minimize the number of nodes  $\ell$ . This approach, despite its simplicity, is very popular (see Sect. 2). Here we use it as a baseline for comparison. We refer to NSGP using  $\ell$  minimization as NSGP $^\ell$ .

The objectives based on  $\ell$  and  $\phi$  are in a comparable scale (considering  $\phi$  as a percentage). To bring the first objective to a similar scale (since scale impacts the crowding distance [9]), during the evolution we multiply the MSE by  $\frac{100}{\sigma^2(y)}$  (i.e., predicting the mean  $\mu(y)$  achieves an error of 100).

NSGP measures the quality of solutions according to the same criteria of [9] (domination ranking and crowding distance). We will report results relative to the front of non-dominated solutions  $\mathcal{F}$  obtained at the end of the run. Recall that a solution is non-dominated if there exists no other solution that is better in at least one objective and not worse in all others. In other words,  $\mathcal{F}$  contains the solutions with best trade-offs between the objectives.

**Datasets and Evaluation.** We consider 13 regression datasets in total, see Table 1. The first 5 datasets are synthetic (S-) and are recommended in [50]. The other 8 regard real-world data (R-) and are (mostly) taken from the UCI machine learning repository [12] and used in recent literature (e.g., [35,44]).

We treat all datasets the same. We apply standardization, i.e., all features are set to have zero mean and unit standard deviation. Before each run, we partition the dataset in exam at random, splitting it into 80% samples for training, 10% for validation, and 10% for testing. The training set is used by NSGP to evolve the solutions. The other sets are used to assess generalization, as is good practice in ML [3]. In particular, using the final population, we re-compute the MSE of the solutions w.r.t. the validation set, and compute the front of non-dominated solutions  $\mathcal{F}$  based on this. The MSE of the solutions in this front is finally re-evaluated on the test set (test MSE).

Because dataset partitioning as well as NSGP are stochastic, we perform 50 runs per dataset. To evaluate whether differences in results between NSGP $^\phi$  and NSGP $^\ell$  are significant, we use the Wilcoxon signed-rank test [10] to the 95% confidence level, including Holm-Bonferroni correction [19].

**Table 1.** Datasets used in this work. For the synthetic datasets,  $N$  is chosen by considering the largest between the training set and test set proposed in [50].

Dataset	Abbr.	$N$	$D$	$\mu(y)$	$\sigma(y)$	Dataset	Abbr.	$N$	$D$	$\mu(y)$	$\sigma(y)$
Keijzer 6	S-Ke6	121	1	4.38	0.98	Airfoil	R-Air	1503	5	124.8	6.9
Korns 12	S-K12	10000	5	2.00	1.06	Boston housing	R-Bos	506	13	22.5	9.2
Nguyen 7	S-Ng7	20	1	0.79	0.48	Dow chemical	R-Dow	1066	57	3.0	0.4
Pagie 1	S-Pa1	625	2	1.56	0.49	Diabetes	R-Dia	442	10	152.1	77.0
Vladislav. 4	S-Vl4	5000	5	0.49	0.19	Energy cooling	R-EnC	768	8	24.6	9.5
						Energy heating	R-EnH	768	8	22.3	10.1
						Tower	R-Tow	4999	25	342.1	87.8
						Yacht	R-Yac	308	6	10.5	15.1

## 6 Results

**Fitting and Generalization Error.** We begin by reporting quantitative results of the models in terms of training and test MSE. Although the test MSE is what ultimately matters in practical applications (i.e., a good formula is one that generalizes to unseen data), we also show the training MSE because it reflects the capability of an algorithm to optimize as much as possible. We present results for different trade-off levels  $\tau$ . Specifically,  $\tau$  is the percentile rank of the solutions in the non-dominated front  $\mathcal{F}$  ordered by increasing MSE:  $\tau = 1$  considers the solution with best MSE and worst PHI;  $\tau = 100$  considers the solution with worst MSE and best PHI (see Fig. 4). Table 2 shows the MSE obtained by NSGP $^\phi$  and NSGP $^\ell$  at training and test times, alongside the values of  $\phi$  and  $\ell$ , for the MSE-specialized part of the fronts ( $\tau = 5, 25, 50$ ).

For a same  $\tau$ , solutions found by NSGP $^\phi$  have typically larger  $\ell$  than those found by NSGP $^\ell$ . The vice versa also holds, as can be expected. Notable examples appear for  $\tau = 25$  in S-Pa1 and R-EnC/H: NSGP $^\phi$  achieves approximately double  $\ell$  compared to the NSGP $^\ell$ , while the latter achieves approximately double  $\phi$  compared to the former.

Regarding the training MSE, the use of  $\phi$  leads to the best optimization. This is particularly evident for  $\tau = 5$  where all results are significantly better when using NSGP $^\phi$ , except for S-V14. Using  $\phi$  instead of  $\ell$  has a smaller detrimental impact on finding well-fitting formulas. A plausible explanation is that NSGP $^\phi$  explores the search space better than NSGP $^\ell$ . This hypothesis is also supported by considering the sizes of the non-dominated fronts  $|\mathcal{F}|$ : although the fronts are generally small for both  $\phi$  and  $\ell$  (reasonable because both depend on discrete properties of the solutions [39]), they are consistently larger when  $\phi$  is used.

Less differences between NSGP $^\phi$  and NSGP $^\ell$  are significant when considering the test MSE (also due to Holm-Bonferroni correction). This is a normal consequence of assessing generalization as gains in training errors are lost due to (some) overfitting. What is important though is that NSGP $^\phi$  remains preferable. For  $\tau = 5$  ( $\tau = 25$ ), this is the case for 9 (7) out of 13 datasets.

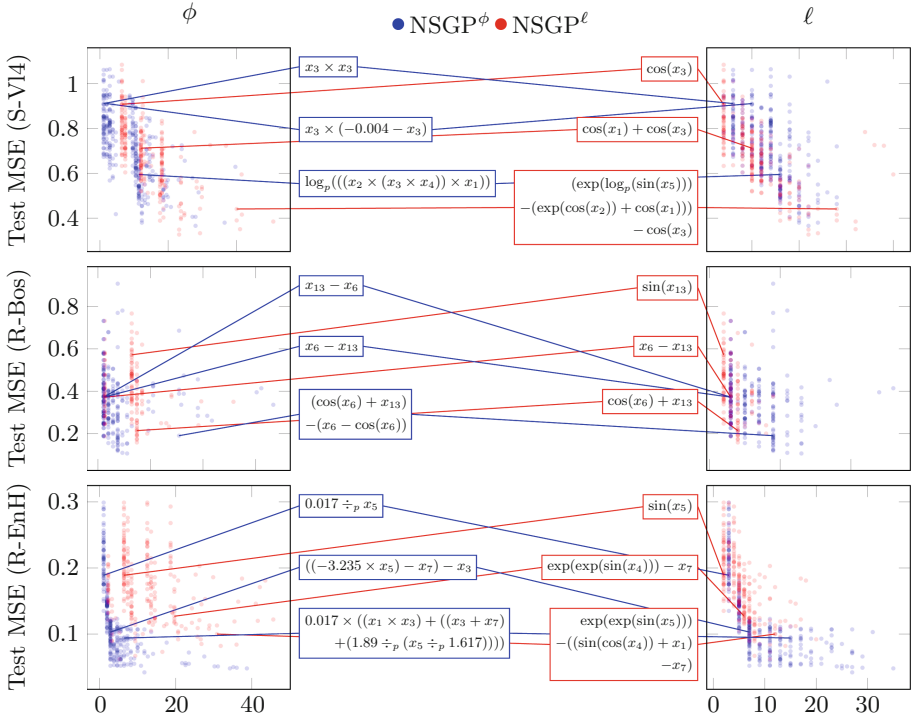
**Qualitative Results.** We delve deeper into the results to assess the behavior of NSGP using  $\phi$  and  $\ell$ , from a qualitative perspective. We consider three datasets: S-V14 where no version of NSGP is superior to the other; R-Bos where NSGP $^\phi$  is only better at training time; and R-EnH, where NSGP $^\phi$  is favorable also at test time. Figure 4 shows *all* validation fronts obtained from the 50 runs, re-evaluated in terms of test MSE for both versions of NSGP, and plotted w.r.t.  $\phi$  (left plots) and  $\ell$  (right plots). We also show, for  $\tau \in \{1, 50, 100\}$ , the solutions obtained by considering always the first run (seed 1 in the results on our online code repository at <https://github.com/MaLeLabTs/GPFormulasInterpretability>).

The scatter plots show that, in general, NSGP $^\phi$  obtains more points with small test MSE. This is most evident for R-EnH, where the results are found to be statistically significant. Note how, instead, this is not the case for  $\tau = 100$  in S-V14, where in fact the use of  $\phi$  is no better than the use of  $\ell$  (see Table 2).

**Table 2.** Median performance from 50 runs of the solutions found by NSGP $^\phi$  and NSGP $^\ell$  at different trade-off levels  $\tau$  ( $\tau = 1$  for best MSE,  $\tau = 100$  for best PHI). Median front sizes ( $|\mathcal{F}|$ ) are computed w.r.t. the validation set. MSE values in bold for one version of NSGP are significantly lower than the corresponding ones for the other version of NSGP at the 95% confidence level after Holm-Bonferroni correction.

Dataset	$\tau$	NSGP $^\phi$					NSGP $^\ell$					Train $p$ -value	Test $p$ -value
		Train MSE	Test MSE	$\phi$	$\ell$	$ \mathcal{F} $	Train MSE	Test MSE	$\phi$	$\ell$	$ \mathcal{F} $		
S-Ke6	5	<b>0.000</b>	<b>0.001</b>	11.4	11	7	0.007	0.006	14.5	8	5	0.000	0.000
	25	<b>0.001</b>	<b>0.002</b>	9.4	8		0.013	0.007	13.5	6		0.000	0.000
	50	<b>0.005</b>	<b>0.007</b>	3.8	7		0.023	0.023	7.4	4		0.000	0.000
S-K12	5	<b>0.997</b>	0.998	2.9	7	3	0.998	0.997	7.4	4	2	0.000	0.924
	25	<b>0.998</b>	0.998	2.9	7		0.998	0.997	7.4	4		0.000	0.941
	50	<b>0.998</b>	0.997	2.0	5		0.998	0.997	7.4	3		0.000	0.454
S-Ng7	5	<b>0.000</b>	<b>0.000</b>	4.7	9	4	0.004	0.003	12.6	4	2	0.000	0.000
	25	<b>0.001</b>	<b>0.001</b>	2.9	7		0.005	0.003	12.6	4		0.000	0.000
	50	<b>0.001</b>	<b>0.001</b>	2.0	5		0.005	0.003	12.6	3		0.000	0.000
S-Pa1	5	<b>0.174</b>	<b>0.190</b>	15.9	16	10	0.216	0.221	22.8	7	6	0.000	0.001
	25	0.221	0.231	14.1	12		0.257	0.269	19.7	6		0.038	0.004
	50	0.396	0.392	10.5	8		0.338	0.387	13.5	5		0.029	0.950
S-V14	5	0.509	0.536	13.9	9	6	0.580	0.563	18.1	8	5	0.194	0.241
	25	0.616	0.621	11.4	8		0.632	0.611	18.1	6		0.398	0.579
	50	0.770	0.719	10.5	6		<b>0.656</b>	0.684	12.0	5		0.000	0.004
R-Air	5	<b>0.501</b>	<b>0.519</b>	5.5	13	6	0.566	0.586	2.3	5	3	0.000	0.000
	25	<b>0.534</b>	<b>0.538</b>	4.7	10		0.566	0.586	2.3	5		0.000	0.000
	50	<b>0.565</b>	<b>0.586</b>	2.0	5		0.596	0.624	1.3	3		0.000	0.000
R-Bos	5	<b>0.245</b>	0.287	4.7	9	5	0.281	0.338	7.4	4	3	0.000	0.057
	25	<b>0.254</b>	0.290	3.8	9		0.282	0.338	7.4	4		0.000	0.021
	50	<b>0.283</b>	0.332	2.0	5		0.347	0.355	1.3	3		0.000	0.054
R-Dia	5	<b>0.510</b>	0.546	2.9	7	4	0.531	0.578	1.3	3	2	0.000	0.051
	25	<b>0.515</b>	0.546	2.9	7		0.533	0.577	1.3	3		0.000	0.046
	50	<b>0.525</b>	0.551	2.0	5		0.538	0.571	1.3	3		0.000	0.482
R-Dow	5	<b>0.336</b>	<b>0.357</b>	3.8	9	4	0.449	0.445	2.3	3	2	0.000	0.000
	25	<b>0.369</b>	<b>0.372</b>	3.8	9		0.449	0.451	2.3	3		0.000	0.000
	50	<b>0.395</b>	<b>0.418</b>	2.0	5		0.469	0.466	1.3	3		0.000	0.000
R-EnC	5	<b>0.099</b>	<b>0.108</b>	7.3	15	6	0.149	0.145	14.5	7	4	0.000	0.000
	25	<b>0.104</b>	<b>0.113</b>	5.5	12		0.157	0.155	13.8	7		0.000	0.000
	50	<b>0.117</b>	<b>0.127</b>	3.8	9		0.175	0.176	13.5	5		0.000	0.000
R-EnH	5	<b>0.082</b>	<b>0.085</b>	6.0	13	5	0.130	0.132	14.5	8	5	0.000	0.000
	25	<b>0.085</b>	<b>0.087</b>	4.7	11		0.142	0.141	13.5	7		0.000	0.000
	50	<b>0.089</b>	<b>0.098</b>	2.9	7		0.164	0.162	8.4	5		0.000	0.000
R-Tow	5	<b>0.290</b>	<b>0.288</b>	3.8	9	4	0.373	0.381	8.4	6	4	0.000	0.000
	25	<b>0.298</b>	<b>0.302</b>	2.9	7		0.379	0.389	3.3	5		0.000	0.000
	50	<b>0.371</b>	<b>0.370</b>	2.0	5		0.449	0.457	7.4	4		0.000	0.000
R-Yac	5	<b>0.011</b>	<b>0.014</b>	11.4	13	9	0.013	0.017	7.4	4	2	0.000	0.000
	25	<b>0.012</b>	0.016	5.5	11		0.013	0.017	7.4	4		0.000	0.037
	50	0.015	0.024	3.8	9		0.013	<b>0.018</b>	7.4	4		0.006	0.000





**Fig. 4.** Scatter plots of validation fronts as re-evaluated on the test set for all 50 runs, in terms of  $\phi$  (left column) and  $\ell$  (right column). Formulas in the middle are picked from the front of run 1, using  $\tau = 1$  (bottom), 50 (middle), 100 (top). Note that  $x_{13} - x_6$  and  $x_6 - x_{13}$  (R-Bos) are equivalent due to linear scaling.

By visually inspecting the formulas, we find results that are in line with what found in Table 2. Formulas found by  $\text{NSGP}^\phi$  with small MSE ( $\tau = 1, 50$ ) can often be (slightly) longer than their counterpart found by  $\text{NSGP}^\ell$  (except for S-V14), however, they typically contain less non-arithmetic operations, and less of their compositions. Even for very small formulas, those found  $\text{NSGP}^\ell$  rely more on non-arithmetic operations, meaning these operations help achieving small MSE, at least up to the validation stage. All in all, the most complex formulas found by  $\text{NSGP}^\phi$  are either more easily or similarly interpretable than those found by  $\text{NSGP}^\ell$ .

## 7 Discussion

To realize our data-driven approach, we relied on a survey aimed at measuring human-interpretability of mathematical formulas. While we did our best to design a survey that could gather useful human feedback, a clear limitation of our work is the relatively small number of survey respondents (334), which in

turn led to obtaining a relatively small dataset (73 samples, 4 formula features). Fitting of a high-bias (linear) model resulted in a decent test  $R^2$  of 0.5, while having the model be interpretable itself. Still, the model need not be interpretable. With more data available in the future, we will investigate the use of a larger number of more sophisticated features [26], and the use of more complex (possibly even black-box) models. Moreover, our approach can also be investigated for ML models other than formulas (e.g., decision trees).

In terms of results with NSGP, we found that  $\phi$  allows the discovery of good solutions w.r.t. the competing objective, i.e., the MSE, better than  $\ell$ . We also found that using  $\phi$  leads to the discovery of larger fronts. There is no reason to expect this outcome beforehand, as  $\phi$  was not designed to achieve this. We believe these findings boil down to one fundamental reason: diversity preservation. Because the estimation of  $\phi$  relies on more features compared to the measurement of  $\ell$ , more solutions can co-exist that do not dominate each other. Hence, the use of  $\phi$  fares better against premature convergence [40].

Regarding the examples of formulas we obtained, one may think that  $\phi$  leads to arguably more interpretable formulas than  $\ell$  simply because it accounts for non-arithmetic operations (and their composition). In fact, we agree that  $\ell$  is simplistic. However, we believe that minimizing  $\ell$  represents one of the first baselines to compare against (and it was the only one we found being used to specifically promote interpretability [22]), and that designing a competitive baseline is non-trivial. We will investigate this further in future work.

What about formula simplification? We did not present results regarding formulas after a simplification step. We attempted to use the sympy library [28] to assess the effect of formula simplification during the evolution, but to no avail as runtimes exploded. Moreover, we looked at what happens if we simplify (with sympy) the formulas in the final front, and re-compute their  $\phi$  and  $\ell$ . Results were mixed. For example, regarding the three datasets of Fig. 4, re-measuring  $\phi$  and  $\ell$  after simplification led to a mean improvement ratio of 1.08 (1.17) and 1.00 (1.00) respectively, when all (only the most complex) formulas from the fronts were considered. Hence, the use of  $\phi$  seems more promising than  $\ell$  w.r.t. simplification. Yet, as improvements were small (also in visual assessments), further investigation will be needed.

## 8 Conclusion

We presented a data-driven approach to learn, from responses to a survey on mathematical formulas we designed, a model of interpretability. This model is itself an interpretable (linear) formula, with reasonable properties. We plugged-in this model within multi-objective genetic programming to promote formula interpretability in symbolic regression, and obtained significantly better results when comparing with traditional formula size minimization. As such, our approach represents an important step towards better interpretable machine learning, especially by means of multi-objective evolution. Furthermore, the model we found can be used as a proxy of formula interpretability in future studies.

**Acknowledgments and Author Contributions.** We thank the Maurits en Anna de Kock Foundation for financing a high-performance computing system that was used in this work. Author contributions, in order of importance, follow. Conceptualization: M.V.; methodology: M.V., E.M.; software: M.V., A.D.L., F.R.; writing: M.V., E.M., A.D.L., F.R.

## References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE Access* **6**, 52138–52160 (2018)
2. Arrieta, A.B., et al.: Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion* **58**, 82–115 (2020)
3. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer, New York (2006)
4. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification and Regression Trees*. Wadsworth, Belmont (1984)
5. Burgess, T.F.: *Guide to the design of questionnaires*. A general introduction to the design of questionnaires for survey research. University of Leeds (2001)
6. Cano, A., Zafra, A., Ventura, S.: An interpretable classification rule mining algorithm. *Inf. Sci.* **240**, 1–20 (2013)
7. Chen, Q., Zhang, M., Xue, B.: Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression. *IEEE Trans. Evol. Comput.* **23**(4), 703–717 (2018)
8. Chouldechova, A.: Fair prediction with disparate impact: a study of bias in recidivism prediction instruments. *Big Data* **5**(2), 153–163 (2017)
9. 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)
10. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**(Jan), 1–30 (2006)
11. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint [arXiv:1702.08608](https://arxiv.org/abs/1702.08608) (2017)
12. Dua, D., Graff, C.: UCI machine learning repository (2017). [archive.ics.uci.edu/ml](https://archive.ics.uci.edu/ml)
13. Ekárt, A., Nemeth, S.Z.: Selection based on the Pareto nondomination criterion for controlling code growth in genetic programming. *Genet. Program Evolvable Mach.* **2**(1), 61–73 (2001)
14. Evans, B.P., Xue, B., Zhang, M.: What’s inside the black-box? A genetic programming method for interpreting complex machine learning models. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1012–1020 (2019)
15. Goodman, B., Flaxman, S.: European union regulations on algorithmic decision-making and a “right to explanation”. *AI Mag.* **38**(3), 50–57 (2017)
16. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. *ACM Comput. Surv.* **51**(5), 93 (2018)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
18. Hein, D., Udluft, S., Runkler, T.A.: Interpretable policies for reinforcement learning by genetic programming. *Eng. Appl. Artif. Intell.* **76**, 158–169 (2018)
19. Holm, S.: A simple sequentially rejective multiple test procedure. *Scand. J. Stat.* **6**, 65–70 (1979)

20. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36599-0\\_7](https://doi.org/10.1007/3-540-36599-0_7)
21. Keijzer, M.: Scaled symbolic regression. *Genet. Program Evolvable Mach.* **5**(3), 259–269 (2004)
22. Lensen, A., Xue, B., Zhang, M.: Genetic programming for evolving a front of interpretable models for data visualization. *IEEE Trans. Cybern.*, 1–15 (2020). <https://ieeexplore.ieee.org/abstract/document/9007046>
23. Liang, Y., Zhang, M., Browne, W.N.: Multi-objective genetic programming for figure-ground image segmentation. In: Ray, T., Sarker, R., Li, X. (eds.) ACALCI 2016. LNCS (LNAI), vol. 9592, pp. 134–146. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-28270-1\\_12](https://doi.org/10.1007/978-3-319-28270-1_12)
24. Lipton, Z.C.: The mythos of model interpretability. *Queue* **16**(3), 31–57 (2018)
25. Lou, Y., Caruana, R., Gehrke, J.: Intelligible models for classification and regression. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 150–158. ACM (2012)
26. Maruyama, M., Pallier, C., Jobert, A., Sigman, M., Dehaene, S.: The cortical representation of simple mathematical expressions. *Neuroimage* **61**(4), 1444–1460 (2012)
27. McCormack, J., Lomas, A.: Understanding aesthetic evaluation using deep learning. In: Romero, J., Ekárt, A., Martins, T., Correia, J. (eds.) EvoMUSART 2020. LNCS, vol. 12103, pp. 118–133. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-43859-3\\_9](https://doi.org/10.1007/978-3-030-43859-3_9)
28. Meurer, A., et al.: SymPy: symbolic computing in Python. *PeerJ Comput. Sci.* **3**, e103 (2017)
29. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
30. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: A Field Guide to Genetic Programming. Lulu.com, Morrisville (2008)
31. Poli, R., McPhee, N.F.: Parsimony pressure made easy: solving the problem of bloat in GP. In: Borenstein, Y., Moraglio, A. (eds.) Theory and Principled Methods for the Design of Metaheuristics. NCS, pp. 181–204. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-33206-7\\_9](https://doi.org/10.1007/978-3-642-33206-7_9)
32. Poursabzi-Sangdeh, F., Goldstein, D.G., Hofman, J.M., Vaughan, J.W., Wal-lach, H.: Manipulating and measuring model interpretability. *arXiv preprint arXiv:1802.07810* (2018)
33. Raymond, C., Chen, Q., Xue, B., Zhang, M.: Genetic programming with Rademacher complexity for symbolic regression. In: IEEE Congress on Evolutionary Computation (CEC), pp. 2657–2664 (2019)
34. Ribeiro, M.T., Singh, S., Guestrin, C.: “Why should I trust you?” Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144 (2016)
35. Ruberto, S., Terragni, V., Moore, J.H.: SGP-DT: semantic genetic programming based on dynamic targets. In: Hu, T., Lourenço, N., Medvet, E., Divina, F. (eds.) EuroGP 2020. LNCS, vol. 12101, pp. 167–183. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-44094-7\\_11](https://doi.org/10.1007/978-3-030-44094-7_11)
36. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **1**(5), 206–215 (2019)

37. Sambo, A.S., Azad, R.M.A., Kovalchuk, Y., Indramohan, V.P., Shah, H.: Time control or size control? Reducing complexity and improving accuracy of genetic programming models. In: Hu, T., Lourenço, N., Medvet, E., Divina, F. (eds.) EuroGP 2020. LNCS, vol. 12101, pp. 195–210. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-44094-7\\_13](https://doi.org/10.1007/978-3-030-44094-7_13)
38. Silva, S., Dignum, S., Vanneschi, L.: Operator equalisation for bloat free genetic programming and a survey of bloat control methods. *Genet. Program Evolvable Mach.* **13**(2), 197–238 (2012)
39. Smits, G.F., Kotanchek, M.: Pareto-front exploitation in symbolic regression. In: O’Reilly, U.M., Yu, T., Riolo, R., Worzel, B. (eds.) *Genetic Programming Theory and Practice II*. GPEM, vol. 8, pp. 283–299. Springer, Boston (2005). [https://doi.org/10.1007/0-387-23254-0\\_17](https://doi.org/10.1007/0-387-23254-0_17)
40. 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)
41. Tran, B., Xue, B., Zhang, M.: Genetic programming for multiple-feature construction on high-dimensional classification. *Pattern Recogn.* **93**, 404–417 (2019)
42. Vanneschi, L., Castelli, M., Silva, S.: Measuring bloat, overfitting and functional complexity in genetic programming. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 877–884 (2010)
43. Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.N.: Improving model-based genetic programming for symbolic regression of small expressions. Accepted in *Evolutionary Computation*. ArXiv preprint [arXiv:1904.02050](https://arxiv.org/abs/1904.02050) (2019)
44. Virgolin, M., Alderliesten, T., Bosman, P.A.N.: Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019*, pp. 1084–1092. Association for Computing Machinery (2019)
45. Virgolin, M., Alderliesten, T., Bosman, P.A.N.: On explaining machine learning models by evolving crucial and compact features. *Swarm Evol. Comput.* **53**, 100640 (2020)
46. 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 (2008)
47. Wang, P., Tang, K., Weise, T., Tsang, E., Yao, X.: Multiobjective genetic programming for maximizing ROC performance. *Neurocomputing* **125**, 102–118 (2014)
48. Wang, W., Shen, J.: Deep visual attention prediction. *IEEE Trans. Image Process.* **27**(5), 2368–2378 (2017)
49. Watchareeruetai, U., Matsumoto, T., Takeuchi, Y., Kudo, H., Ohnishi, N.: Construction of image feature extractors based on multi-objective genetic programming with redundancy regulations. In: *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1328–1333. IEEE (2009)
50. White, D.R., et al.: Better GP benchmarks: community survey results and proposals. *Genet. Program Evolvable Mach.* **14**(1), 3–29 (2013)
51. Zhang, B.T., Mühlenbein, H.: Balancing accuracy and parsimony in genetic programming. *Evol. Comput.* **3**(1), 17–38 (1995)
52. Zhao, H.: A multi-objective genetic programming approach to developing Pareto optimal decision trees. *Decis. Support Syst.* **43**(3), 809–826 (2007)
53. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. *J. Roy. Stat. Soc.: Ser. B (Stat. Methodol.)* **67**(2), 301–320 (2005)

# **Landscape Analysis**



# On Stochastic Fitness Landscapes: Local Optimality and Fitness Landscape Analysis for Stochastic Search Operators

Brahim Aboutaib<sup>1,2</sup>(✉), Sébastien Verel<sup>1</sup>, Cyril Fonlupt<sup>1</sup>, Bilel Derbel<sup>3</sup>,  
Arnaud Liefoghe<sup>4</sup>, and Belaïd Ahiod<sup>2</sup>

<sup>1</sup> Université du Littoral Côte d'Opale, LISIC, 62100 Calais, France  
[brahim.aboutaib@univ-littoral.fr](mailto:brahim.aboutaib@univ-littoral.fr)

<sup>2</sup> Faculty of Science, LRIT, Mohammed V University in Rabat, Rabat, Morocco

<sup>3</sup> Univ. Lille, CNRS, Centrale, Inria, UMR 9189 - CRIStAL, 59000 Lille, France

<sup>4</sup> JFLI – CNRS IRL 3527, University of Tokyo, Tokyo 113-0033, Japan

**Abstract.** Fitness landscape analysis is a well-established tool for gaining insights about optimization problems and informing about the behavior of local and evolutionary search algorithms. In the conventional definition of a fitness landscape, the neighborhood of a given solution is a set containing nearby solutions whose distance is below a threshold, or that are reachable using a deterministic local search operator. In this paper, we generalize this definition in order to analyze the induced fitness landscape for *stochastic* search operators, that is when neighboring solutions are reachable under different probabilities. More particularly, we give the definition of a *stochastic* local optimum under this setting, in terms of a probability to reach strictly improving solutions. We illustrate the relevance of stochastic fitness landscapes for enumerable combinatorial benchmark problems, and we empirically analyze their properties for different stochastic operators, neighborhood sample sizes, and local optimality thresholds. We also portray their differences through stochastic local optima networks, intending to gather a better understanding of fitness landscapes under stochastic search operators.

**Keywords:** Combinatorial optimization · Local optimality · Fitness landscape · Stochastic search operators.

## 1 Introduction

Originally coming from evolutionary biology [23], the fitness landscape is one of the most common abstractions used to depict and analyze dynamical systems. In evolutionary computation and related stochastic optimization algorithms, a fitness landscape is the association of a search space of potential solutions, a fitness function to be optimized, and a neighborhood relation between solutions on which the optimization process is expected to move during the search process. The aim of fitness landscape analysis is twofold. The first one is to understand

the relation between the geometry of the optimization problem and the search dynamics, using the pictures of peaks, valleys, or plateaus. More recently, with the renew of machine learning techniques, the second goal uses features computed from the fitness landscape to predict algorithm performance or to select relevant algorithm components according to the optimization problem [10]. From the large number of features designed for evolutionary computation, one of the most intuitive and most important one relates to *local optimality*. A local optimum, which can be depicted as a peak of the landscape, is a solution with the best fitness value locally, among its set of neighboring solutions. This definition arises from the neighborhood relation of the fitness landscape. In combinatorial optimization, the neighborhood is a finite set, most often defined by a natural distance between solutions such as Hamming distance or Kendall distance.

However, most local search and evolutionary algorithms use stochastic local search operators, which do not always directly match with a finite set of neighboring solutions. For example, the bit-flip mutation operator conventionally used in genetic algorithms, which flips each bit independently with a given rate, does not produce a finite set of neighboring solutions with an equal probability of being reached. Similarly, hyperheuristics might combine several local search operators with different probability distributions [16]. The explored solutions at each iteration is not properly caught by any finite set. Even when the neighborhood is defined as a finite set, its cardinality might be too large for being computationally enumerated, as it is the case, for instance, in genetic programming or population-based multi-objective search [20]. Therefore, a subpart of the neighborhood is typically sampled at random or in a heuristic way, making the deterministic definition of neighborhood and local optimum less relevant.

In this paper, we extend the definition of neighborhood to stochastic local search operators, and we investigate *stochastic local optimality* in this context. The principle is to define a stochastic local optimum when the probability to strictly improve a solution by applying a stochastic operator is small, below a given threshold. The threshold can be related to the inverse of the computational budget available to find an improving solution. Intuitively, a stochastic local optimum is a solution that is difficult to improve in a reasonable computational effort using the stochastic operator, and therefore constitutes an attraction point for evolutionary or local search. By extending the fitness landscape paradigm, and in particular its neighborhood relation, to stochastic search operators, we expect the definition of local optimality to reveal new insights into the search space structure, and to allow for a better analysis of the design of algorithms based on stochastic operators. In this work, the methodology is to support the relevance of the new definition of stochastic local optimum with an experimental analysis on enumerable combinatorial optimization problems with different properties, and to show the potential additional benefit of this approach for fitness landscape analysis.

Our contributions can be summarized as follows:

- (1) We extend the definition of fitness landscape to stochastic operators, and we propose a new definition of local optimum for stochastic local search.



- (2) We empirically show the relevance of this definition on enumerable NK-landscapes.
- (3) We show some potential interests of stochastic fitness landscapes for iterated local search, and we reveal, for the first time, the structure of local optima networks for stochastic operators.

*Outline.* The paper is organized as follows. In Sect. 2, we first recall necessary definitions and related works. In Sect. 3, we present the stochastic fitness landscape and stochastic local optimality definitions. In Sect. 4, we detail the experimental setup, and the results of the analysis. The discussion and conclusions close the paper in Sect. 5.

## 2 Preliminaries

A *fitness landscape* [14] is defined as a triplet  $(\mathcal{X}, f, \mathcal{N})$  such that  $\mathcal{X}$  is the search space of candidate solutions,  $f : \mathcal{X} \rightarrow \mathbb{R}$  is the fitness function, and  $\mathcal{N}$  is a neighborhood relation. In combinatorial optimization, the search space is a finite set. The neighborhood relation  $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$  assigns a set of solutions, called *neighbors*, to any solution from the search space. Although this definition is quite general, the basic idea behind it is to define neighboring solutions in the vicinity of a solution. Standard definitions are based on a distance measure between elements from the search space, such as the Hamming distance:  $\mathcal{N}(x) = \{x' \in \mathcal{X} \mid d(x, x') \leq D\}$ , where  $d$  is a distance function and  $D$  is the radius of the neighborhood. Other standard definitions are based on a deterministic local search operator that performs a move from one solution to another:  $\mathcal{N}(x) = \{x' \in \mathcal{X} \mid \exists \theta \text{ s.t. } op_{\theta}(x) = x'\}$ , where  $op_{\theta}$  is a parametric deterministic operator such as swap or insertion. In both cases, the neighborhood relation allows one to depict the fitness landscape with peaks, plateaus, and valleys, but also to analyze the fitness landscape using tools from graph theory [14, 21].

One of the main intuitive and fundamental feature of a fitness landscape deals with local optimality, where local optima represent the peaks in the pictures collection of the fitness landscape. For a maximization problem, a *local optimum* is a solution  $x \in \mathcal{X}$  such that:

$$\forall x' \in \mathcal{N}(x), f(x') \leq f(x) \quad (1)$$

The number of local optima in the fitness landscape provides a first information about the difficulty of a combinatorial optimization problem, and about the performance of local and evolutionary search algorithms [8]. For large landscapes, different methods allow one to estimate the number of local optima using uniform random sampling, biased random sampling [1, 7], or the length of an adaptive walk before being trapped [11]. In addition to the number of local optima, the size, the distribution and the structure of local optima's basins of attraction is one major feature related to algorithm performance [5, 8], including for problems from machine learning [3]. The basin of attraction of a local optimum  $x^*$  is defined as the set of solutions from which a hill-climbing algorithm  $h$  falls into:

$\mathcal{B}(x^*) = \{y \in \mathcal{X} \mid h(y) = x^*\}$ . Depending on the pivot rule used by the hill-climber, e.g., first or best improvement, the structure of the basins is different, and so is its impact on search performance [2, 15]. Besides local optima and their basins of attraction, a complementary view of fitness landscapes is given by the so-called *local optima network* (LON) [4, 13]. In particular, the LON with escape edges [19] is defined as a weighted directed graph  $(V, E)$ , where vertices  $V$  are local optima, and there is an edge  $(x_i, x_j) \in E$  between local optima  $x_i$  and  $x_j$  if there is a solution  $y \in \mathcal{X}$  such that the distance between  $x_i$  and  $y$  is below a given threshold, and  $y$  belongs to the basin of attraction of  $x_j$ ; i.e.  $(x_i, x_j) \in E$  iff  $\exists y \in \mathcal{X}$  such that  $d(x_i, y) \leq D$  and  $y \in \mathcal{B}(x_j)$ . The weight  $w_{ij}$  of an edge  $(x_i, x_j)$  gives the ratio of such solutions  $y \in \mathcal{X}$  that satisfies the definition above from the set of solutions at distance  $D$ . Several metrics have been proposed to characterize LONs, and have been related to problem difficulty or search performance for both single-objective [18] and multi-objective optimization [6, 11].

However, although many evolutionary and local search algorithms are based on a stochastic operator, all definitions related to local optimality and fitness landscape analysis are based on a deterministic neighborhood relation, considering a set of neighbors that is often finite. We argue that this only partially reflects the properties of stochastic search operators, and we introduce the notion of stochastic local optimality and fitness landscapes in the next section.

### 3 Stochastic Fitness Landscapes and Local Optimality

A local search algorithm is based on a local search operator  $\text{op} : \mathcal{X} \rightarrow \mathcal{X}$  which moves from one solution to another solution. A stochastic local search operator defines a probability distribution over the search space. When the search space is finite, as in discrete or combinatorial optimization, a stochastic operator can be defined by the probabilities of moving from a solution  $x \in \mathcal{X}$  to a solution  $y \in \mathcal{X}$ :  $\mathbb{P}\{\text{op}(x) = y\} = p_{x \rightarrow y}$  such that  $\forall x, y \in \mathcal{X}, p_{x \rightarrow y} \geq 0$ , and  $\forall x \in \mathcal{X}, \sum_{y \in \mathcal{X}} p_{x \rightarrow y} = 1$ . When the search space is infinite, a stochastic operator can be defined with probability density functions over the search space.

A typical example of a stochastic local search operator is the bit-flip mutation operator used in standard genetic algorithms. It flips each bit from a given bitstring of size  $n$  independently at random, with a rate  $p$ . In this case,  $p_{x \rightarrow y} = p^k(1-p)^{n-k}$ , where  $k$  is the Hamming distance between  $x$  and  $y$ . Besides, a stochastic operator can be derived from any neighborhood relation with finite support. For a given neighborhood relation  $\mathcal{N}$  such that,  $\forall x \in \mathcal{X}, |\mathcal{N}(x)| < \infty$ , a stochastic operator  $\text{op}_{\mathcal{N}}$  can be defined using a uniform random distribution over the set of neighbors:  $\forall y \in \mathcal{N}(x), \mathbb{P}\{\text{op}_{\mathcal{N}}(x) = y\} = \frac{1}{|\mathcal{N}(x)|}$ .

It is straightforward to extend the definition of a fitness landscape by replacing the neighborhood relation with a stochastic search operator.

**Definition 1.** A *stochastic fitness landscape (SFL)* is a triplet  $(\mathcal{X}, \text{op}, f)$  where  $\mathcal{X}$  is the search space,  $f : \mathcal{X} \rightarrow \mathbb{R}$  the fitness function, and  $\text{op}$  is the stochastic local search operator.

However, this definition only makes sense if it is possible to define reasonable basic features of such fitness landscapes. As pointed out in Sect. 2, one of the main features is the concept of local optimality. According to Eq. (1), a local optimum is a solution for which all neighbors have a lower or equal fitness value. Roughly speaking, if we translate this definition in terms of probability, the probability to reach a neighbor with a lower or equal fitness value from a local optimum is very high. However, this probability cannot always be considered as 1. Indeed, for an ergodic operator and a finite search space, there is a non-zero probability to reach the global optimum from any solution from the search space. As such, considering a probability of 1 would end up having the global optimum as the single local optimum. We thus introduce a threshold  $\epsilon > 0$  to define a stochastic local optimum. The probability to reach a neighbor with lower quality is higher than  $(1 - \epsilon)$ :  $\mathbb{P}\{f(\text{op}(x)) \leq f(x)\} \geq (1 - \epsilon)$ , or equivalently  $\mathbb{P}\{f(x) < f(\text{op}(x))\} \leq \epsilon$ .

**Definition 2.** *Given a SFL  $(\mathcal{X}, \text{op}, f)$ , and a real number  $\epsilon \geq 0$ , a solution  $x \in \mathcal{X}$  is a stochastic local optimum (SLO) at a local optimality threshold  $\epsilon$  iff  $\mathbb{P}\{f(x) < f(\text{op}(x))\} \leq \epsilon$ .*

In other words, a solution is a stochastic local optimum at threshold  $\epsilon$  when the probability to reach a strictly improving solution by applying the stochastic operator is below  $\epsilon$ . Notice that the definition is still effective when the probability to obtain the same solution is not null, i.e. when  $\mathbb{P}\{\text{op}(x) = x\} > 0$ .

Interestingly, the definitions of deterministic and stochastic local optimality can be connected. Let us consider a neighborhood relation with finite support; i.e.  $\forall x \in \mathcal{X}, |\mathcal{N}(x)| < \infty$ . A solution  $x \in \mathcal{X}$  is a deterministic local optimum under  $\mathcal{N}$  iff  $x$  is a stochastic local optimum under  $\text{op}_{\mathcal{N}}$  at threshold  $\epsilon < 1/|\mathcal{N}(x)|$ . Indeed, the probability to strictly improve a solution  $x$  is equal to  $n^+ / |\mathcal{N}(x)|$ , where  $n^+$  is the number of strictly improving neighbors. As a consequence, when  $x$  is a deterministic local optimum for  $\mathcal{N}$ , the probability to strictly improve  $x$  with  $\text{op}_{\mathcal{N}}$  is zero. Conversely, when the probability to strictly improve  $x$  is strictly below  $1/|\mathcal{N}(x)|$ , no neighbor from  $\mathcal{N}(x)$  has a higher fitness value.

In the definition of a stochastic local optimum, the threshold  $\epsilon$  is critical, and a relevant value has to be carefully chosen with respect to the considered landscape and search scenario. Intuitively, we can think of the  $\epsilon$ -value as the inverse of the expected computational budget (in terms of fitness evaluations) required to escape from a local optimum with a stochastic operator. For instance, in one scenario, if at a given step of the search process, the remaining computational budget is  $n_{eval}$ , we could define  $\epsilon = \frac{1}{n_{eval}}$ . Alternatively, in an iterated local search scenario switching between a local search operator and a perturbation operator, the threshold could be defined as the inverse of the budget dedicated to each local search run:  $\epsilon = \frac{1}{n_{ls}}$ . At last, in an evolutionary algorithm scenario for which  $\lambda$  candidate solutions are computed at each iteration, as in a  $(\mu + \lambda)$ -evolution strategy, the  $\epsilon$ -value can be set to  $\epsilon = \frac{1}{\lambda}$ . As such, the probability of strictly improving the current population is below the computational budget of one iteration of the algorithm. Furthermore, we argue that a fitness landscape

analysis could actually benefit from the use of a broader range of  $\epsilon$ -values. Fitness landscape metrics such as the number of local optima or the size of the basins of attraction can be studied according to the  $\epsilon$ -value in order to show the metrics spectrum, and not only for a given accurate value of  $\epsilon$ . We illustrate this point and analyze the impact of  $\epsilon$  empirically in the next section.

## 4 Experimental Analysis

### 4.1 Experimental Setup

**Problem Testbed.** We consider NK-landscapes as a problem-independent model of combinatorial optimization problems defined on binary strings. NK-landscapes were proposed in [9] for constructing multi-modal fitness landscapes in a tunable way, by adjusting the epistatic (or non-linearity) degree  $K$ . Given a binary string  $x \in \{0, 1\}^N$ , its fitness  $f(x) \in [0, 1]$ , is defined as follows:

$f(x) := \frac{1}{N} \sum_{i=1}^N f_i(x_i, x_{i_1}, \dots, x_{i_K})$ , where  $f_i : \{0, 1\}^{K+1} \rightarrow [0, 1]$ : is the epistasis

level of the  $i$ -th bit, its value depend on the allele at the bit  $i$  but also on the other alleles at the  $K$  other epistatic bits, and  $\{i_1, \dots, i_K\} \subset \{1, \dots, i-1, i+1, \dots, N\}$ . For each variable  $x_i$ , there exist two ways for selecting the  $K$  epistatic bits: either randomly, or by choosing the  $K$  closest ones. Beside that NK-landscapes belong to the family of NP-hard problems for  $K > 1$  [22], it has the property of modeling many interesting optimization problems [21]. Thus, it is not just a serious testbed for randomized search heuristics, but also a proxy for other combinatorial optimization problems. We shall mention that the best algorithm to tackle NK-landscapes is not our concern here. We do not consider this problem for benchmarking purposes, but rather for illustration/concept-testing purposes, given that the number of local optima is known to be closely related to the epistatic degree  $K$ .

In order to analyze the impact of different parameters on the definition of SLO, we consider enumerable instances from NK-landscapes with a bitstring length  $N \in \{10, 12, 16, 18\}$ , and an epistatic degree  $K \in \{0, 1, 2, 3\}$ . These  $K$ -values were chosen as they correspond to linear, quadratic, cubic, and quartic versions, the most recurrent problem types encountered in combinatorial optimization. We report the result over 20 different instances for each combination of  $N$  and  $K$ , that we generate with adjacent epistatic interactions.

**Stochastic Operators.** We investigate the bit-flip mutation operator as a stochastic local search operator. It flips each bit independently at random, with a rate  $p$ . We experiment different rates  $p = \frac{c}{N}$ , inversely proportional to the bitstring length  $N$ , such that  $c \in \{1, 2, 4, 8\}$ . For comparison purposes, we also consider the standard 1-bit flip neighborhood operator, which flips 1 bit precisely. Thus, the neighborhood consists of all solutions located at Hamming distance 1, and the neighborhood size is  $N$ .

**Estimating Stochastic Local Optimality.** Even for small enumerable problem instances, it is not computationally doable to enumerate all possible neighbors for each solution to compute exactly all SLO from the search space. Therefore, to estimate the improving probability given in Definition 2, we use the classical estimator of the empirical mean:  $\hat{p}^+ = \frac{\lambda^+}{\lambda}$  where  $\lambda^+$  is the number of strictly improving solution over a random sample of  $\lambda$  solutions produced by the stochastic operator. Thus, a solution is depicted as a SLO when this estimation  $\hat{p}^+$  is lower or equal to the threshold  $\epsilon$ . The quality of the estimation depends on local optimality threshold values, and the sample size  $\lambda$ . We study different values, proportional to the bitstring length:  $\epsilon \in \{1/(j \cdot N), \mid j \in \{1, 2, 4, 8\}\}$ , and  $\lambda \in \{2^i N \mid i \in \{0, \dots, 7\}\}$ .

## 4.2 Experimental Results

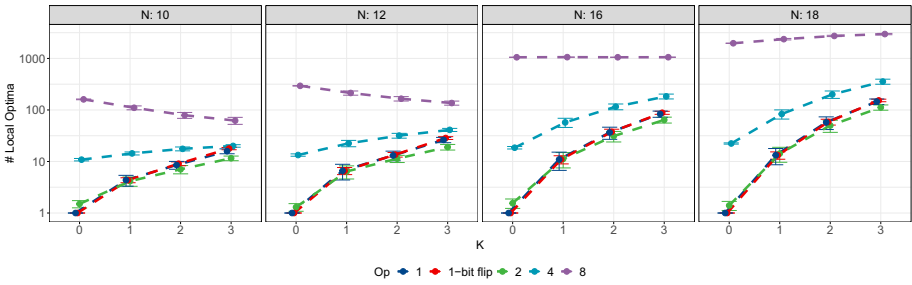
In this section, we illustrate the relevance of the SLO definition, and we show preliminary scenarios for analyzing stochastic fitness landscapes.<sup>1</sup>

**Number of Stochastic Local Optima.** Figure 1 shows the number of SLO over all instances according to the degree of non-linearity  $K$ , the problem dimension  $N$ , and different mutation rates  $p$ . The threshold is set to  $\epsilon = 1/(4N)$  (other values are reported below). In order to increase the estimation accuracy, the sample size is set to the largest value  $\lambda = 128N$ . Except for a very large mutation rate  $p = 8/N$ , and a small problem dimension  $N < 16$ , the expected number of SLO seems to increase fast with the degree of non-linearity  $K$ . It also increases with the problem size  $N$ . As expected, the trend follows the number of local optima for the classical 1 bit-flip neighborhood. These first results show that the proposed definition of SLO make sense w.r.t. the multimodality of NK-landscapes. The number of SLO is also impacted by the mutation rate, and a more precise analysis is detailed below.

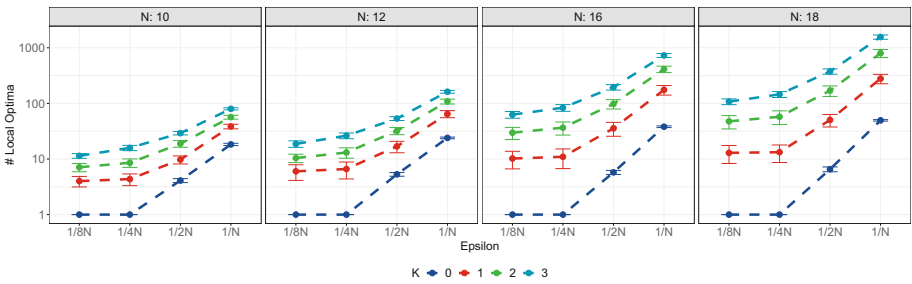
**Local Optimality Threshold.** In Fig. 2, we report the number of SLO as a function of the local optimality threshold  $\epsilon$ . The sample size is set to the largest value  $\lambda = 128N$ , and the bit-flip mutation rate is set to the typical setting of  $p = 1/N$ . The number of SLO increases with  $\epsilon$  for all problem instances. Indeed, a small threshold allows fewer solutions to be a SLO. Let us remind that the extreme setting of  $\epsilon = 0$  implies that a single solution is a SLO: the global optimum. Notice that the number of SLO increases with the degree of non-linearity  $K$ , and the problem dimension  $N$  for all investigated  $\epsilon$ -values. For large  $\epsilon$ -values, the number of SLO is larger than for the standard 1 bit-flip neighborhood. In this case, the inverse of the  $\epsilon$ -value is larger than  $1/N$ .

**Impact of the Sample Size.** We now investigate how the sample size impacts the estimation of the number of SLO. From a statistical point of view, a large

<sup>1</sup> Code, and data are available on <https://gitlab.com/b.aboutaib/slo>.



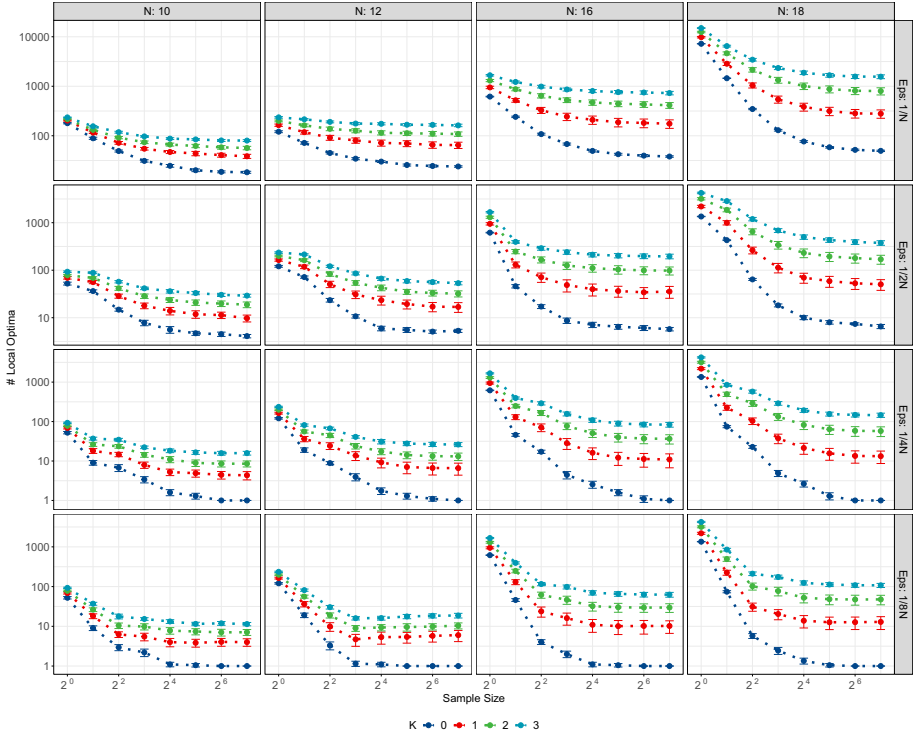
**Fig. 1.** Number (average value and confidence interval, computed over 20 considered instances) of SLO with respect to the problem non-linearity  $K$ , for different problem sizes  $N$  and stochastic operators  $op$  with a bit-flip mutation rate  $c/N$ .



**Fig. 2.** Number (average value and confidence interval) of SLO with respect to the local optimality threshold  $\epsilon$ , for different instances (size  $N$  and non-linearity  $K$ ).

sample size gives a better estimation of the improving probability considered in Definition 2. Therefore, a better estimation of the number of SLO is expected. Figure 3 shows the estimated number of SLO as a function of the sample size for different values of  $N$ ,  $K$ , and  $\epsilon$ . The estimated number decreases with the sample size to converge toward the number of SLO. The empirical mean estimator of improving probability tends to overestimate the number of SLO when the sample size is too small with respect to the  $\epsilon$ -value. At a first sight, a sample size of about 2 or 3 times the inverse of  $\epsilon$  seems to provide a fair estimation. However for a given sample size, the estimated number of SLO increases with non-linearity  $K$ , and problem dimension  $N$ . Further theoretical studies should allow us to improve this first empirical finding.

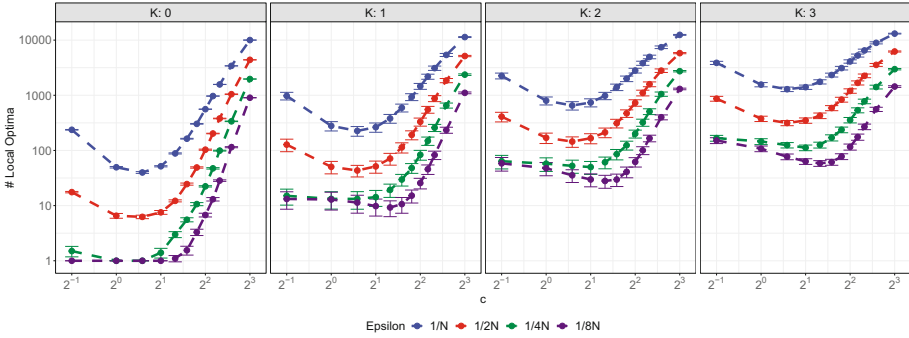
**Mutation Rate vs. Local Optimality Threshold.** In Fig. 4, we report the number of SLO as a function of the bit-flip mutation rate  $p$ , for different  $\epsilon$ -values and different degrees of non-linearity  $K$ . The problem dimension is set to  $N = 18$ , and the largest sample size of  $\lambda = 128 N$  is used. In order to better appreciate the trend, additional rate values are experimented:  $p = c/N$ , with  $c \in \{0.5, 1, \dots, 8\}$ . The local optimality threshold is set to  $\epsilon \in \{1/N, 1/(2N), 1/(4N), 1/(8N)\}$ .



**Fig. 3.** Number (average value and confidence interval) of SLO with respect to the sample size, for different instances (size  $N$  and non-linearity  $K$ ) and local optimality threshold  $\epsilon$ . The mutation rate is set to  $p = 1/N$  (i.e.,  $c = 1$ ).

All curves have a convex shape. As such, given a threshold value  $\epsilon$ , there is a bit-flip mutation rate that minimizes the expected number of SLO, which does not map to the extreme bounds of the domain. In other words, the lowest number of SLO is reached at a particular trade-off point between low and high bit-flip mutation rates. For example, for  $K = 2$  and  $\epsilon = 1/(2N)$ , the mutation rate that minimizes the number of SLO is  $1.5/N$ , but changing  $\epsilon$  to  $1/(8N)$  shifts this mutation rate to  $2.5/N$ . Interestingly, this observation suggests that there exists an accurate mutation rate that reduces the multimodality of the stochastic fitness landscape, and that does *not* correspond to the largest bit-flip mutation rate. To the best of our knowledge, this is the first fitness landscape analysis that brings an understanding of a relevant mutation rate for a stochastic operator. A high mutation rate results in a landscape with many local optima. Hence, a local search would be easily stuck, and would not benefit from potential local improvements. A low mutation rate also induces a large number of local optima. However, by contrast, it shall be understood as a lack of exploration with respect to the local optimality threshold. Remember that a single feature

is not able to explain all facets of search difficulty, and other metrics from the fitness landscape would be required to have a better global picture.



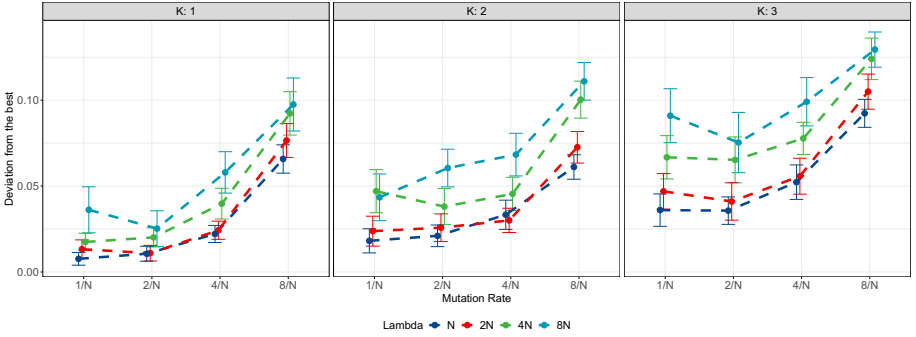
**Fig. 4.** Number (average value and confidence interval) of SLO with respect to the bit-flip mutation rate  $c/N$ , for different instances (non-linearity  $K$ ) and local optimality threshold  $\epsilon$  (see legend). The problem size is  $N = 18$ .

### 4.3 Stochastic Fitness Landscape Analysis

**Iterated Local Search.** In this section, we illustrate the potential usefulness of a stochastic fitness landscape analysis for the design of search algorithms based on stochastic local search operators. We consider the Iterated Local Search (ILS) framework [12] as a case study. ILS aims at escaping from poor local optima based on a perturbation mechanism followed by a local search procedure. Whenever the local search falls into a local optimum, it is perturbed by means of random modifications to obtain a new (inferior) solution from which another local search round starts. This process is iterated until the computational budget is exhausted. In the following, we analyze the performances of ILS on NK-landscapes. In particular, we perform 30 independent ILS executions on a randomly generated instance with  $N = 18$  and  $K \in \{1, 2, 3\}$ . We set the perturbation rate to 0.3; that is, each bit is flipped with a rate of 0.3. The local search components considered within the ILS is a first-improvement hill-climbing algorithm, where at most  $\lambda$  solutions are produced at each step by means of a stochastic bit-flip mutation with a rate of  $c/N$ . If there is no strict improvement, the current solution is considered as a stochastic local optimum, and a perturbation is performed for further iterations. The maximum number of fitness evaluations for the ILS is set to  $10^4$ .

In Fig. 5, we report the relative deviation (to be minimized) from the best know solution of the final fitness value obtained by the ILS for different mutation rates,  $\lambda$ -values, and degrees of non-linearity. For any  $\lambda$ -value, there is a mutation rate that maximizes the expected ILS performance. Interestingly, in most of  $\lambda$  scenarios, this value corresponds, to the mutation rate that minimizes the





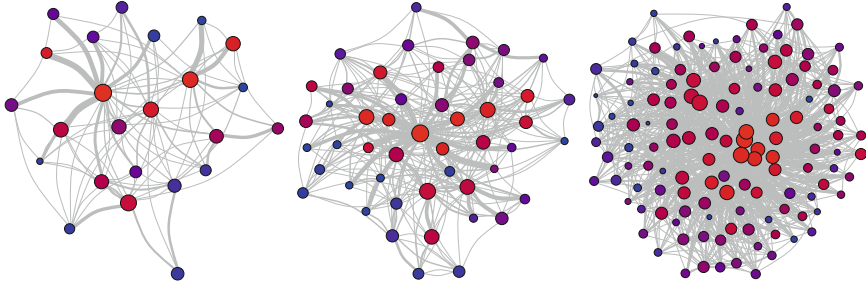
**Fig. 5.** Deviation of solution quality, from the best known solution, (average value and confidence interval) obtained by ILS with respect to the bit-flip mutation rate, for different instances (non-linearity  $K$ ) and sample sizes  $\lambda$ . The problem size is  $N = 18$ .

number of SLO, as disclosed above. This illustrates that analyzing the stochastic fitness landscape of the target problem might actually provide insightful information about the suitable configuration of stochastic local search algorithms.

**Stochastic Local Optima Network.** In order to complement our analysis of stochastic fitness landscapes, we now naturally extend the concept of Local Optima Network (LON) [4, 13] to stochastic local optima. We define the Stochastic LON (SLON) as a graph where nodes are *stochastic* local optima, and edges represent the pairwise connections between stochastic local optima with respect to another stochastic operator. More particularly, an edge  $(x_i, x_j)$  is defined and weighted so as to render the probability of reaching SLO  $x_j$  from SLO  $x_i$ . In this work, escape edges [19], as defined in Sect. 2, are computed according to a bit-flip mutation operator.

We generate a SLON for an NK-landscape with  $N = 18$  and  $K = 2$ . The SLO is defined by the bit-flip stochastic operator with three different mutation rates  $p \in \{1/N, 2/N, 3/N\}$ , and a stochastic operator for escape edges with a mutation rate of  $3/N$ . The sample size to estimate SLO and edges is set to  $\lambda = 4N$ . Figure 6 shows the obtained SLON for the same NK-landscape. The node colors indicate the fitness value: the redder, the better. The node size is logarithmically proportional to the size of the corresponding basin of attraction. The edge size is linearly proportional to its weight (self-loops are omitted to improve readability). We cannot report the SLON and related metrics for all instances due to space restriction. However, they are consistent with the visual impression of Fig. 6. For the considered sample size  $\lambda$ , the main observations are as follows: the number of SLO increases with the mutation rate, the density of edges decreases with the mutation rate, and so does the self-loop weights  $w_{ii}$ . The feature from standard (deterministic) LON that is the most correlated with the performance of ILS is known to be the average distance between local optima and the global optimum [13]. The distance between two

nodes  $i$  and  $j$  is defined as the inverse of the weight  $w_{ij}$ . Over all instances with  $N = 18$  and  $K = 2$ , the expected average distance to the global optimum is 8.48, 7.67, and 9.80, respectively, for mutation rates  $1/N$ ,  $2/N$ , and  $3/N$ . This metric suggests that the stochastic fitness landscape corresponding to a mutation rate of  $2/N$  is ‘easier’ to search than the one corresponding to a mutation rate of  $1/N$  or  $3/N$ . This observation is in line with the performance of ILS reported in Fig. 5. Indeed, the SLON for  $p = 1/N$  has a lower number of SLO, but nodes with higher fitness values seem to be clustered, which corresponds to a funnel structure [17]. By contrast, for  $p = 2/N$ , despite a larger number of SLO, paths to the global optimum seem to be shorter, and then more likely to happen during the search process of ILS. For  $p = 3/N$ , we infer that the huge number of SLO decreases the probability of reaching the global optimum by following a path on the network, thus inhibiting the performance of ILS.



**Fig. 6.** Stochastic local optima networks for an NK-landscape with  $N = 18$  and  $K = 2$ . From left to right, the mutation rate is:  $p = 1/N$ ,  $2/N$ , and  $3/N$ , respectively.

## 5 Discussion and Further Considerations

In this paper, we defined and analyzed fitness landscapes based on stochastic search operators. Based on this definition, we empirically investigated enumerable instances from NK-landscapes. More particularly, we studied the number of stochastic local optima as a preliminary feature of such stochastic fitness landscapes, showing the relevance of stochastic local optima when measuring the multimodality of stochastic fitness landscapes. We also studied the underlying stochastic local optima networks. We found out that there is a critical region in the stochastic operator setting (the mutation rate) in which stochastic local optima are more scarce. A proper setting within this region would make the computational effort to solve the problem much more effective and would result in better solution quality.

Let us emphasize that the proposed definition of local optimality and fitness landscape for stochastic operators is not entitled to any particular problem

class. Moreover, although we exhaustively enumerated the search space of NK-landscapes in order to avoid any bias in our current analysis, this is obviously not practical for large-scale optimization problems. As such, we plan to investigate sampling procedures that will enable studying the stochastic fitness landscape of other academic and real-world optimization problems. We hope that the proposed definition will enable analyzing better many recurrent problems and optimization algorithms, such as population-based evolutionary, estimation of distribution, or genetic programming algorithms.

**Acknowledgements.** We are very thankful to the CALCULCO center of Université du Littoral Côte d’Opale for providing computational resources used in this paper.




## References

1. Alyahya, K., Rowe, J.E.: Simple random sampling estimation of the number of local optima. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 932–941. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_87](https://doi.org/10.1007/978-3-319-45823-6_87)
2. Basseur, M., Goëffon, A.: Climbing combinatorial fitness landscapes. *Appl. Soft Comput.* **30**, 688–704 (2015)
3. Bosman, A.S., Engelbrecht, A., Helbig, M.: Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions. *Neurocomputing* (2020)
4. Chicano, F., Daolio, F., Ochoa, G., Vérel, S., Tomassini, M., Alba, E.: Local optima networks, landscape autocorrelation and heuristic search performance. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012. LNCS, vol. 7492, pp. 337–347. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32964-7\\_34](https://doi.org/10.1007/978-3-642-32964-7_34)
5. Elorza, A., Hernando, L., Mendiburu, A., Lozano, J.A.: Estimating attraction basin sizes of combinatorial optimization problems. *Progress in Artificial Intelligence* **7**(4), 369–384 (2018). <https://doi.org/10.1007/s13748-018-0156-6>
6. Fieldsend, J.E., Alyahya, K.: Visualising the landscape of multi-objective problems using local optima networks. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1421–1429 (2019)
7. Hernando, L., Mendiburu, A., Lozano, J.A.: An evaluation of methods for estimating the number of local optima in combinatorial optimization problems. *Evol. Comput.* **21**(4), 625–658 (2013)
8. Hernando, L., Mendiburu, A., Lozano, J.A.: Anatomy of the attraction basins: breaking with the intuition. *Evol. Comput.* **27**(3), 435–466 (2019)
9. Kauffman, S.A.: *The origins of order: Self-organization and selection in evolution*. OUP USA (1993)
10. Liefoghe, A., Daolio, F., Verel, S., Derbel, B., Aguirre, H., Tanaka, K.: Landscape-aware performance prediction for evolutionary multi-objective optimization. *IEEE Trans. Evol. Comput.* (2019, accepted)
11. Liefoghe, A., Derbel, B., Verel, S., López-Ibáñez, M., Aguirre, H., Tanaka, K.: On pareto local optimal solutions networks. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 232–244. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_19](https://doi.org/10.1007/978-3-319-99259-4_19)

12. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: framework and applications. Iterated local search: framework and Applications. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 146. Springer, Boston (2010). [https://doi.org/10.1007/978-1-4419-1665-5\\_12](https://doi.org/10.1007/978-1-4419-1665-5_12)
13. Ochoa, G., Verel, S., Daolio, F., Tomassini, M.: Local optima networks: a new model of combinatorial fitness landscapes. In: Richter, H., Engelbrecht, A. (eds.) Recent Advances in the Theory and Application of Fitness Landscapes. ECC, vol. 6, pp. 233–262. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-41888-4\\_9](https://doi.org/10.1007/978-3-642-41888-4_9)
14. Stadler, P.F.: Fitness landscapes. In: Lässig, M., Valleriani, A. (eds.) Biological Evolution and Statistical Physics. Lecture Notes in Physics, vol. 585, pp. 187–207. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45692-9\\_10](https://doi.org/10.1007/3-540-45692-9_10)
15. Tari, S., Basseur, M., Goëffon, A.: Worst improvement Based iterated local search. In: Liefvooghe, A., López-Ibañez, M. (eds.) EvoCOP 2018. LNCS, vol. 10782, pp. 50–66. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77449-7\\_4](https://doi.org/10.1007/978-3-319-77449-7_4)
16. Thierens, D.: An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 1539–1546 (2005)
17. Thomson, S.L., Daolio, F., Ochoa, G.: Comparing communities of optima with funnels in combinatorial fitness landscapes. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 377–384 (2017)
18. Thomson, S.L., Ochoa, G., Verel, S., Veerapen, N.: Inferring future landscapes: sampling the local optima level. In: Evolutionary Computation, pp. 1–22 (2020)
19. Vérel, S., Daolio, F., Ochoa, G., Tomassini, M.: Local optima networks with escape edges. In: Hao, J.-K., Legrand, P., Collet, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) EA 2011. LNCS, vol. 7401, pp. 49–60. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-35533-2\\_5](https://doi.org/10.1007/978-3-642-35533-2_5)
20. Verel, S., Ochoa, G., Tomassini, M.: Local optima networks of NK landscapes with neutrality. *IEEE Trans. Evol. Comput.* **15**(6), 783–797 (2011)
21. Weinberger, E.D.: Local properties of kauffman’s n-k model: a tunably rugged energy landscape. *Phys. Rev. A* **44**(10), 6399 (1991)
22. Wright, A.H., Thompson, R.K., Zhang, J.: The computational complexity of NK fitness functions. *IEEE Trans. Evol. Comput.* **4**(4), 373–379 (2000)
23. Wright, S.: The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In: Proceedings of the Sixth International Congress of Genetics, vol. 1, pp. 356–366 (1932)



# Fitness Landscape Analysis of Dimensionally-Aware Genetic Programming Featuring Feynman Equations

Marko Durasevic<sup>1</sup>, Domagoj Jakobovic<sup>1</sup>(✉) ,  
Marcella Scoczynski Ribeiro Martins<sup>2</sup> , Stjepan Picek<sup>3</sup>,  
and Markus Wagner<sup>4</sup> 

<sup>1</sup> Faculty of Electrical Engineering and Computing,  
University of Zagreb, Zagreb, Croatia

{marko.durasevic, domagoj.jakobovic}@fer.hr

<sup>2</sup> Federal University of Technology Paraná (UTFPR), Curitiba, Brazil  
marcella@utfpr.edu.br

<sup>3</sup> Delft University of Technology, Delft, The Netherlands  
s.picek@tudelft.nl

<sup>4</sup> Optimisation and Logistics Group, The University of Adelaide, Adelaide, Australia  
markus.wagner@adelaide.edu.au

**Abstract.** Genetic programming is an often-used technique for symbolic regression: finding symbolic expressions that match data from an unknown function. To make the symbolic regression more efficient, one can also use dimensionally-aware genetic programming that constrains the physical units of the equation. Nevertheless, there is no formal analysis of how much dimensionality awareness helps in the regression process. In this paper, we conduct a fitness landscape analysis of dimensionally-aware genetic programming search spaces on a subset of equations from Richard Feynman's well-known lectures. We define an initialisation procedure and an accompanying set of neighbourhood operators for conducting the local search within the physical unit constraints. Our experiments show that the added information about the variable dimensionality can efficiently guide the search algorithm. Still, further analysis of the differences between the dimensionally-aware and standard genetic programming landscapes is needed to help in the design of efficient evolutionary operators to be used in a dimensionally-aware regression.

**Keywords:** Genetic programming · Dimensionally-Aware GP · Fitness landscape · Local optima network

## 1 Introduction

Symbolic regression is a unique and very general type of multivariate regression analysis. In this analysis the task is to find the mathematical expression that links a number of variables in a domain with an unknown target function

that would fit a dataset  $S = \{(\mathbf{x}^{(i)}, y^{(i)})\}$ , i.e., a set of pairs of an unknown multivariate target function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ . With more than a quarter of a century of research in the field, the results obtained attracted the interests of many researchers to work in this area. A large number of applications of symbolic regression is both impressive, and it is also constantly expanding. For instance, symbolic regression has helped to extract physical laws using experimental data of chaotic dynamical systems without any knowledge of Newtonian mechanics [16]. Others have used it to design more efficient antennas [10] and to analyse satellite data [6]. Symbolic regression via Genetic Programming (GP) implementations has been used to model mechanisms of drug response in cancer cell lines using genomics and experimental data [4], to discover hidden relationships in astronomical datasets [7], to predict wind farm output from weather data [20], to generate computer game scenes [5], and for many other scenarios.

In some sense, Evolutionary Computation (EC) methods for symbolic regression (most commonly employing GP-based implementations) somewhat “compete” with other strategies like support vector regression and artificial neural networks. However, many researchers prefer to use symbolic regression since they tend to produce models with a significantly smaller number of variables, leading to solutions in a form amenable to downstream studies (e.g., uncertainty propagation and sensitivity analysis) and more “explainable” outcomes.

Although symbolic regression methods – and in particular GP-based methods – are popular, the research often does not use problem-domain information, and even commercial products like Eureqa [16] do not make use of it. With this paper, we propose to revisit the idea of Dimensionally-Aware Genetic Programming [9] and to analyse the impact of design decisions using modern fitness landscape analysis tools. To this end, we take a recent benchmark suite of symbolic regression problems [17], which also includes information about the dimensionality of input variables and the resulting model outputs. Taking this information into account, we devise and employ a deterministic local search algorithm which at all times satisfies the imposed dimensionality constraints. Using the local search, a complete network of local optima is built, considering given neighbourhood operators. After the local optima network (LON) is obtained, information from the search is used to infer characteristics of the underlying fitness landscape. At the same time, a comparison is made with the regular GP that does not restrict the dimensionality of the variables, to estimate the problem difficulty and the potential effectiveness of this approach.

## 2 Background

### 2.1 Feynman’s Equations

We will apply our methods to “rediscover” the fundamental physical laws. We consider equations from Feynman Lectures on Physics [3], covering topics like classical mechanics, electromagnetism, and quantum mechanics. Here, we follow the equation selection from Udrescu and Tegmark [17]. The authors listed 100 equations that do not contain derivatives or integrals and have between one

**Table 1.** Feynman equations considered in this article; the units column shows the number of different physical units of the corresponding variables.

ID	Feynman eq.	Equation	Variables	Units
1	I.8.14	$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	4	1
2	I.12.1	$F = \mu N_n$	2	3
3	I.12.2	$F = \frac{q_1 q_2}{4\pi\epsilon r^2}$	4	4
4	I.12.5	$F = q_2 E_f$	2	4
5	I.13.4	$K = \frac{1}{2}m(v^2 + u^2 + w^2)$	4	3
6	I.14.3	$U = mgz$	3	3
7	I.14.4	$U = \frac{k_{spring}x^2}{2}$	2	3
8	I.18.4	$r = \frac{m_1 r_1 + m_2 r_2}{m_1 + m_2}$	4	2
9	I.24.6	$E = \frac{1}{4}m(\omega^2 + \omega_0^2)x^2$	4	3
10	I.25.13	$V_e = \frac{q}{C}$	2	4
11	I.27.6	$f_f = \frac{1}{\frac{1}{d_1} + \frac{1}{d_2}}$	3	1
12	I.29.4	$k = \frac{\omega}{c}$	2	2
13	I.32.5	$P = \frac{q^2 a^2}{6\pi\epsilon c^3}$	4	4
14	I.34.8	$\omega = \frac{qvB}{p}$	4	4
15	I.39.1	$E_n = \frac{3}{2}pV$	2	3
16	I.39.22	$P_F = \frac{nk_b T}{V}$	4	4
17	I.43.16	$v = \frac{\mu q V_e}{d}$	4	4
18	I.43.31	$D = \mu_e k_b T$	3	4
19	II.2.42	$P = \frac{\kappa(T_2 - T_1)A}{d}$	5	4
20	II.8.31	$E_{den} = \frac{\epsilon E_f^2}{2}$	2	4
21	II.11.3	$x = \frac{qE_f}{m(\omega_0^2 - \omega^2)}$	5	4
22	II.15.4	$E = -\mu_M B \cos(\theta)$	3	4
23	II.34.2	$\mu_M = \frac{qvr}{2}$	3	4
24	II.34.29b	$E = \frac{g\mu_M B J_z}{h}$	5	4
25	II.38.3	$F = \frac{YAx}{d}$	4	3
26	III.13.18	$v = \frac{2Ed^2k}{h}$	4	3
27	III.15.14	$m = \frac{h^2}{2Ed^2}$	3	3

and nine independent variables. The same authors also provide the Feynman Symbolic Regression Database [18], where for each equation, there is a data table whose rows are of the form  $x_1, x_2, \dots, y$ , where  $y = f(x_1, x_2, \dots)$ . Table 1 contains the 27 equations that we consider in the present paper. This subset was selected to involve equations with a varying number of variables, different types of operators, varying degrees of complexity, and a different number of physical units. For the sake of readability, we will refer to these as the Feynman equations from now on.

## 2.2 Fitness Landscape Analysis

Fitness landscapes illustrate the correlation between the search and fitness space [12, 13, 15], and are commonly used to describe or predict the performance of a heuristic search. Fitness landscape analysis can help predict the performance of heuristics by using search cost models. Local Optima Network (LON) is a fitness landscape model proposed in [14] for combinatorial landscapes, considering that the number and distribution of local optima in a search space represents an important impact on the performance of heuristic search algorithms [2]. In this network model, the nodes are the local optima of a given optimisation problem, and the edges represent transitions among them using a neighbourhood operator [19]. Therefore, the fitness landscape is represented as a graph of connected local optima.

In general, a local search heuristic **LS** maps the solution space  $S$  to the set of locally optimal solutions  $S^*$ . A solution  $i$  in the solution space  $S$  is a local optimum given a neighbourhood operator  $\mathcal{N}$  if  $F(i) \geq F(s), \forall s \in \mathcal{N}(i)$ . Each local optima  $i$  has an associated basin of attraction corresponding to the set composed of all the solutions that, after applying the local search heuristic starting from each of them, the procedure returns  $i$ . Therefore, the basin of attraction associated to a local optima  $i$  is the set  $B_i = \{s \in S | \mathbf{LS}(s) = i\}$  whose size is the cardinality of  $B_i$ . In this paper, a connection (undirected) edge between two basins is created if at least one solution in one basin has a neighbour solution in the other basin, given a neighbourhood operator. This approach was also used in other works (e.g., [14, 22]).

## 3 Technical Details

### 3.1 Dimensionally-Aware Genetic Programming

The Dimensionally-Aware GP, first introduced by Keijzer and Babovic [9], can only be applied if there is information about the physical units of the model variables. In [18], the authors provide the unit table that specifies the physical units of the input and output variables for all Feynman equations. There are five different physical units appearing in all the equations: length [ $m$ ], time [ $s$ ], mass [ $kg$ ], temperature [ $K$ ], and potential [ $V$ ]. For every equation and each variable, the exact *unit signature* is given. For instance, a variable denoting the distance is expressed in meters, and the corresponding signature would be  $[1, 0, 0, 0, 0]$ ; a variable denoting acceleration is expressed in meters per second squared, and its signature can be presented with  $[1, -2, 0, 0, 0]$ . Using the same notation, the result of each equation will have a corresponding *target signature*. Following the dimensionally-aware paradigm, the local search algorithm we employ will always conform to the given target signature. In other words, at all times, we only consider those candidate expressions that result in the desired signature. Furthermore, when including the arithmetic operators in the expression, we follow the simple rules illustrated in Table 2: multiplication and division operators simply add or subtract the exponent values in the signature, while addition and



**Table 2.** Effect of operations.

Function	Operations dimensionality
Addition	$[v, w, x, y, z], [v, w, x, y, z] \rightarrow [v, w, x, y, z]$
Subtraction	$[v, w, x, y, z], [v, w, x, y, z] \rightarrow [v, w, x, y, z]$
Multiplication	$[v, w, x, y, z], [v, w, x, y, z] \rightarrow [v + v, w + w, x + x, y + y, z + z]$
Division	$[v, w, x, y, z], [v, w, x, y, z] \rightarrow [v - v, w - w, x - x, y - y, z - z]$

subtraction can only be applied to expressions with the commensurate signature, and the resulting signature remains unchanged.

### 3.2 Initialisation Procedure

The goal of the initialisation procedure is to generate expressions whose result conforms to the target unit signature. This is achieved by using all of the available variables and only multiplication and division operators. In such an expression (e.g.  $x^1y^{-2}z^0$ ), each variable can be represented only by its exponent, which is an integer value. In initialisation, we consider exponents in the range  $[-3, \dots, 3]$ ; if  $r$  is the cardinality of the range and if an equation has  $p$  variables, this makes  $r^p$  combinations to test. In the end, all combinations that yield the correct signature define the set of all possible initial solutions. For instance, if the available variables represent time  $t$  and distance  $d$ , and the target signature requires speed, the correct initial expressions would be  $(t^{-1}d^1)$ ,  $(t^{-2}d^2)$ , etc. Note, in the case where the chosen exponent range is not expressive enough to generate a single valid expression, the maximum exponent values can be increased and the initialisation simply restarted (this was not needed in our experiments).

### 3.3 Neighbourhood Operators

For our variation operators, we consider custom operators designed to be dimensionally-aware, i.e., their application does not change the signature of the overall expression encoded as a tree.

- **Replacement operator.** Select a subtree  $t$  with a signature  $s_t = [v, w, x, y, z]$  from the tree  $T$  and replace it with a subtree  $\hat{t}$  that has a commensurate signature, i.e.,  $s_t = s_{\hat{t}}$ .
- **Multiplication with integer.** Select a subtree  $t$  with a signature  $s_t = [v, w, x, y, z]$  from the tree  $T$  and replace it with a tree  $\hat{t}$  where the root is multiplication, one child is  $t$  and the other one is integer (dimensionless) in the range  $[-3, \dots, 3]$  (not dependent on the max exponent value). The signatures of  $t$  and  $\hat{t}$  are the same.
- **Divison with integer.** Same as the previous one, except the two subtrees are connected with the division operator.

- **Addition with a commensurate value.** Select a subtree  $t$  with a signature  $s_t = [v, w, x, y, z]$  from the tree  $T$  and replace it with a tree  $\hat{t}$  where the root is addition, one child is  $t$  and the other one is  $q$  that has the same signature as  $t$ , i.e.,  $s_t = s_q$ .
- **Subtraction with a commensurate value.** Same as the previous one, except the two subtrees are connected with the subtraction operator.

In all of the above operators, the new subtree is generated by following the same approach as in the initialisation procedure, enumerating all subtrees with the appropriate signature where the variable exponents are in the range  $[-3, \dots, 3]$ . This set of operators can produce expressions with only the four basic arithmetic operations; while executing the operations, the signatures of each subtree are updated according to the rules in Table 2. In the local search procedure, we use all the neighbourhood operators to generate all possible neighbours, and only the one with the best fitness measure is retained. This procedure is deterministic since it considers all possible variations and is in this regard similar to deterministic symbolic regression methods such as [11] and [21]. However, these approaches do not consider the dimensionality constraints as employed in the above operators. In the implementation, the maximum tree size is limited to 42 nodes, since with the repeated application of the same operator the expressions can bloat, i.e. achieve slightly smaller error values while the number of nodes becomes arbitrarily large.<sup>1</sup>

Since the Feynman equations also contain constants in multiplication or addition operations, we additionally employ the *linear scaling* technique [8]. With linear scaling, the original expression encoded as a tree  $T$  is evaluated as  $(a + b \cdot T)$ ; the coefficients  $a$  and  $b$  are determined by a simple linear regression where the sum of squared errors between the desired output and  $(a + b \cdot T)$  is minimised.

### 3.4 Local Search Procedure

The local search used in our study is described in Algorithm 1, where  $\mathcal{N}(\cdot)$  represents the neighbourhood of the given solution. The algorithm is deterministic; if there are multiple solutions with the same fitness value within the neighbourhood, the algorithm will retain the first one that it encounters. The local search is started using *all* initial solutions obtained with initialisation to generate a LON for every considered equation.

As the local search fitness measure, we use the mean squared error (*MSE*) of the expression; a strict improvement is required for a new solution to be accepted. The described local search with operators conforming to the dimensional constraints will be denoted as “DAGP” in the remainder of the text.

---

<sup>1</sup> We have experimented with a range of more open-ended bloat-control mechanism, e.g., lexicographic optimisation for fitness and size. However, we observed that even in our rather discrete setting, optimising I.8.14 or I.27.6 would result in trees of a size of over 256 nodes.

**Algorithm 1.** A greedy local search heuristic

---

```

1:  $s \leftarrow$  initial solution
2: while there is an improvement do
3:    $s^* = s$ 
4:   for each  $s^{**}$  in  $\mathcal{N}(s)$  do
5:     if  $F(s^{**}) > F(s^*)$  then
6:        $s^* \leftarrow s^{**}$ 
7:     end if
8:   end for
9:    $s = s^*$ 
10: end while

```

---

**3.5 Genetic Programming Regression**

Apart from the DAGP, we also applied a regular form of GP symbolic regression to the chosen set of equations. The purpose of these GP experiments is to estimate the problem difficulty regarding the number of variables and complex dimensionality relations among the variables. The GP regression is not concerned with physical units but is guided exclusively with the minimisation of *MSE* given the training data. In our experiments, the GP – which is based on the GP package ECF [1] – uses the same parameters for all considered equations, which are listed in Table 3. The selection scheme is simple: in each iteration  $k = 3$  individuals are selected at random, and the worst one is eliminated. The remaining two are recombined to produce one offspring, which is then mutated with given individual mutation probability and returned to the population; both the crossover and the mutation type are chosen randomly in each invocation.

**Table 3.** Genetic programming parameters.

Parameter	Value
Population size	500
Function set	$+$ , $-$ , $*$ , $/$ , sin, cos
Individual mutation rate	0.5
Tree max depth	6
Crossover operators	Subtree, one point, size fair, uniform, context preserved
Mutation operators	Subtree, hoist, node replace, permutation, shrink
Termination criteria	$10^5$ evaluations
Number of runs	50

**4 Results**

In our experiments, we are considering the selected 27 Feynman’s equations and apply the dimensionally-aware local search (DAGP) and a standard symbolic

regression GP. The number of data points for each equation was equal to 100, which were uniformly sampled from the available datasets [17]. The primary goal of DAGP is the exploration of the dimensionally-aware fitness landscape by building a corresponding LON for each equation. The second goal is an estimate of the effort needed to successfully navigate such a landscape, in comparison with the standard symbolic regression. In addition to the described DAGP configuration, we experimented with the following modifications: (a) reducing the integer constant range to  $[-2, \dots, 2]$  and  $[-1, \dots, 1]$ ; and (b) different operator ordering in local search (five permutations). Furthermore, both the GP and all DAGP configurations were tested with and without the linear scaling.

#### 4.1 Algorithm Efficiency

When considering the efficiency of the search, we define an acceptance criterion with the  $MSE < 10^{-9}$ , i.e., a solution is considered “correct” (a hit) if its  $MSE$  falls below this limit.

Table 4 shows the number of evaluations needed to find a correct solution, while a dash denotes no such solution was found. In the case of DAGP, these values are non-volatile since the local search procedure is deterministic. In the case of GP, the number of evaluations needed is just an estimate; GP is executed 50 times, which either terminate after 100 000 evaluations or when a correct solution is found. In case a solution is found in at least one run, the estimate is calculated as the total number of evaluations across all runs, divided by the number of successful runs (e.g., if each run was successful, this is equivalent to the average number of evaluations over all runs).

From the table, we can divide the equations into several groups; the first group are trivial problems, in which the dimensionally-aware approach needed very few evaluations to construct the correct solution. In most cases, this is because the unit constraints result with only a single initial solution with the correct target signature. The second group are the equations which are not trivial, but the DAGP can construct a correct solution using the local search operators and linear scaling. For all these, the number of evaluations needed is considerably smaller than the corresponding GP search.

Finally, the third group includes equations which were not reconstructed; in some cases, this is because they include operators we have not considered, such as square root (I.18.14) or trigonometric functions (II.15.4). The rest of those equations (I.13.4, I.18.4) also presented a challenge to the GP, since it was successful in a small number of runs requiring a large number of evaluations. For both GP and DAGP, linear scaling was beneficial and provided improvement of the model, regardless of the representation. It is also interesting to note that both DAGP modifications (a) and (b) made no difference in the number of equations whose solution was found, so we omit those results. As an illustration, we applied the DAGP local search and GP with scaling to 39 additional equations from the benchmark (the ones not including trigonometric functions); the DAGP was able to find a solution for 28 equations, whereas GP succeeded in 29 cases.

**Table 4.** Number of evaluations needed to obtain the optimum. A value in brackets denotes the number of successful GP runs, ‘–’ denotes unsuccessful run.

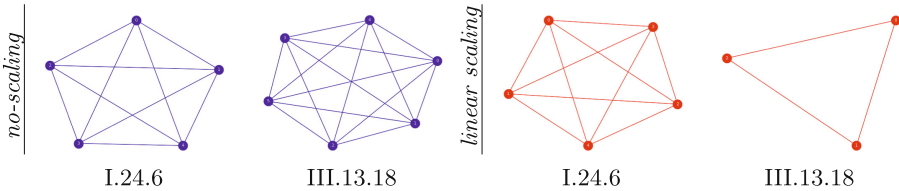
Eq. label	DAGP local search		GP	
	No scaling	Scaling	No scaling	Scaling
I.8.14	–	–	–	–
I.12.1	267	214	680 (50)	620 (50)
I.12.2	–	5	–	1 6750 (46)
I.12.5	1	1	580 (50)	580 (50)
I.13.4	–	–	–	2 464 750 (2)
I.14.3	1	1	2 060 (50)	2 000 (50)
I.14.4	–	1	908 400 (5)	1 740 (50)
I.18.4	–	–	675 785 (7)	1 613 300 (3)
I.24.6	–	2 086	–	2 425 250 (2)
I.25.13	1	1	960 (50)	780 (50)
I.27.6	72 575	2 817	223 735 (17)	740 500 (6)
I.29.4	1	1	950 (50)	840 (50)
I.32.5	–	1	–	33 370 (43)
I.34.8	1	1	20 076 (46)	4 620 (50)
I.39.1	–	1	1 574 500 (3)	560 (50)
I.39.22	517	408	15 904 (47)	4 800 (50)
I.43.16	1	1	21 488 (45)	6 260 (50)
I.43.31	1	1	2 080 (50)	2 110 (50)
II.2.42	19 468	29 556	98 450 (30)	22 500 (48)
II.8.31	–	1	1 155 625 (4)	1 760 (50)
II.11.3	1 000	2 042	4 921 500 (1)	940 000 (5)
II.15.4	–	–	43 397 (39)	3 750 (50)
II.34.2	–	1	1 161 875 (4)	1 820 (50)
II.34.29b	–	4 355	–	8 400 (50)
II.38.3	120	120	11 030 (49)	4 100 (50)
III.13.18	–	45	–	6 400 (50)
III.15.14	–	1	–	10 950 (48)

## 4.2 LON Characteristics for DAGP

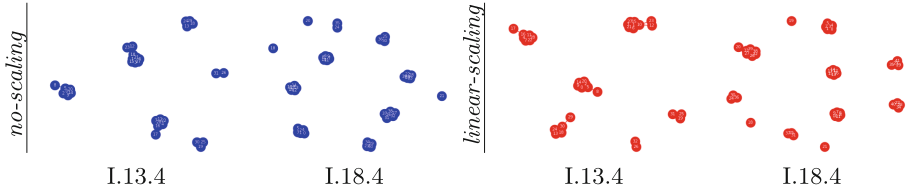
We expand the analysis extracting LONs from both DAGP landscapes, linear and no-scaling strategies. The obtained networks can be analysed according to some general graph metrics useful to understand the landscape behaviour. Table 5 reports the considered metrics:  $n_v$  and  $n_e$  represent the number of vertices (or nodes) and the number of edges of the generated LON, respectively.  $C$  is the average clustering coefficient which measures cliquishness of a

**Table 5.** Graph metrics for DAGP local search.

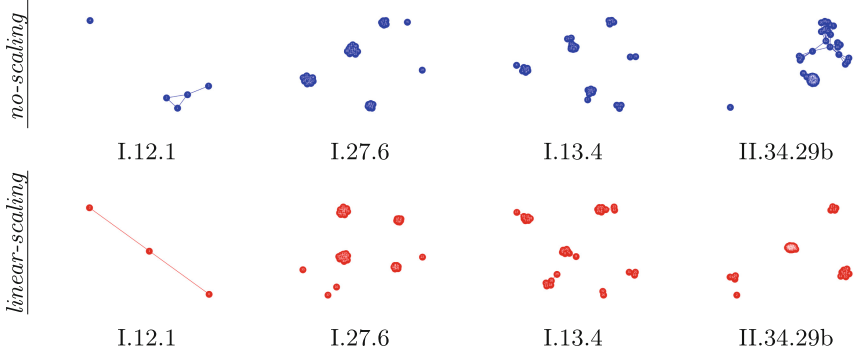
Equation	No-scaling								Linear-scaling							
	$n_v$	$n_e$	$\overline{C}$	$\overline{C}_r$	$\overline{l}$	$\pi$	$S$	$n_{hits}$	$n_v$	$n_e$	$\overline{C}$	$\overline{C}_r$	$\overline{l}$	$\pi$	$S$	$n_{hits}$
I.8.14	220	1641	0.85	0.07	-1.00	0	17	0	223	1805	0.87	0.07	-1.00	0	6	0
I.12.1	5	4	0.47	0.00	-1.00	0	2	5	3	2	0.00	0.00	1.33	1	1	3
I.12.2	5	6	0.80	0.53	-1.00	0	2	0	5	6	0.80	0.53	-1.00	0	2	5
I.12.5	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
I.13.4	32	66	0.84	0.15	-1.00	0	6	0	33	67	0.80	0.18	-1.00	0	6	0
I.14.3	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
I.14.4	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
I.18.4	41	73	0.77	0.03	-1.00	0	11	0	42	72	0.80	0.06	-1.00	0	11	0
I.24.6	5	10	1.00	1.00	1.00	1	1	0	5	10	1.00	1.00	1.00	1	1	4
I.27.6	39	100	0.61	0.09	-1.00	0	6	3	41	100	0.58	0.10	-1.00	0	8	25
I.29.4	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
I.32.5	1	0	0.00	0.00	0.00	1	1	0	1	0	0.00	0.00	0.00	1	1	1
I.34.8	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
I.39.1	1	0	0.00	0.00	0.00	1	1	0	1	0	0.00	0.00	0.00	1	1	1
I.25.13	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
I.39.22	6	6	1.00	0.44	-1.00	0	2	6	7	9	1.00	0.21	-1.00	0	2	7
I.43.16	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
I.43.31	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
II.2.42	24	57	0.96	0.14	-1.00	0	4	3	23	49	0.92	0.14	-1.00	0	4	2
II.8.31	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
II.11.3	4	0	0.00	0.00	-1.00	0	4	4	5	1	0.00	0.00	-1.00	0	4	3
II.15.4	7	6	0.86	0.21	-1.00	0	3	0	5	3	0.60	0.00	-1.00	0	3	0
II.34.2	1	0	0.00	0.00	0.00	1	1	1	1	0	0.00	0.00	0.00	1	1	1
II.38.3	13	10	0.64	0.00	-1.00	0	6	12	14	11	0.60	0.00	-1.00	0	6	14
II.34.29b	46	250	0.76	0.24	-1.00	0	3	0	39	238	0.87	0.33	-1.00	0	5	36
III.13.18	6	15	1.00	1.00	1.00	1	1	0	3	3	1.00	1.00	1.00	1	1	3
III.15.14	1	0	0.00	0.00	0.00	1	1	0	1	0	0.00	0.00	0.00	1	1	1

**Fig. 1.** LON examples of fully-connected networks using no-scaling (left-blue) and linear scaling (right-red) for I.24.6 and II.13.18. (Color figure online)

neighbourhood, and it characterises the degree to which nodes in a graph tend to cluster together;  $C_r$  is the average clustering coefficient of corresponding random graphs (i.e., random graphs with the same number of vertices and mean degree).  $l$  is the average shortest path length between any two local optima.  $\pi$  is the connectivity, which indicates if the LON is a connected graph with  $S$  being



**Fig. 2.** LON examples of dense local clusters using no-scaling (left-blue) and linear scaling (right-red) for I.13.4 and I.18.4. (Color figure online)

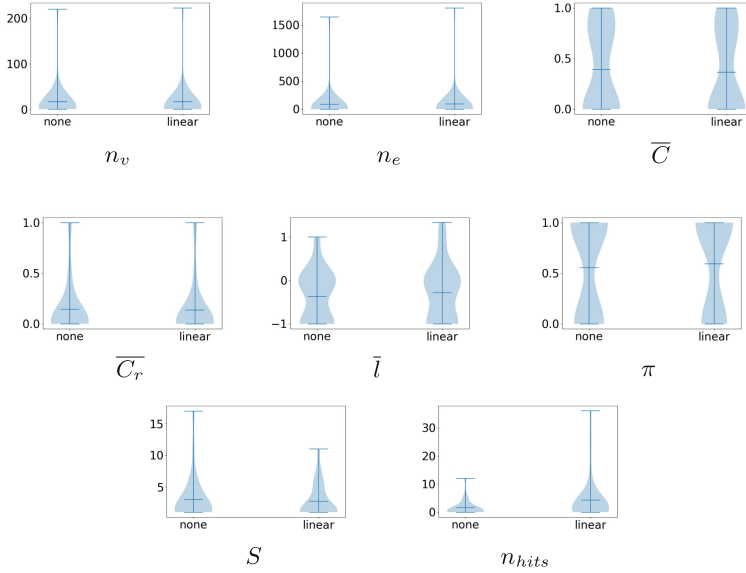


**Fig. 3.** LON using no-scaling (top-blue) and linear scaling (bottom-red) for particular equation examples with 2 (I.12.1), 3 (I.27.6), 4 (I.8.14) and 5 (II.34.29b) variables. (Color figure online)

the number of connected components (sub-graphs). Finally,  $n_{hits}$  is the number of nodes which represent a hit; as before, we consider a solution to be a hit if its mean square error is  $MSE < 10^{-9}$ . Some landscapes (13 of the 27 reported in Table 5) consist of only a single node. Within the non-scaling experiments, the optimum appears in seven of these 13 cases; for linear scaling, the optimum is found in all 13 landscapes with unique nodes.

Analysing the average shortest path lengths ( $\bar{l}$ ), some results show that the network is weakly and sometimes not connected ( $\bar{l} = -1$ ). Few reported cases present  $\bar{l} \geq 1$ , i.e., any pair of local optima can be connected by traversing at least other local optima, such as in I.24.6 and III.13.18  $\bar{l} = 1$ . Besides, in these examples,  $\pi = 1$  and  $S = 1$ , meaning the network is connected in one entire component (see Fig. 1 for examples).

We can also observe small-world properties by looking at the clustering coefficients ( $\bar{C}$ ,  $\bar{C}_r$ ) for some equations. Some LONs show a significantly high degree of local clustering compared with their corresponding random graphs, meaning that the local optima are connected in two ways: dense local clusters and sparse interconnections, which can be challenging to find and exploit (see examples in Fig. 2 for I.13.4 and I.18.4).



**Fig. 4.** Violin plots for each graph metric over all equations. The bar in the center represents the mean while the extremes denote upper and lower bounds.

In Fig. 3, we highlight particular LON examples with two (I.12.1), three (I.27.6), four (I.8.14), and five (II.34.29b) variables. Note that the  $\overline{C}$  coefficient is higher for linear scaling in II.34.29b in comparison with no-scaling. Moreover, I.12.1 and I.27.6 present  $n_{hits} > 0$ ; this also happens for II.34.29b but only considering linear scaling  $n_{hits} = 36$ .

Figure 4 summarises each metric considering all addressed equations for both cases no-scaling and linear scaling. We note that with few exceptions ( $\overline{l}$  and  $n_{hits}$ ), the metrics present similar distributions for both strategies. Since the two DAGP modifications (a) and (b) exhibit very similar behaviour, their graph metrics are not included.

## 5 Conclusions and Future Work

In many regression problems, only the raw data, obtained with the help of some measurements, is available to infer the governing model. It is not often the case that the information about the physical units of the result and the variables are documented; however, if this information is available, it can significantly improve regression to the extent that some problems become trivial to solve with the right approach.

Our experiments on a subset of equations of Richard Feynman’s have shown that a very simple local search procedure, adhering to the dimensionally-aware constraints, can efficiently navigate the corresponding landscape and arrive at the correct solution. However, it must be noted that in real-world situations, a



certain amount of noise in the data can be expected, which was not present in this study.

We also have extracted Local Optima Networks (LONs) providing a fitness landscape analysis for the dimensionally-aware genetic programming search space. The networks presented small-world properties for some equations meaning that the local optima can be connected as dense local clusters but also in sparse interconnections – and sparse interconnections might make the search process harder even using strategies such as linear scaling.

We plan to extend the dimensionally-aware local search to cover additional operators such as square root, exponential and trigonometric functions. Besides local search, experiments can be performed by incorporating DA constraints into the standard GP, with appropriate mutation and crossover operators, where different fitness landscape models can be applied. At the same time, a transition from the regular GP and DAGP could be achieved with the use of maximum deviation to the target signature, which could be gradually decreased over the course of the evolution.

**Acknowledgements.** We'd like to thank Prof. Pablo Moscato for introducing us to [17]. We'd also like to acknowledge support by the Australian Research Council, project DP200102364.

## References

1. Evolutionary computation framework (2019). <http://ecf.zemris.fer.hr/>
2. Daolio, F., Verel, S., Ochoa, G., Tomassini, M.: Local optima networks and the performance of iterated local search. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 369–376. ACM (2012)
3. Feynman, R.P., Leighton, R.B., Sands, M.: The Feynman Lectures on Physics, New millennium edn. Basic Books, New York (2010). <https://cds.cern.ch/record/1494701>. Originally published 1963–1965
4. Fitzsimmons, J., Moscato, P.: Symbolic regression modelling of drug responses. In: First IEEE Conference on Artificial Intelligence for Industries (2018)
5. Frade, M., de Vega, F.F., Cotta, C.: Breeding terrains with genetic terrain programming: the evolution of terrain generators. *Comput. Games Technol.* **2009**, 125714:1–125714:13 (2009)
6. Graham, M.J., Djorgovski, S.G., Mahabal, A., Donalek, C., Drake, A., Longo, G.: Data challenges of time domain astronomy. *Distrib. Parallel Databases* **30**(5), 371–384 (2012)
7. Graham, M., Djorgovski, S., Mahabal, A., Donalek, C., Drake, A.: Machine-assisted discovery of relationships in astronomy. *Mon. Not. R. Astron. Soc.* **431**(3), 2371–2384 (2013)
8. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 70–82. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36599-0\\_7](https://doi.org/10.1007/3-540-36599-0_7)
9. Keijzer, M., Babovic, V.: Dimensionally aware genetic programming. In: 1st Annual Conference on Genetic and Evolutionary Computation (GECCO), vol. 2, pp. 1069–1076. Morgan Kaufmann Publishers Inc. (1999)

10. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge (1992)
11. McConaghy, T.: FFX: fast, scalable, deterministic symbolic regression technology. In: Riolo, R., Vladislavleva, E., Moore, J. (eds.) *Genetic Programming Theory and Practice IX*. GEVO, pp. 235–260. Springer, New York (2011). [https://doi.org/10.1007/978-1-4614-1770-5\\_13](https://doi.org/10.1007/978-1-4614-1770-5_13)
12. Moscato, P.: An introduction to population approaches for optimization and hierarchical objective functions: a discussion on the role of tabu search. *Ann. Oper. Res.* **41**(2), 85–121 (1993)
13. Moscato, P., Fontanari, J.: Stochastic versus deterministic update in simulated annealing. *Phys. Lett. A* **146**(4), 204–208 (1990)
14. Ochoa, G., Tomassini, M., Vérel, S., Darabos, C.: A study of NK landscapes' basins and local optima networks. In: *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 555–562. ACM (2008)
15. Richter, H., Engelbrecht, A.: *Recent Advances in the Theory and Application of Fitness Landscapes*. Springer, Heidelberg (2014). <https://doi.org/10.1007/978-3-642-41888-4>
16. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. *Science* **324**(5923), 81–85 (2009)
17. Udrescu, S.M., Tegmark, M.: *Ai Feynman: a physics-inspired method for symbolic regression* (2019)
18. Udrescu, S.M., Tegmark, M.: *The Feynman database for symbolic regression* (2020). <https://space.mit.edu/home/tegmark/aifeynman.html>. Accessed 31 Jan 2020
19. Verel, S., Daolio, F., Ochoa, G., Tomassini, M.: Sampling local optima networks of large combinatorial search spaces: the QAP case. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) *PPSN 2018*. LNCS, vol. 11102, pp. 257–268. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_21](https://doi.org/10.1007/978-3-319-99259-4_21)
20. Vladislavleva, E., Friedrich, T., Neumann, F., Wagner, M.: Predicting the energy output of wind farms based on weather data: important variables and their correlation. *Renew. Energy* **50**, 236–243 (2013)
21. Worm, T., Chiu, K.: Prioritized grammar enumeration: symbolic regression by dynamic programming, pp. 1021–1028, July 2013. <https://doi.org/10.1145/2463372.2463486>
22. Yafrani, M.E., et al.: A fitness landscape analysis of the travelling thief problem. In: *Genetic and Evolutionary Computation Conference (GECCO)*, pp. 277–284 (2018)



# Global Landscape Structure and the Random MAX-SAT Phase Transition

Gabriela Ochoa<sup>1</sup>, Francisco Chicano<sup>2(✉)</sup>, and Marco Tomassini<sup>3</sup>

<sup>1</sup> Computing Science and Mathematics, University of Stirling, Stirling, Scotland, UK

<sup>2</sup> ITIS Software, University of Malaga, Malaga, Spain

`chicano@lcc.uma.es`

<sup>3</sup> Faculty of Business and Economics, Information Systems Department, University of Lausanne, Lausanne, Switzerland

**Abstract.** We revisit the fitness landscape structure of random MAX-SAT instances, and address the question: *what structural features change when we go from easy underconstrained instances to hard overconstrained ones?* Some standard techniques such as autocorrelation analysis fail to explain what makes instances hard to solve for stochastic local search algorithms, indicating that deeper landscape features are required to explain the observed performance differences. We address this question by means of local optima network (LON) analysis and visualisation. Our results reveal that the number, size, and, most importantly, the connectivity pattern of local and global optima change significantly over the easy-hard transition. Our empirical results suggests that the landscape of hard MAX-SAT instances may feature sub-optimal funnels, that is, clusters of sub-optimal solutions where stochastic local search methods can get trapped.

## 1 Introduction

Understanding when a specific class of problems go from being computationally easy to hard, remains a fundamental question. It is well-known that random instances of satisfiability problems exhibit a dramatic easy-to-hard phase transition with respect to the problem constrainedness [1–3]. Some standard fitness landscape analysis techniques such as the correlation structure fall short in explaining the performance differences of local search algorithms when solving easy and hard instances [4]. Studies of the configuration landscape of a set of local optima show that there exist big-valley structures (also called clusters) in the landscapes of 3-SAT and MAX-3SAT [5], and studies of the size and characteristics of local optima and plateaus [6] do offer interesting insight and help to explain performance differences. These studies, however, do not convey a view of the connectivity structure of local optima in MAX-SAT, as seen by stochastic local search algorithms.

Our motivating research question is: *what fitness landscape features change when we go from easy underconstrained instances to hard overconstrained ones?*

Another motivation for our study is the lack of tools for visualising high-dimensional fitness landscapes, specially in the presence of high levels of neutrality as is known to be the case for MAX-SAT. Neutrality is present in a landscape when neighbouring points have equal fitness values. A discrete landscape is regarded as neutral if a substantial fraction of adjacent pairs of solutions are neutral [7].

We conduct a detailed experimental investigation of the phase transition from underconstrained to overconstrained randomly generated instances of MAX-3SAT problems. We explore this phenomenon by means of local optima networks (LONs) [8,9] analysis and visualisation. In particular, we use the compressed LON model (CLON) [10], which allows us to deal with the high levels of neutrality observed in MAX-SAT. Our contribution is to offer a new set of fitness landscape features and visualisation tools reflecting the number, size, and, most importantly, the connectivity pattern of optima, which capture and help to explain why the computational effort of stochastic local search methods increases dramatically in the phase transition region and beyond.

## 2 SAT, MAX-SAT and the Phase Transition

The propositional satisfiability problem (SAT) is a prominent combinatorial decision problem with a central role in several areas of computer science. Given a Boolean formula, SAT checks if there is an assignment of variables to Boolean values such that the formula is satisfiable. The Boolean formula is commonly expressed as a conjunction of clauses (Conjunctive Normal Form). A clause is a list of literals (a Boolean variable or its negation) that is satisfied if at least one literal is true. The Boolean formula is satisfiable if all the clauses are true. MAX-SAT is the optimisation version of SAT, where the goal is to find an assignment that maximises the number of satisfied clauses.

It is well known [1–3] that random instances of the kSAT problem display a computational phase transition for a certain critical value  $\alpha_c$  of the ratio  $\alpha = m/n$  between the number of clauses  $m$  and the number of variables  $n$ . For fixed integers  $k$  and  $n$ , the probability that a randomly generated formula is satisfiable is a decreasing function of  $\alpha$ . For  $\alpha \rightarrow 0$  the probability that the formula is satisfiable goes to 1; it goes to 0 for  $\alpha \rightarrow \infty$ . Thus, a random formula is satisfiable for  $\alpha < \alpha_c$  and it is unsatisfiable for  $\alpha > \alpha_c$ , with probability approaching 1 as  $n \rightarrow \infty$ . The crossover from high to low probability becomes sharper as  $n$  increases and there is a transition at a finite value of  $\alpha_c$  which, for 3SAT, is  $\alpha_c \approx 4.3$ . For a given  $n$ , at low  $\alpha$  the problems are easy but in the phase transition region they become hard and the computational effort has a peak at the SAT/UNSAT boundary.

Zhang [11] investigated the relationship between the phase transitions of 3-SAT and MAX-3SAT, and found a near linear correlation between these two phase transitions. The computational cost of MAX-3SAT envelops the computational cost peaks of 3-SAT. It is worth noticing that in decision problems there is an easy-hard-easy phase transition, while in optimisation problems the

transition is easy-hard. That is, the computational cost remains high after the transition.

### 3 Local Optima Networks

The local optima network (LON) model [8] is a tool to capture the global structure of fitness landscape as seen by stochastic local search algorithms. In this paper, we use a coarse variant of LONs, the compressed LONs (CLONs) [10], which helps us to model the structure of landscapes with high amounts of neutrality. We describe below the LON model, before introducing the compressed model (CLON).

#### 3.1 LON Model

A fitness landscape [12] is a triplet  $(S, N, f)$  where  $S$  is a set of potential solutions i.e., a search space,  $N : S \rightarrow 2^S$ , is a neighbourhood structure, a function that assigns a set of neighbours  $N(s)$  to every solution  $s \in S$ , and  $f : S \rightarrow \mathbb{R}$  is a fitness function that can be pictured as the *height* of the corresponding solutions.

In our study, the search space is  $\{0, 1\}^n$ , i.e., the space of binary strings of length  $n$ , so its size is  $2^n$ . As neighbourhood, we consider the standard Hamming distance 1 neighbourhood, that is, the set of all solutions at a maximum Hamming distance of 1 from the current solution.

**Local Optima.** A local optimum, which in MAX-SAT is a maximum, is a solution  $l$  such that  $\forall s \in N(l), f(l) \geq f(s)$ . Notice that the inequality is not strict, in order to allow the treatment of neutrality (local optima of equal fitness), which is known to widely occur on MAX-SAT. The set of local optima, which corresponds to the set of nodes in the network model, is denoted by  $L$ .

**Perturbation Edges.** Edges are directed and based on the perturbation operator ( $p$  bit flips). There is an edge from local optimum  $l_1$  to local optimum  $l_2$ , if  $l_2$  can be obtained after applying a random perturbation ( $p$  bit flips) to  $l_1$  followed by hill climbing. Edges are weighted with estimated frequencies of transition. We determined the edge weights in a sampling process. The weight is the number of times a transition between two local optima occurred. The set of edges is denoted by  $E$ .

**LON.** Is the directed and weighted graph  $\text{LON} = (L, E)$ , where nodes are the local optima  $L$ , and edges the perturbation edges  $E$ .

#### 3.2 Compressed LON Model

When the number of local optima is high in a LON it is difficult to visualise the structure of the landscape. A natural way of reducing the model size in landscapes with high-levels of neutrality is to redefine the nodes of the model. The compressed LON model joins the local optima that are connected and have the same fitness value.

**Compressed Local Optima.** A compressed local optimum is a set of connected nodes (a connected component) in the LON with the same fitness value. This set of compressed optima, denoted by  $CL$ , corresponds to the set of nodes in the Compressed LON model.

**Compressed Perturbation Edges.** The set of perturbation edges is defined as above for the LON model. However, after compression, there are no edges between nodes with the same fitness, as connected components with the same fitness become a single node. The set of edges among compressed nodes are also aggregated and their weights summed. We call this set compressed edges,  $CE$ .

**Compressed LON.** Is the directed graph  $CLON = (CL, CE)$ , where nodes are the compressed local optima  $CL$  and edges the compressed perturbation edge set  $CE$ .

## 4 Methodology

### 4.1 Benchmark Instances

For the computational experiments, we used unweighted MAX3-SAT instances from the well-studied Uniform Random 3-SAT distribution [2]. Our instances have a relatively low number  $n = 50$  of variables to allow us to compute the global optimum in all of them. Instances with  $n = 50$  are not a challenge for state-of-the-art MAX-SAT solvers. However,  $n = 50$  is large enough to well capture the performance transition region [2,3], thus suitable for our study. Our goal is to gain a deeper understanding of the fitness landscape structure. Since we are interested in studying instances around the phase transition, we generated random instances with  $\alpha \in [3.0, 5.0]$  in steps of 0.2. For each value of  $\alpha$  we generated 10 instances with different random seeds. The source code of the instance generator and the instructions to replicate the experiments are available at: <https://github.com/jfrchicanog/EfficientHillClimbers>.

---

#### Algorithm 1. Iterated Local Search

---

```

1:  $x \leftarrow \text{generateRandomSolution}()$ ;
2:  $x \leftarrow \text{applyLocalSearch}(x)$ ;
3: while not stopping condition do
4:    $y \leftarrow \text{perturb}(x)$ ;
5:    $y \leftarrow \text{applyLocalSearch}(y)$ ;
6:    $\text{reportEdge}(x, y)$ ;
7:   if  $f(y) > f(x)$  then
8:      $x \leftarrow y$ ;
9:   end if
10: end while
11: return  $x$ ;

```

---

## 4.2 Sampling and Construction of the Network Models

To construct the network models for a given instance, we aggregate the (unique) local optima and transition edges obtained by 20 runs of a fast iterated local search (ILS) algorithm [13] that incorporates Grey-Box optimisation techniques [14]. An outline can be found in Algorithm 1. Iterated local search is a simple yet powerful metaheuristic that combines two steps: one for reaching local optima efficiently, and the other for escaping local optima (known as the perturbation step). The stopping condition was set as a fixed running time (60 s). Weights are added to edges indicating the number of times they appear in the sampling process. In our ILS implementation, the perturbation step flips 10% of the variables selected at random (which corresponds to 5 bit flips for  $n = 50$ ). The local search operator is a first improvement local search applied in the 1-flip neighbourhood. That is, if a flip in one bit of the current solution increases the number of satisfied clauses, the current solution is replaced by the new one. The local search is applied until a local optimum is reached (no neighbour can increase the number of satisfied clauses). A new local optimum is only accepted in Line 7 if it improves the incumbent solution. However, we report all the edges encountered between local optima in Line 6, which includes neutral and worsening edges. All the local optima (and edges) visited after finding the global optimum are removed from the LON. The reason is that they would bias the metrics, since they generate only worsening edges and they do not provide additional information on the difficulty of the search process.

## 4.3 Determining the Global Optimum

For all the instances, we computed the global optimum using exact methods. Instead of using an exhaustive enumeration (which could take a long time) we used `minisat`<sup>1</sup> to prove that there is no better solution than the one provided by ILS. The approach to prove optimality changes depending on the solution provided by ILS:

- If ILS finds an assignment satisfying all the clauses, the formula is satisfiable and the global optimum is found.
- If the optimal assignment found by ILS leaves one single clause unsatisfied, we apply `minisat` to check if the formula is satisfiable. If it is unsatisfiable, then the global optimum was found by ILS, otherwise (formula satisfiable) ILS didn't find a global optimum.
- If the optimal assignment found by ILS leaves  $u$  clauses unsatisfied, then we generate all the MAX-3SAT instances derived from the original instance where exactly  $u - 1$  different clauses are removed and apply `minisat` to all of them. If `minisat` finds all of them unsatisfiable, then ILS found the global optimum. If `minisat` finds at least one of them satisfiable, then a better assignment can be found with  $u - 1$  unsatisfied clauses at most, which means that ILS didn't find the global optimum.

---

<sup>1</sup> <http://minisat.se>.

The previous procedure to figure out the global optimum can be costly when the number of unsatisfied clauses in the optimal assignment is high. We expected this to happen when  $\alpha$  is high. However, in the range of values of  $\alpha$  used in the experiments the maximum number of unsatisfied clauses was 2, which made it possible to apply minisat in the way described above to certify that the global optimum was reached.

## 5 Results

### 5.1 Performance and Network Metrics

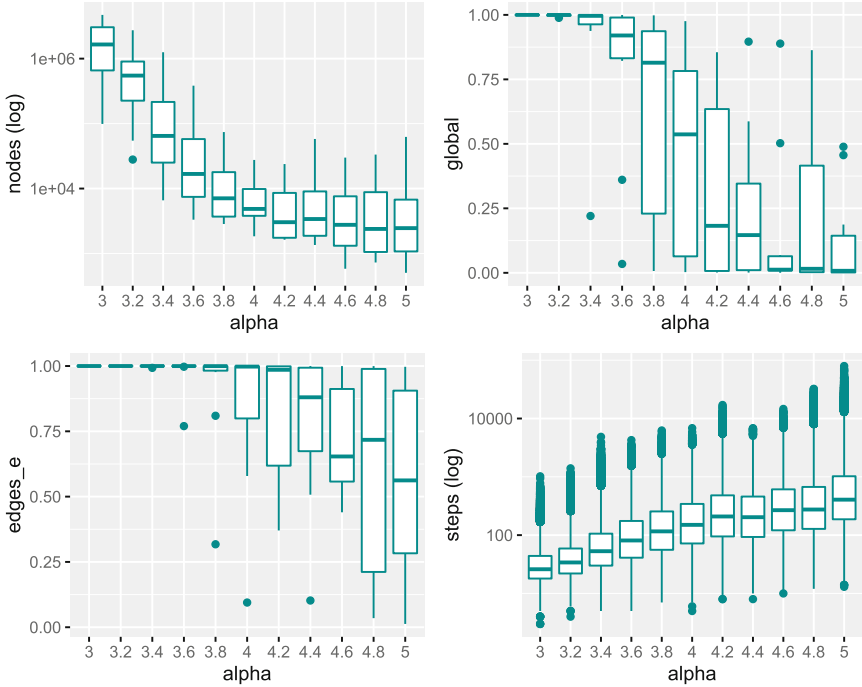
Our aim is to identify fitness landscape features that correlate with and help to explain the search difficulty of stochastic search algorithms. In order to measure search difficulty, we selected WalkSAT, a well-known local search algorithm for SAT and MAX-SAT [15]. WalkSAT has a wide influence among modern local search algorithms and is known to be very efficient in solving random 3-SAT and MAX-3SAT instances. We ran WalkSAT 10 000 times per instance and counted the number of bit flips (steps) it needs to find a global optimum. We have 10 instances per  $\alpha$  value, thus a total of 100 000 measures per  $\alpha$ . The distribution of this metric is shown in Fig. 1 (steps) with log scale, indicating a large variability, but a clear exponential increase in the search cost with increasing  $\alpha$ .

**Table 1.** Description of metrics.

Performance metric	
<i>steps</i>	Number of bit flips (steps) by WalkSAT before reaching the global optimum
LON metrics	
<i>nodes</i>	Number of nodes (local and global optima)
<i>global</i>	Proportion of nodes that are global optima
<i>edges<sub>e</sub></i>	Proportion of edges between nodes with equal fitness
CLON metrics	
<i>cnodes</i>	Number of compressed nodes
<i>subopt-size</i>	Size of the largest suboptimal compressed node
<i>path-length</i>	Average length of the shortest paths from start nodes to the global optimum
<i>cedges<sub>w</sub></i>	Proportion of edges to compressed nodes with worse fitness



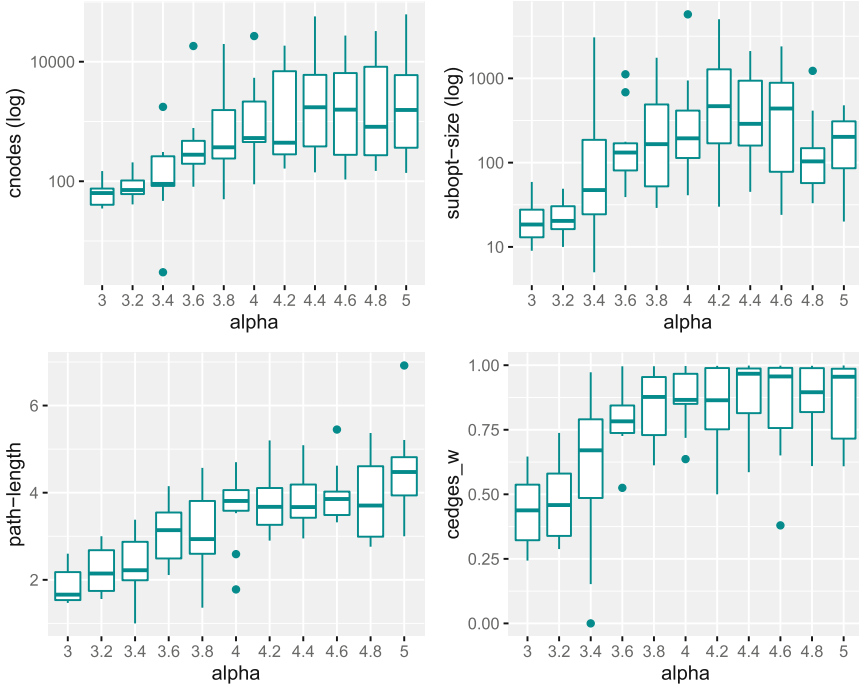
For each of the 10 instances per  $\alpha$  value, we extracted the LON and CLON models and computed the measurements described in Table 1. Network metrics are shown as the distribution of values over the 10 generated instances (Figs. 1 and 2); a large variability is observed across the 10 instances, meaning that different instances will have different structure and performance, this is known to be the case for randomly generated instances. Results are shown as box-plots with instances grouped by the value of  $\alpha$  in order to analyse the variation in the metrics as the value of  $\alpha$  changes.



**Fig. 1.** Distribution of the LON metrics and the performance metric, as described in Table 1, for all values of  $\alpha$ .

As Fig. 1 indicates, the number of nodes (unique local and global optima) visited decreases exponentially with increasing  $\alpha$ . For low values of  $\alpha$ , over one million nodes are visited. This number drops to a few thousands and even several hundreds for high values of  $\alpha$ . The proportion of global optima (*global* in Fig. 1) reveals that for  $\alpha < 4.0$ , the vast majority of nodes (above 75%) are global optima, and this proportion drastically decreases for larger values of  $\alpha$ . This is a known result [1]; in the underconstrained region instances are satisfiable and the density of solutions is high, thus making it relatively easy to find a global optimum. For high values of  $\alpha$ , instances are overconstrained, making it unlikely to find a solution (when the formula is satisfiable), or an assignment

with the maximum possible number of true clauses (for unsatisfiable formulas). Our network analysis complements this finding by revealing that most of the search transitions in the underconstrained region are among candidate solutions with equal fitness (traversing plateaus). This is revealed by the high proportion of equal edges ( $edges_e$  in Fig. 1), which is almost 100% for  $\alpha < 4.0$ . Our network models capture search transitions that are either improving, deteriorating or neutral. The proportion of deteriorating transitions also correlates with search difficulty as is revealed by our analysis of the compressed LONs (Fig. 2).



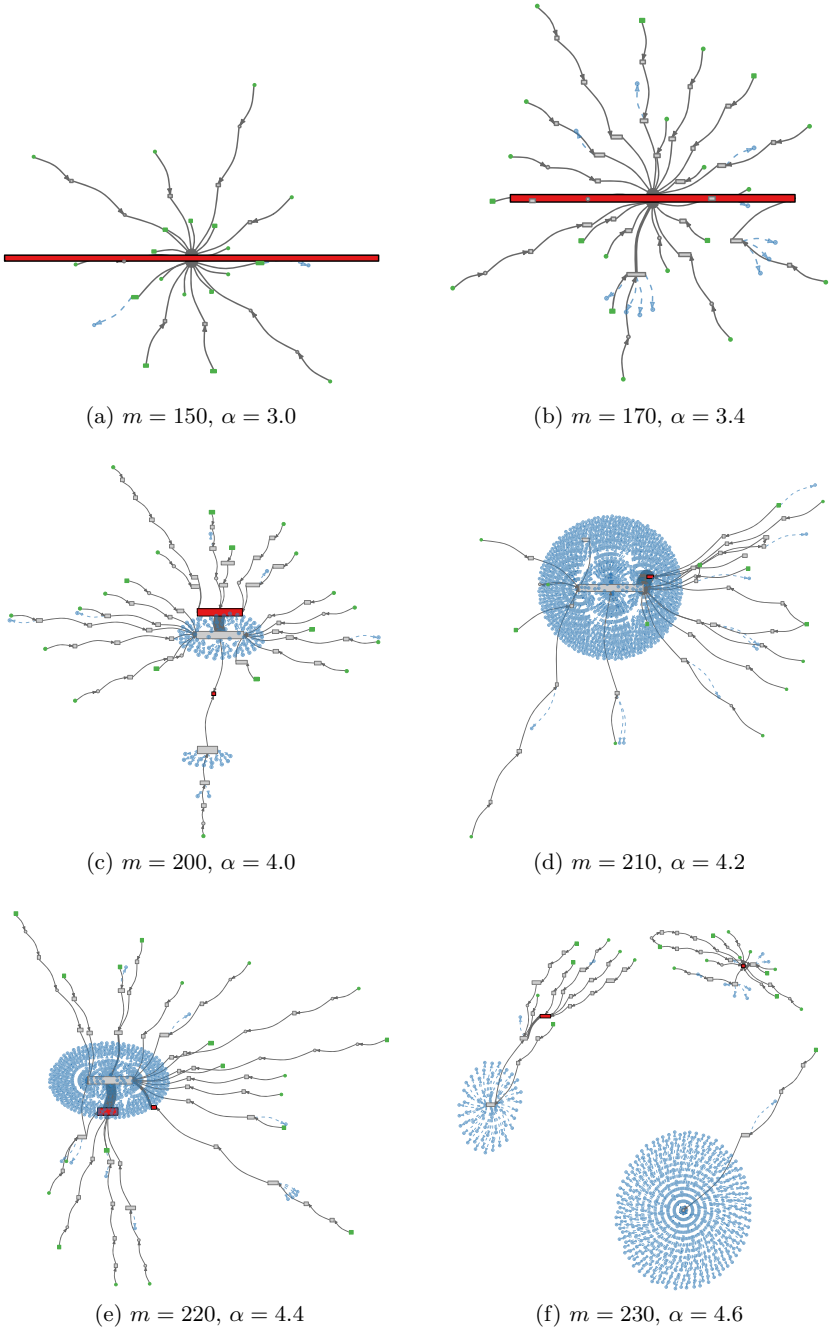
**Fig. 2.** Distribution of the CLON metrics, as described in Table 1, for all values of  $\alpha$ .

Figure 2 shows the metrics distribution for the compressed LON models. Compressed nodes aggregate connected local optima with the same fitness. The number of compressed nodes ( $cnodes$  in Fig. 2) shows an opposite tendency than the number of nodes ( $nodes$  in Fig. 1), that is, the number of compressed nodes increases exponentially with increasing  $\alpha$ . This is related to the proportion of equal edges in the LON model ( $edges_e$  in Fig. 1); if there are many equal edges, there will be many local optima per compressed node, reducing their number while increasing their size. This means that, for low  $\alpha$  values, CLONs will have a few nodes, indicating an easy to traverse fitness landscape. We can expect in this case a big node in the CLON capturing all the global optima,

this is what our network visualisations reveal (Figs. 3a, and b). The average path lengths towards the global optimum are also correlated with search difficulty (*path-length* in Fig. 2); shorter paths (of 1 to 4 hops) are observed in the underconstrained region. The size of the largest sub-optimal compressed node (*subopt-size* in Fig. 2) increases exponentially with increasing  $\alpha$ , which helps to explain the increased search cost in the overconstrained region, search processes spend time traversing sub-optimal plateaus. The CLON models only have improving and deteriorating edges, as the neutral edges are collapsed in the compressed nodes. When looking at the proportion of worsening edges (*edges<sub>w</sub>* in Fig. 2), we can observe that it increases noticeably in the overconstrained region; for  $\alpha > 4.0$ , over 80% of the search transitions are deteriorating, indicating that search processes spend a long time finding exits from sub-optimal compressed nodes. These observations are supported by our network visualisations (Figs. 3c–f), where larger sub-optimal nodes, higher proportion of deteriorating (blue) edges, and longer path lengths to the global optima can be observed.

## 5.2 Visualisation

The network visualisations in Fig. 3 capture the compressed local optima networks (CLONs) on representative instances with different values of  $\alpha$ . Plots were produced with the R statistics language and the *igraph* library [16]. Network plots are decorated to reflect features relevant to search dynamic. Single optima are visualised as circles of fixed size. When more than one local optimum is compressed in a local optimum plateau, the node is represented as a rectangle with length proportional to its size (ie. number of local optimum configurations). Red nodes are global optima, green nodes indicate the start configuration of trajectories in the sampling process; recall that our implementation combines 20 independent ILS runs to construct the networks; thus up to 20 green starting nodes can be visualised in the figures. Grey identifies the intermediate nodes, with grey edges representing improving transitions. Blue edges indicate deteriorating transitions, with blue nodes indicating the end of these transitions. The edges width is proportional to the sampled frequency of transitions, thus thick edges represent common search paths. In order to have manageable images, only 5% of the worsening transitions are shown. That is, the networks are pruned before visualisation by removing 95% of the deteriorating edges selected uniformly at random. The CLON visualisations in Figure 3 support what the metrics indicate. Instances get harder to search as  $\alpha$  increases, because more intermediate nodes appear and the trajectories get longer. The size of the global optimum node decreases drastically with increasing  $\alpha$ . For low values of  $\alpha$  (plots Fig. 3a, and b) a large “central” global optimum node is observed, which attracts all the search trajectories. As  $\alpha$  increases, (plots 3c–f), the size of the global optimum decreases, and additional disconnected global optima may appear. Clearly, with increasing  $\alpha$ , large sub-optimal compressed nodes emerge, and the proportion of deteriorating edges (visualised in blue) is larger. This indicates that the search process gets trapped in large sub-optimal nodes, requiring several escape attempts before finding an exit towards the global optimum. With increasing  $\alpha$ ,



**Fig. 3.** CLONs for representative benchmark instances. The global optimum (optima) are indicated in red, while the start nodes in green. Edge widths are proportional to their weight. Blue dashed edges indicate deteriorating transitions. (Color figure online)

the path lengths of the trajectories to the global optimum (optima), measured as the number of edges they contain, tends to increase.

The term ‘funnel’ was introduced in the protein folding community to describe “a region of configuration space that can be described in terms of a set of downhill pathways that converge on a single low-energy structure or a set of closely-related low-energy structures.” [17]. It has been suggested that the energy landscape of proteins is characterised by a single deep funnel, a feature that underpins their ability to fold to their native state. In contrast, some shorter polymer chains (polypeptides) that misfold, are expected to have other funnels that can act as traps. We follow this loose definition here, by a funnel we mean a grouping of local optima in a coarse-grained structure around a high quality solution. According to this definition, most of the instances we analysed across the different values of  $\alpha$  reveal a single funnel. This is consistent with the results reported in [5], showing that there exist big valley structures in the landscapes of 3-SAT and MAX-3SAT. However, in some of the hardest overconstrained instances, our results reveal multiple funnel structures. An example is visualised in Fig. 3f, where three funnels can be identified as separate connected components. The two structures at the top converge to a global optimum (red node), so we can consider them as global funnels, whereas the structure at the bottom is a sub-optimal funnel. It is worth stressing that the global structures here encountered are approximations, as our approach is based on sampling, and on a particular value of the perturbation strength of the ILS sampling process. We argue that this is still an interesting observation, as it suggests that multiple funnels may exist in hard to solve MAX-SAT instances. When sub-optimal funnels exist, search can get trapped and fail to reach the global optimum despite a large computational effort.

## 6 Related Work

Some detailed analyses of random MAX-SAT fitness landscapes, using standard techniques, concentrate on the over-constrained region [18, 19]. More relevant to this research is the work presented in [4, 6, 20], where the landscape structure at and around the phase transition of random MAX-3SAT is explicitly examined. Frank et al. [6] identify several interesting features of plateaus (such as their size and number of exits) that impact the performance of local search algorithms. Sutton et al. [20] also studied the neutral regions by establishing theoretical bounds on plateau sizes, and assessing their accuracy on sampled problem instances. Sutton et al. [4] computed the exact correlation structure of random MAX-3SAT instances, showing that the correlation structure is oblivious to the phase transition, that is landscapes before and after the phase transition show the same correlation structure. This last result indicates that alternative techniques for studying the global structure of fitness landscapes, such as our proposal in this article, are required to gain a deeper understanding of search difficulty.

## 7 Discussion and Conclusion

We revisited the global structure of random MAX-3SAT fitness landscapes across the transition from underconstrained to overconstrained instances, using local optima networks analysis and visualisation. Our results confirm some previous findings, but also bring new structural metrics that correlate and help to explain the increased search cost incurred by stochastic local search algorithms in the phase transition region and beyond. The compressed LON model proved very valuable as a tool to analyse and visualise the landscapes' global structure. Before the phase transition, a large global optimal node is observed that is easy to reach (after a few hops) by local search algorithms. During the phase transition and beyond, however, the size of the global optimum drastically decreases, while the size of sub-optimal compressed nodes increases. The proportion of transitions to deteriorating solutions greatly increases, as well as the length of the trajectories towards the global optimum.

Under our empirical setting, most of the instances we studied revealed a single 'valley' or 'funnel'. A rigorous and well established definition of funnels is still lacking in evolutionary computation. We take the term here to loosely refer to a grouping of local optima, forming a coarse-level gradient towards a high quality solution at the end. Some of the hardest instances we considered (for high values of  $\alpha$ ) showed several disconnected groups of local optima. We suggest that these groupings may be related to the notion of sub-optimal funnels. Multiple funnels have been empirically observed in other hard combinatorial landscapes [10, 21, 22], contributing to our understanding of why some instances are harder to solve than others. To the best of our knowledge, multiple funnels have not been documented in the study of MAX-SAT fitness landscapes. This observation deserves further investigation. The funnel structure, as studied by local optima networks, depends on the amount of perturbation defining the transition edges [22]. Moreover, when sampling is involved, the identification of any global structure is approximated. A more precise characterisation of these structures, as well as the study of larger and different classes of MAX-SAT instances, is left as future work.

**Acknowledgements.** This research has been partially funded by the Spanish Ministry of Economy and Competitiveness (MINECO) and the European Regional Development Fund (FEDER) under contract TIN2017-88213-R (6city project), the University of Málaga, Consejería de Economía y Conocimiento de la Junta de Andalucía and FEDER under contract UMA18-FEDERJA-003 (PRECOG project), the Ministry of Science, Innovation and Universities and FEDER under contract RTC-2017-6714-5 (ECOIoT project), and the University of Málaga under contract PPIT.UMA.B1.2017/07 (EXHAURO Project).

## References

1. Cheeseman, P., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 331–337. Morgan Kaufmann (1991)

2. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of SAT problems. In: National Conference on Artificial Intelligence (AAAI), pp. 459–465 (1992)
3. Kirkpatrick, S., Selman, B.: Critical behavior in the satisfiability of random Boolean expressions. *Science* **264**(5163), 1297–1301 (1994)
4. Sutton, A.M., Whitley, L.D., Howe, A.E.: A polynomial time computation of the exact correlation structure of k-satisfiability landscapes. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 365–372. ACM (2009)
5. Zhang, W.: Configuration landscape analysis and backbone guided local search. Part I: Satisfiability and maximum satisfiability. *Artif. Intell.* **158** 1–26 (2004)
6. Frank, J., Cheeseman, P., Stutz, J.: When gravity fails: local search topology. *J. Artif. Intell. Res.* **7**, 249–281 (1997)
7. Reidys, C.M., Stadler, P.F.: Neutrality in fitness landscapes. *Appl. Math. Comput.* **117**(2), 321–350 (2001)
8. Ochoa, G., Tomassini, M., Verel, S., Darabos, C.: A study of NK landscapes’ basins and local optima networks. In: Genetic and Evolutionary Computation Conference (GECCO), pp. 555–56. ACM (2008)
9. Verel, S., Ochoa, G., Tomassini, M.: Local optima networks of NK landscapes with neutrality. *IEEE Trans. Evol. Comput.* **15**(6), 783–797 (2011)
10. Ochoa, G., Veerapen, N., Daolio, F., Tomassini, M.: Understanding phase transitions with local optima networks: number partitioning as a case study. In: Hu, B., López-Ibañez, M. (eds.) *EvoCOP 2017*. LNCS, vol. 10197, pp. 233–248. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-55453-2\\_16](https://doi.org/10.1007/978-3-319-55453-2_16)
11. Vérel, S., Daolio, F., Ochoa, G., Tomassini, M.: Local optima networks with escape edges. In: Hao, J.-K., Legrand, P., Collet, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) *EA 2011*. LNCS, vol. 7401, pp. 49–60. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-35533-2\\_5](https://doi.org/10.1007/978-3-642-35533-2_5)
12. Stadler, P.F.: Fitness landscapes. *Appl. Math. Comput.* **117**, 187–207 (2002)
13. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. In: *Handbook of Metaheuristics*, pp. 320–353 (2003)
14. Chicano, F., Whitley, L.D., Ochoa, G., Tinos, R.: Optimizing one million variable NK landscapes by hybridizing deterministic recombination and local search. In: Genetic and Evolutionary Computation Conference, pp. 753–760. ACM (2017)
15. Selman, B., Kautz, H.A., Cohen, B.: Local search strategies for satisfiability testing. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 521–532 (1996)
16. Cserdi, G., Nepusz, T.: The igraph software package for complex network research. *Int. J. Complex Syst.* **1695**(5), 1–9 (2006)
17. Doye, J.P.K., Miller, M.A., Wales, D.J.: The double-funnel energy landscape of the 38-atom Lennard-Jones cluster. *J. Chem. Phys.* **110**(14), 6896–6906 (1999)
18. Hoos, H.H., Smyth, K., Stützle, T.: Search space features underlying the performance of stochastic local search algorithms for MAX-SAT. In: Yao, X., et al. (eds.) *PPSN 2004*. LNCS, vol. 3242, pp. 51–60. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30217-9\\_6](https://doi.org/10.1007/978-3-540-30217-9_6)
19. Prugel-Bennett, A., Tayarani-Najaran, M.H.: Maximum satisfiability: Anatomy of the fitness landscape for a hard combinatorial optimization problem. *IEEE Trans. Evol. Comput.* **16**(3), 319–338 (2011)
20. Sutton, A.M., Howe, A.E., Whitley, L.D.: Estimating bounds on expected plateau size in MAXSAT problems. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) *SLS 2009*. LNCS, vol. 5752, pp. 31–45. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03751-1\\_3](https://doi.org/10.1007/978-3-642-03751-1_3)

21. Hains, D., Whitley, L.D., Howe, A.E.: Revisiting the big valley search space structure in the TSP. *JORS* **62**(2), 305–312 (2011)
22. Ochoa, G., Herrmann, S.: Perturbation strength and the global structure of QAP fitness Landscapes. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) *PPSN 2018*. LNCS, vol. 11102, pp. 245–256. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_20](https://doi.org/10.1007/978-3-319-99259-4_20)





# Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy

Quentin Renau<sup>1,2</sup>, Carola Doerr<sup>3(✉)</sup>, Johann Dreo<sup>1</sup>, and Benjamin Doerr<sup>2</sup>

<sup>1</sup> Thales Research and Technology, Palaiseau, France  
{quentin.renau,johann.dreo}@thalesgroup.com

<sup>2</sup> Laboratoire d'Informatique (LIX), CNRS, École Polytechnique,  
Institut Polytechnique de Paris, Palaiseau, France

<sup>3</sup> Sorbonne Université, CNRS, LIP6, Paris, France  
carola.doerr@lip6.fr

**Abstract.** Exploratory landscape analysis (ELA) supports supervised learning approaches for automated algorithm selection and configuration by providing sets of features that quantify the most relevant characteristics of the optimization problem at hand. In black-box optimization, where an explicit problem representation is not available, the feature values need to be approximated from a small number of sample points. In practice, uniformly sampled random point sets and Latin hypercube constructions are commonly used sampling strategies.

In this work, we analyze how the sampling method and the sample size influence the quality of the feature value approximations and how this quality impacts the accuracy of a standard classification task. While, not unexpectedly, increasing the number of sample points gives more robust estimates for the feature values, to our surprise we find that the feature value approximations for different sampling strategies do not converge to the same value. This implies that approximated feature values cannot be interpreted independently of the underlying sampling strategy. As our classification experiments show, this also implies that the feature approximations used for training a classifier must stem from the same sampling strategy as those used for the actual classification tasks.

As a side result we show that classifiers trained with feature values approximated by Sobol' sequences achieve higher accuracy than any of the standard sampling techniques. This may indicate improvement potential for ELA-trained machine learning models.

**Keywords:** Exploratory landscape analysis · Automated algorithm design · Black-box optimization · Feature extraction

## 1 Introduction

The impressive advances of machine learning (ML) techniques are currently shaking up literally every single scientific discipline, often in the function to support

decisions previously requiring substantial expert knowledge by recommendations that are derived from automated data-processing techniques. Computer science is no exception to this, and an important application of ML is the selection and configuration of optimization heuristics [11, 13, 31], where automated techniques have proven to yield tremendous efficiency gains in several classic optimization tasks, including SAT solving [34] and AI planning [33].

In the context of numerical optimization, supervised learning approaches are particularly common [3, 16, 24]. These methods often build on features developed in the context of *fitness landscape analysis* [18, 26], which aims at quantifying the characteristics of an optimization problem through a set of features. More precisely, a *feature* maps a function (the optimization problem)  $f : S \subseteq \mathbb{R}^d \rightarrow \mathbb{R}$  to a real number. Such a feature could measure, for example, the *skewness* of  $f$ , its *multi-modality*, or its similarity to a quadratic function.

In practice, many numerical optimization problems are *black-box problems*, i.e., they are not explicitly modeled but can be accessed only through the evaluation of *samples*  $x \in S$ . Hyper-parameter optimization is a classical example for such an optimization task for which we lack an a priori functional description. In these cases, i.e., when  $f$  is not explicitly given, the feature values need to be approximated from a set of  $(x, f(x))$  pairs. The approximation of feature values through such samples is studied under the notion of *exploratory landscape analysis* (ELA [22]). ELA has been successfully applied, for example, in per-instance hyperparameter optimization [3] and in algorithm selection [16]. When applying ELA to a black-box optimization problem, the user needs to decide **how many samples to take** and **how to generate these**.

When the functions are fast to evaluate, a typical recommendation is to use around  $50d$  samples [14]. For costly evaluations, in contrast, one has to resort to much fewer samples [2]. It is well known that the quality of the feature approximation depends on the sample size. Several works have investigated how the dispersion of the feature approximation decreases with increasing sample size, see [27, 29] and references mentioned therein. The recommendation made in [14] is meant as a compromise between a good accuracy of the feature value and the computational effort required to approximate it.

Interestingly, the question which sampling strategy to use is much more widely open. In the context of ELA, the by far most commonly used strategies are uniform sampling (see, e.g. [3, 23]) and Latin Hypercube Sampling (LHS, see, e.g., [14, 16]). These two strategies are also predominant in the broader ML context, although a few works discussing alternative sampling techniques exist (e.g., [30]). A completely different approach, which we do not further investigate in this work, but which we mention for the sake of completeness, is to compute feature values from the search trajectory of an optimization heuristic. Examples for such approaches can be found in [5, 12]. Note though, that such trajectory-based feature value approximations are only applicable when the user has the freedom to chose the samples from which the feature values are computed, a prerequisite not always met in practice.

We share in this work the interesting observation that **the feature value approximations obtained from different sampling methods do not converge to the same values**. Put differently, the feature values are not absolute, but strongly depend on the distribution from which the  $(x, f(x))$  pairs have been sampled. This finding is in sharp contrast to what seems to be common belief in the community. For example, it is argued in Saleem et al. [29, page 81] that “As  $N$  [the sample size]  $\rightarrow \infty$  the feature  $\Phi_i$  approaches a true value”. We show in this work that no such “true value” exist: the feature values cannot be interpreted as stand-alone measures, but only in the context of the samples from which they are approximated.

Our observation has the undesirable side effect that machine-trained models achieve peak performance only when the sampling method applied to train the models was identical to the method used to approximate the feature values for the problem under consideration. Since the latter can often not be sampled arbitrarily (e.g., because we are forced to use existing evaluations), this implies that one might have to re-do a training ensemble from scratch. In application where we are free to chose the samples  $(x, f(x))$ , we need to ensure to store and share the points (or at least the distributions) that were used to approximate the feature values for the training data. Also, when using feature values to compare problems (see [7, 8] for examples), one needs to pay particular attention that the differences in feature values are indeed caused by the function properties, and not by the sampling technique.

Given the sensitivity with respect to the sampling *distribution*, one may worry that even the random number generator may have an impact on the feature value approximations. On a more positive note than the results described above, we show that this is not the case. More precisely, we show that uniform sampling based on two very different random number generators, Mersenne Twister and RANDU, respectively, give comparable results.

Another observation that we share with this paper is the fact that sampling strategies different from uniform and LHS sampling seem worth further investigation. More precisely, we show that classifiers trained with feature values that are approximated from samples generated by Sobol’s low-discrepancy sequences perform particularly well on our benchmark problems. This challenges the state-of-the-art sampling routines used in ELA, and raises the question whether properties such as low discrepancy, good space-filling designs, or small stochastic dispersion correlate favorably with good performance in supervised learning.

**Reproducibility:** The landscape data used for our analysis as well as several plots visualizing it are available at [28].

## 2 The Impact of Low Feature Robustness on Classification Accuracy

Instead of directly measuring and comparing the dispersion and the modes of the feature value approximation, we consider their impact on the accuracy in a simple classification task. We believe this approach offers a very concise way

to demonstrate the effects that the different sampling strategies can have in the context of classical ML tasks. The classification task and its experimental setup is described in Sect. 2.1. We then briefly comment on the distribution of the feature values (Sect. 2.2), on the classifiers (Sect. 2.3), and on the sampling strategies (Sect. 2.4). The impact of the low feature value robustness will then be discussed in Sect. 2.5, whereas all discussions related to the impact of the sampling strategy are deferred to Sect. 3.

## 2.1 Classification of BBOB Functions

We consider the 24 BBOB functions that form the set of noiseless problems within the COCO (Comparing Continuous Optimisers) platform [9], a standard benchmark environment for numerical black-box optimization. For each BBOB problem we take the first instance of its 5-dimensional version. Each of these instances is a real-valued function  $f : [-5, 5]^5 \rightarrow \mathbb{R}$ . The choice of dimension and instance are not further motivated, but are identical to those made in [27], for better comparability.

For the feature approximation, we sample for each of the 24 functions  $f$  a number  $n$  of random points  $x^{(1)}, \dots, x^{(n)}$ , and we evaluate their function values  $f(x^{(1)}), \dots, f(x^{(n)})$ . The pairs  $(x^{(i)}, f(x^{(i)}))_{i=1}^n$  are then fed to the *flacco* package [15], which returns a vector of 46 features.<sup>1</sup> We repeat this procedure 100 independent times, each time sampling from the same random distribution. This leaves us with 100 feature vectors per each function. From this set we use 50 uniformly chosen feature vectors (per function) for training a classifier that, given a previously unseen feature vector, shall output which of the 24 functions it is faced with. We test the classifier with all 50 feature vectors that were not chosen for the training, and we record the average classification accuracy, which we measure as the fraction of correctly attributed function labels. We apply 50 independent runs of this uniform random sub-sampling validation, i.e., we repeat the process of splitting the  $24 \times 100$  feature vectors into  $24 \times 50$  training instances and  $24 \times 50$  test instances 50 independent times.

To study the effects of the sample size, we conduct the above-described experiment for three different values of  $n$ :  $n = 30$ ,  $n = 300$ , and  $n = 5^5 = 3125$ .

The BBOB functions are designed to cover a broad spectrum of numerical problems found in practice. They are therefore meant to be quite different in nature. Visualizations of the functions provided in [10] support this motive. We should therefore expect to see very good classification accuracy, even with non-tuned off-the-shelf classifiers.

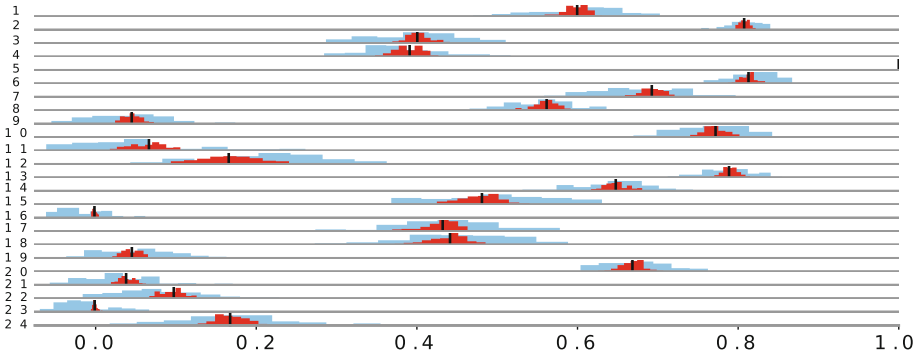
---

<sup>1</sup> Note here that *flacco* covers 343 features in total, which are grouped into 17 feature sets [13]. However, following the discussion in [27] we only use 6 of these sets: dispersion (disp), information content (ic), nearest better clustering (nbc), meta model (ela.meta),  $y$ -distribution (ela.distr), and principal component analysis (pca).

## 2.2 Feature Value Distributions

Figure 1 shows the distribution of the feature value approximations for one particular feature, which measures the adjusted fit to a linear model (observe that function 5 is correctly identified as a linear slope with an  $R^2$  value of 1). Results are shown for  $n = 300$  (blue) and  $n = 3125$  (red) LHS samples.

We observe that the median values (black bars) of the single feature plotted in Fig. 1 are already quite diverse, i.e. – taking a few exceptions aside – they show fairly large pairwise distances. However, we also see that the dispersion of the approximated feature values is large enough to require additional features for proper classification. We also see that, in line with observations made in [27, 29], the dispersion of the approximations reduces with increasing sample size.



**Fig. 1.** Distribution of the approximations for the `ela_meta.lin_simple.adj_r2` feature value, for 100 independently drawn LHS designs of  $n = 300$  (blue) and  $n = 3125$  (red) samples. Each row corresponds to one of the 24 BBOB functions. The black bars indicate the median value of the  $n = 3125$  data. (Color figure online)

## 2.3 Classifiers: Decision Trees and KNN

All the classification experiments are made using the Python package *scikit learn* [25, version 0.21.3]. Since we are not interested in our work to compare accuracy of different classifiers, but rather aim at understanding the sensitivity of the classification result with respect to the random feature value approximations, and since more sophisticated classifiers (in particular ensemble learning methods such as random forests) tend to require more computational overhead, we do not undertake any effort in optimizing the performance of these classifiers, and resort to default implementations of two common, but fairly different, classification techniques instead. Concretely, we use ***K* Nearest Neighbors (KNN)** (we use  $K = 5$ ) and **decision trees**. We decided to run the experiments with two different classifiers to analyze whether the effects observed for one method also occur with the other one. This should help us avoid reporting classifier-specific effects. For some selected results, we have performed a cross-validation with 5 independent runs of a random forest classifier, and found that – while

the overall classification results are better than for KNN and decision trees – the structure of the main results (precisely, the results reported in Fig. 4) is very similar to that of the two classifiers discussed below.

## 2.4 Sampling Designs

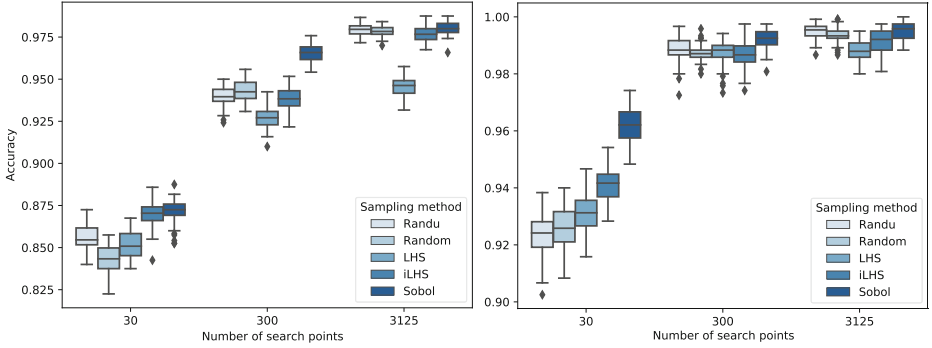
As mentioned previously, the two most commonly used sampling strategies in feature extraction, and more precisely in exploratory landscape analysis, are Latin Hypercube Sampling (LHS) and uniform random sampling. To analyze whether the sensitivity of the random feature value approximations depend on the strategy, we investigate a total number of five different sampling strategies, which we briefly summarize in this section.

*Uniform Sampling.* We compare uniform random sampling based on two different pseudo-random number generators:

- **random:** We report under the name *random* results for the Mersenne Twister [20] random number generator. This generator is commonly used by several programming languages, including Python, C++, and R. It is widely considered to be a reliable generator.
- **RANDU:** we compare the results to those for the linear congruential number generator RANDU. This generator is known to have several deficits such as an inherent bias that results in the numbers falling into parallel hyperplanes [17]. We add this generator to investigate whether the quality of the random sampling has an influence on the feature value approximations and to understand (in Sect. 3) whether apart from the sampling strategy also the random number generator needs to be taken into account when transferring ELA-trained ML-models to new applications.

*Latin Hypercube Sampling (LHS).* LHS [21] is a commonly used quasi-random method to generate sample points for computer experiments. In LHS, new points are sampled avoiding the coordinates of the previously sampled points. More precisely, the range of each coordinate is split into  $n$  equally-sized intervals. From the resulting  $n \times \dots \times n$  grid the points are chosen in a way that each one-dimensional projection has exactly one point per interval.

- **LHS:** Our first LHS designs are those provided by the *pyDOE* Python package (version 0.3.8). We use the centered option, which takes the middle point of each selected cube as sample.
- **iLHS:** The “*improved*” LHS (**iLHS**) designs available in *flacco*. This strategy builds on work of Beachofski et Grandhi [1]. Essentially, it implements a greedy heuristic to choose the next points added to the design. At each step, it first samples a few random points, under the condition of not violating the Latin Hypercube design. From these candidates the algorithm chooses the one whose distance to its nearest neighbor is closest to the ideal distance  $n/\sqrt[n]{n}$ .



**Fig. 2.** Classification accuracy by sampling strategy, sample size, and classifier (left = KNN, right = decision trees). Note the different scale of the  $y$ -axes.

*Sobol’s Low-Discrepancy Sequence.* We add to our investigation a third type of sampling strategies, the sequences suggested by **Sobol’** in [32]. Sobol’ sequences are known to have small *star discrepancy*, a property that guarantees small approximation errors in several important numerical integration tasks. The interested reader is referred to [6, 19] for an introduction to these important families of quasi-random sampling strategies.

For our experiments we generate the Sobol’ sequences from the Python package *sobol\_seq* (version 0.1.2), with randomly chosen initial seeds.

### 2.5 Classification Accuracy

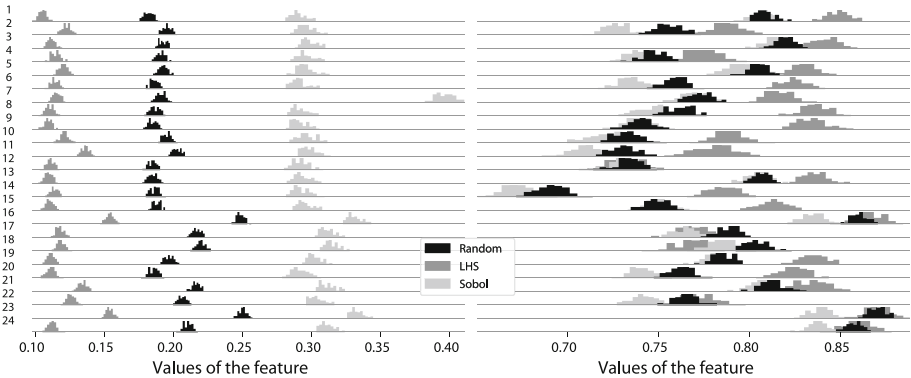
Figure 2 reports the distribution of the classification accuracy achieved by each of the five sampling strategies, when training and testing uses the same sampling strategy. The results on the left are for KNN classifiers, the ones on the right for decision trees. The absolute value of the medians can be inferred from Fig. 4 (which we will discuss in Sect. 3). As expected, we see higher classification accuracy with increasing sample size. We also observe that the KNN results are slightly (but with statistical significance) worse than those of the decision trees. Recall, however, that this is not a focus of our search, and no fine-tuning was applied to the classification methods. Comparison between the two classifiers should therefore only be taken with great care.

For KNN we nicely observe that the dispersion of the classification error reduces with increasing sample size. This aligns with the reduced variance of the feature value approximations discussed in Sect. 2.2. For the decision tree classifier the dispersion of the classification accuracy reduces significantly from 30 to 300 samples, but then stagnates when increasing further to 3125 samples.

No substantial differences between the two random number generators can be observed. For LHS, in contrast, the centered sampling method yields considerably worse classification accuracy than iLHS.

Finally, we also observe that in each of the six reported (classifier, sample size) combinations the median and also the average (not plotted) classification accuracy of the Sobol’ sampling strategy is largest, with box plots that are well separated from those of the other sampling strategies, in particular for  $n \leq 300$  samples. Kolmogorov-Smirnov tests confirm statistical significance in almost all cases. We omit a detailed discussion, for reasons of space.

The good performance of Sobol’ sampling suggests to question the state of the art in feature extraction, which considers uniform and LHS designs as default. Interestingly, our literature research revealed that Sobol’ points were already recommended in the book of Santner *et al.* [30]. It is stated there that Sobol’ sequences may enjoy less popularity in ML because of their slightly more involved generation. Santner *et al.* therefore recommend LHS designs as fall-back option for large sample sizes. Our data, however, does not support this suggestion, and the (very small) advantages of the random sampling strategies over iLHS are indeed statistically significant.



**Fig. 3.** Distribution of feature value approximations for the `nbc.dist_ratio.coeff_var` feature (left) and `ic.h_max` feature (right). Results are for 100 independent evaluations of  $n = 3125$  samples generated by LHS, random, and Sobol’ generators, respectively.

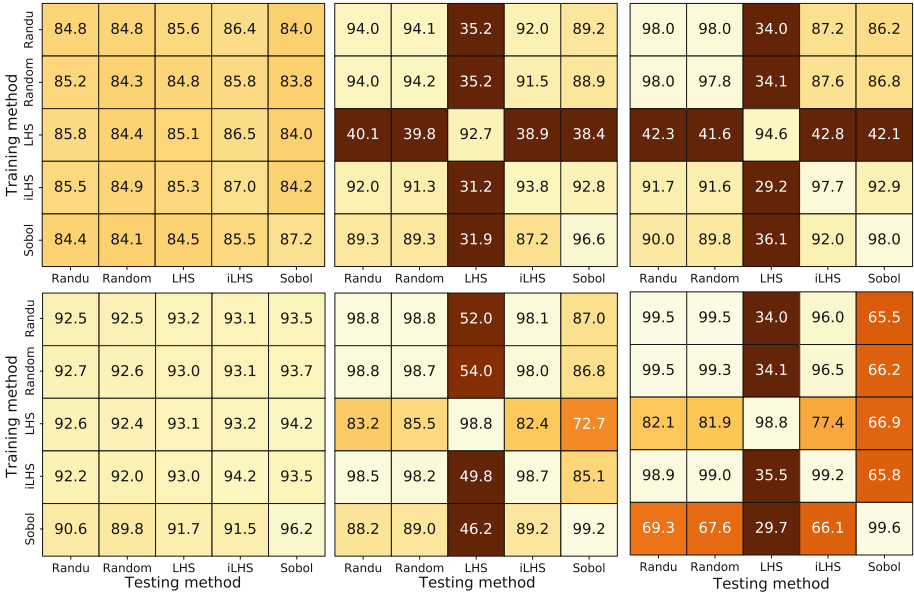
### 3 The Sampling Strategy Matters

Following the discussion above, it seems plausible to believe that the differences in classification accuracy is mostly caused by the dispersion of the feature value approximations. However, while this is true when we compare results for different sample sizes, we will show in this section that dispersion is not the main driver for differences between the tested sampling strategies.

Figure 3 plots the distribution of feature value approximations for two of our 46 features. It illustrates an effect which came as a big surprise to us. Whereas features are typically considered to have an *absolute* value (see the examples mentioned in the introduction), we observe here that the results very strongly



depend on the sampling strategy. For the feature values displayed on the left, not only do the distributions have different medians and means, but they are even non-overlapping. This behavior is consistent for the different sample sizes (not plotted here). While this chart on the left certainly displays an extreme case, the same effect of convergence against different feature values can be observed for a large number of features (but not always for all functions or all different sampling strategies), as we can observe in the right chart of Fig. 3. The latter also squashes hopes for simple translation of feature values from one sampling strategy to another one: looking at functions 10 and 12, for example, we see that random and Sobol’ sampling yield similar feature values for both functions, whereas those approximated by LHS sampling are much larger for f10 as for f12. **We thus observe that the interpretation of a feature value cannot be carried out without knowledge of the sampling strategy.**



**Fig. 4.** Heatmaps of median classification accuracy for KNN (top) and decision trees (bottom), for feature values approximated by 30 search points (left), 300 search points (middle), and 3125 search points (right), respectively. (Color figure online)

We investigate the impact of the strategy-dependent feature values by performing the following classification task. We use the same feature values as generated for the results reported in Sect. 2, but we now train the classifiers with the feature value approximations obtained from one sampling strategy, and we track the classification accuracy when tested with feature value approximations obtained by one of the other strategies. Apart from this twist, the experimental routine is the same as the one described in Sect. 2.1.

The heatmaps in Fig. 4 display the median classification accuracy of the 25 possible combinations of training and testing sampling strategies. We show results for all three sample sizes,  $n = 30$  (left),  $n = 300$  (middle), and  $n = 3125$  (right). Rows correspond to the training strategy, the columns to the test strategy; the diagonals therefore correspond to the data previously discussed in Sect. 2.5. KNN data is shown in the top, those for decision trees on the bottom.

For sample size  $n = 300$  and  $n = 3125$  the best or close-to-best classification accuracy is achieved when the sampling strategy for the testing instances is identical to that of the training instances. This is independent of the classifier. Interestingly, this observation does not apply to the case with  $n = 30$  samples, where, e.g., the KNN classifiers trained with LHS data achieve better accuracy with iLHS feature approximations (86.5%) than with LHS approximations (85.1%). The same holds for the classifiers trained with data from the random sampling strategy (for both random number generators). The differences between the different training and test data combinations, however, are rather small in these cases. In addition, the dispersion of the classification accuracies are relatively large for  $n = 30$  samples, with ranges that are very similar to those plotted in Fig. 2. We also recall that the overall classification accuracy, in light of the high diversity of the 24 BBOB functions, is not as good as it may seem at the first glance.

We also observe that, for  $n = 30$ , the KNN classifiers (except for the Sobol'-trained ones) perform best when tested with iLHS test samples, whereas for decision trees we see better results with Sobol' test data. This however, applies only to the case  $n = 30$ , as we shall discuss in the following.

Moving on to the cases  $n \geq 300$ , we observe that – in line with the observation made in Fig. 2 – the average classification accuracy increases significantly, to values well above 90%, with a few notable exceptions: The poor accuracy of LHS both as test and as training instances stands out, but is consistent for both classifiers, and both sample sizes  $n = 300$  and  $n = 3125$ . Albeit not as bad, the Sobol'-approximated feature values also lead to comparatively poor performance on almost all classifiers not trained with Sobol'-approximations (an exception are the iLHS-trained KNN classifiers using  $n = 300$  samples). Consistent to this, the Sobol'-trained classifiers have low classification accuracy when tested with feature values from the other four strategies. While this effect is most noticeable for the decision tree classifiers, it also applies to KNN. A closer inspection of the feature value approximations reveals that those for iLHS, random sampling, and Randu are much more alike to each other than to the LHS or Sobol' features. For  $947 = 43\%$  of all  $24 \times 46$  (function, feature) pairs, the median of the LHS feature values with  $n = 3125$  samples is either smaller or larger than that of the other strategies. For Sobol' points, this value is  $725 = 33\%$ . Of course, this just gives a first impression. Plots similar to Fig. 3 provide much more details; they are available for all features at [28]. A thorough investigation into *why* these differences exist forms an important next step for our research, cf. Sect. 5.

Given that we use the centered option for the LHS strategy (see Sect. 2.4), one might be tempted to think that the LHS-approximations are more concentrated than those of the other sampling strategies. This, however, cannot be confirmed by our data: the dispersion of the LHS approximations is comparable to that of the other strategies.

Finally, we observe that the two random strategies show high inter-strategy classification accuracy. Their feature approximations work furthermore quite well with classifiers trained on iLHS data. However, while all of the results reported above also apply to average (instead of median) classification accuracy, the average classification error of the iLHS-trained KNN-classifiers is considerably worse for (Mersenne-Twister) random feature value approximations than for those obtained from Randu (91.4% vs. 94.0% accuracy for  $n = 300$  and 91.7% vs. 98% for  $n = 3125$  samples).

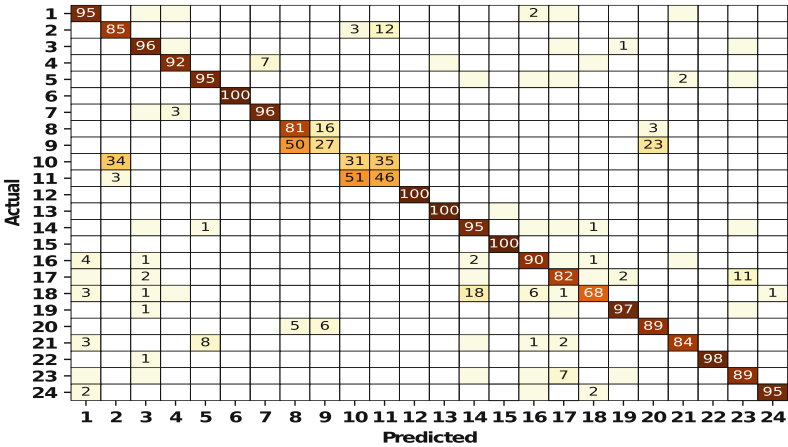


Fig. 5. Average classification result across 50 independently iLHS-trained KNN classifiers, each tested with 50 Sobol’ feature value approximations using  $n = 30$  evaluations. Numbers are provided for  $>1\%$  probabilities only.

### 4 Confusion Matrices

The results reported in the previous sections were all aggregated over the 24 functions from the BBOB benchmark set. In Fig. 5 we analyze which functions are misclassified most frequently, and by which functions they are confused with. The matrix shows results for the 50 KNN classifiers trained with iLHS feature approximations and tested with Sobol’ data (50 tests per classifier), for  $n = 30$  sample points. We recall from Fig. 4 that the median classification accuracy of this combination is 84.2%. This is also the average accuracy.

Most functions are correctly classified with probability at least 80%. For twelve functions we observe at least 95% accuracy. Only four functions (9–11, 18) are misclassified with probability  $\geq 30\%$ , and those are typically confused by the same one or two other functions. Function 2, for example, is misclassified as function 11 in 12% of the tests.

We do not show the confusion matrices for the other  $3 \times 25$  cases, but note that – overall – the patterns are quite consistent across all KNN classifiers. Naturally, the concentration on the diagonal increases with larger sample sizes. We also see a higher concentration for the mis-classifications as well. For example, in the same iLHS-Sobol’ setting as above with  $n = 3125$  samples 15 functions have accuracy  $\geq 95\%$ , and only five function pairs with mis-classification rate  $\geq 5\%$  are observed. Four of these occur with probability  $\leq 8\%$ . One mis-classification stands out: function 9 is classified as function 20 in 93% of the cases.

For decision trees, the structure of the confusion matrices is similar to those of KNN for  $n = 30$  samples. For  $n = 3125$  samples, however, the mis-classifications are much more scattered across several function pairs.

Without going further into details, we note that these results can be used to understand deficits of the benchmark sets (frequent confusion of two functions could indicate some problems are quite alike), of the selected features (if they do not capture major differences between two structurally different functions), and the classifiers (e.g., the scattered confusion matrix of the decision trees for  $n = 3125$  samples).

## 5 Conclusions

We have analyzed the impact of the stochasticity inherent to feature value approximations on the use of exploratory landscape analysis in classic ML tasks. Our key findings are the following.

**(1) ELA features are not absolute, but should be interpreted only in the context of the sampling strategy.** As an important consequence of this observation, we derive the recommendation that the sampling strategy of the training data should match the sampling strategy of the test data. Note that this also implies more data needs to be shared to obtain reproducible and/or high quality results.

**(2) The good results achieved by the classifiers trained with Sobol’ samples suggests to revive a recommendation previously made by Santner *et al.* [30], and to further investigate this sampling strategy in the context of other feature extraction tasks, i.e., beyond applications in exploratory landscape analysis.** In this context, it would also be worthwhile to study other low-discrepancy constructions, which are recently gaining interest in the broader ML context, e.g., in the context of *one-shot optimization* (the task of optimizing a black-box problem through the best of  $n$  parallel samples, see [4] and references therein). Whether good performance in one-shot optimization correlates with a good approximation of feature values forms another interesting avenue for future work.

While we have focused in this work on classification accuracy only, we are also planning on a more detailed analysis of the feature approximations themselves. In particular, we aim at understanding a functional relationship between the sampling strategies and their feature value approximations. This shall help us identify correction methods that translate values obtained from one sampling strategy to another. This, ultimately, may help us by-pass the need for sample-specific training.

We also believe that the confusion matrices such as the one in Fig. 5 should be explored further, to understand which BBOB instances are more alike than others. Such information can be useful for instance selection and generation.

**Acknowledgments.** This research benefited from the support of the FMJH Program PGMO and from the support of EDF and Thales.

## References




1. Beachkofski, B., Grandhi, R.: Improved distributed hypercube sampling. In: 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. American Institute of Aeronautics and Astronautics (2002). <https://doi.org/10.2514/6.2002-1274>
2. Belkhir, N., Dréo, J., Savéant, P., Schoenauer, M.: Surrogate assisted feature computation for continuous problems. In: Festa, P., Sellmann, M., Vanschoren, J. (eds.) LION 2016. LNCS, vol. 10079, pp. 17–31. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-50349-3\\_2](https://doi.org/10.1007/978-3-319-50349-3_2)
3. Belkhir, N., Dréo, J., Savéant, P., Schoenauer, M.: Per instance algorithm configuration of CMA-ES with limited budget. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), pp. 681–688. ACM (2017). <https://doi.org/10.1145/3071178.3071343>
4. Cauwet, M., et al.: Fully parallel hyperparameter search: reshaped space-filling. CoRR abs/1910.08406 (2019). <http://arxiv.org/abs/1910.08406>
5. Derbel, B., Liefoghe, A., Vérel, S., Aguirre, H., Tanaka, K.: New features for continuous exploratory landscape analysis based on the SOO tree. In: Proceedings of Foundations of Genetic Algorithms (FOGA), pp. 72–86. ACM (2019). <https://doi.org/10.1145/3299904.3340308>
6. Dick, J., Pillichshammer, F.: Digital Nets and Sequences. Cambridge University Press, Cambridge (2010)
7. Doerr, C., Dréo, J., Kerschke, P.: Making a case for (hyper-)parameter tuning as benchmark problems. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO, Companion), pp. 1755–1764. AMC (2019). <https://doi.org/10.1145/3319619.3326857>
8. Garden, R.W., Engelbrecht, A.P.: Analysis and classification of optimisation benchmark functions and benchmark suites. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2014), pp. 1641–1649. IEEE (2014). <https://doi.org/10.1109/CEC.2014.6900240>
9. Hansen, N., Auger, A., Mersmann, O., Tusar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. CoRR abs/1603.08785 (2016). <http://arxiv.org/abs/1603.08785>

10. Hansen, N., Finck, A.A.S., Ros, R.: Real-Parameter Black-box Optimization Benchmarking: Experimental Setup. RR-7215, INRIA (2010). <https://hal.inria.fr/inria-00462481>
11. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): Automated Machine Learning. TSS-CML. Springer, Cham (2019). <https://doi.org/10.1007/978-3-030-05318-5>
12. Jankovic, A., Doerr, C.: Adaptive landscape analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO 2019), pp. 2032–2035. ACM (2019)
13. Kerschke, P., Hoos, H., Neumann, F., Trautmann, H.: Automated algorithm selection: survey and Perspectives. *Evol. Comput.* **27**(1), 3–45 (2019)
14. Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: Low-budget exploratory landscape analysis on multiple peaks models. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), pp. 229–236. ACM (2016). <https://doi.org/10.1145/2908812.2908845>
15. Kerschke, P., Trautmann, H.: The R-package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In: Proceedings of CEC, pp. 5262–5269. IEEE (2016). flacco is available at <http://kerschke.github.io/flacco/>
16. Kerschke, P., Trautmann, H.: Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evol. Comput.* **27**(1), 99–127 (2019). <https://doi.org/10.1162/evco.a.00236>
17. Knuth, D.: The Art of Computer Programming: Seminumerical Algorithms. Addison-Wesley, Boston (1998)
18. Malan, K., Engelbrecht, A.P.: A survey of techniques for characterising fitness landscapes and some possible ways forward. *Inf. Sci.* **241**, 148–163 (2013). <https://doi.org/10.1016/j.ins.2013.04.015>
19. Matoušek, J.: Geometric Discrepancy, 2nd edn. Springer, Heidelberg (2009)
20. Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.* **8**(1), 3–30 (1998). <https://doi.org/10.1145/272991.272995>
21. McKay, M., Beckman, R., Conover, W.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21**, 239–245 (1979). <http://www.jstor.org/stable/1268522>
22. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), pp. 829–836. ACM (2011). <https://doi.org/10.1145/2001576.2001690>
23. Morgan, R., Gallagher, M.: Sampling techniques and distance metrics in high dimensional continuous landscape analysis: limitations and improvements. *IEEE Trans. Evol. Comput.* **18**(3), 456–461 (2014). <https://doi.org/10.1109/TEVC.2013.2281521>
24. Muñoz, M.A., Sun, Y., Kirley, M., Halgamuge, S.K.: Algorithm selection for black-box continuous optimization problems: a survey on methods and challenges. *Inf. Sci.* **317**, 224–245 (2015). <https://doi.org/10.1016/j.ins.2015.05.010>
25. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *JMLR* **12**, 2825–2830 (2011)
26. Pitzer, E., Affenzeller, M.: A comprehensive survey on fitness landscape analysis. In: Fodor, J., Klempous, R., Suárez Araujo, C.P. (eds.) Recent Advances in Intelligent Engineering Systems. Studies in Computational Intelligence, vol. 378, pp. 161–191. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-23229-9\\_8](https://doi.org/10.1007/978-3-642-23229-9_8)

27. Renau, Q., Dreo, J., Doerr, C., Doerr, B.: Expressiveness and robustness of landscape features. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO, Companion), pp. 2048–2051. ACM (2019). <https://doi.org/10.1145/3319619.3326913>
28. Renau, Q., Doerr, C., Dreo, J., Doerr, B.: Experimental data set for the study “exploratory landscape analysis is strongly sensitive to the sampling strategy”, June 2020. <https://doi.org/10.5281/zenodo.3886816>
29. Saleem, S., Gallagher, M., Wood, I.: Direct feature evaluation in black-box optimization using problem transformations. *Evol. Comput.* **27**(1), 75–98 (2019)
30. Santner, T., Williams, B., Notz, W.: The Design and Analysis of Computer Experiments. Springer Series in Statistics. Springer, New York (2003). <https://doi.org/10.1007/978-1-4757-3799-8>
31. Smith-Miles, K.A.: Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.* **41**(1) (2009). <https://doi.org/10.1145/1456650.1456656>
32. Sobol', I.: On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Comput. Math. Math. Phys.* **7**(4), 86–112 (1967). [https://doi.org/10.1016/0041-5553\(67\)90144-9](https://doi.org/10.1016/0041-5553(67)90144-9)
33. Vallati, M., Hutter, F., Chrupa, L., McCluskey, T.: On the effective configuration of planning domain models. In: Proceedings of IJCAI 2015. AAAI (2015)
34. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: SATzilla: portfolio-based algorithm selection for sat. *JAIR* **32**, 565–606 (2008). <http://dl.acm.org/citation.cfm?id=1622673.1622687>



# One PLOT to Show Them All: Visualization of Efficient Sets in Multi-objective Landscapes

Lennart Schäpermeier<sup>(✉)</sup> , Christian Grimme<sup></sup>, and Pascal Kerschke<sup></sup>

Statistics and Optimization, University of Münster, Münster, Germany  
{schaepermeier,christian.grimme,kerschke}@uni-muenster.de

**Abstract.** Visualization techniques for the decision space of continuous multi-objective optimization problems (MOPs) are rather scarce in research. For long, all techniques focused on global optimality and even for the few available landscape visualizations, e.g., *cost landscapes*, globality is the main criterion. In contrast, the recently proposed *gradient field heatmaps (GFHs)* emphasize the location and attraction basins of local efficient sets, but ignore the relation of sets in terms of solution quality.

In this paper, we propose a new and hybrid visualization technique, which combines the advantages of both approaches in order to represent local and global optimality together within a single visualization. Therefore, we build on the GFH approach but apply a new technique for approximating the location of locally efficient points and using the divergence of the multi-objective gradient vector field as a robust second-order condition. Then, the relative dominance relationship of the determined locally efficient points is used to visualize the complete landscape of the MOP. Augmented by information on the basins of attraction, this *Plot of Landscapes with Optimal Trade-offs (PLOT)* becomes one of the most informative multi-objective landscape visualization techniques available.

**Keywords:** Multi-objective optimization · Continuous optimization · Visualization · Landscape analysis · Efficient sets

## 1 Introduction

Traditionally, the visualization of optimization problems in decision space is one of the basic approaches to investigate challenges of so-called functional landscapes and to design basic algorithmic principles. Hence, low dimensional visualization is used in text books [1, 3] and algorithm research alike. For a single objective, each point in a continuous two-dimensional search space can be assigned with a function value which is interpreted as height. Overall, this leads to very natural notions of *mountains* and *valleys* for local maxima and minima, *ridges* for discontinuities, as well as *plateaus* for areas of equal height.



In evolutionary computation, many early theoretical results as well as later algorithmic concepts were first developed (and only successively generalized) by using low-dimensional visualizations of benchmark problems and their challenges. For continuous multi-objective optimization problems (MOPs), however, such straight-forward visualization techniques are not available. This is mainly rooted in the fact that MOPs comprise at least two contradicting objectives to be optimized simultaneously. Consequently, not a single or few global optimal solutions are sought but a set of optimal trade-off solutions – the so-called *Pareto set*. These solutions have as many objective function values (and thus height values) as objectives, which makes a standard landscape visualization infeasible.

For few ( $\leq 3$ ) objectives, a classical visualization of the true or approximated set of efficient solutions, usually the Pareto front – the Pareto set’s image in objective space – is used. However, by focusing purely on the objective space, one ignores all interaction effects from the MOP’s input variables in the decision space. Compared to the single-objective case, this is like reducing the entire landscape to the function values of its optimal solutions, and plotting them on a one-dimensional scale. Almost no information on the structural properties of the problem landscapes can be derived from this. Consequently, only little is known on MOP landscape properties and almost no algorithmic design is based on comparable insights like in the single-objective case.

Although a straightforward visualization of MO landscapes is not available, there exist very few techniques for getting insights into these landscapes. The *cost landscapes* proposed by Fonseca [5] use a dominance ranking approach to evaluate each point in decision space w.r.t. the global optimal trade-offs. Although this delivers a kind of landscape in relation to the global optimum, it does not capture local optimal sets and their basins of attraction. An alternative visualization approach that explicitly addresses locality has been proposed by Kerschke and Grimme [10, 12]. It produces (multi-objective) gradient field heatmaps (GFHs) using the Fritz-John (necessary) condition for identifying local optima [11, 18]. The GFHs show local basins and locally efficient sets but have two drawbacks: they do not provide a ranking of local sets w.r.t the global set and indicate local efficiency only indirectly by the height. Within this work, we address the weaknesses of both approaches and contribute the following:

1. We propose a robust approach to determine locally efficient points explicitly. This includes a suitable second-order condition based on the divergence of the multi-objective gradient to confirm or exclude points that are considered to be locally optimal according to the first-order conditions.
2. Additionally, we combine and extend the two aforementioned state-of-the-art visualization approaches, i.e., the cost landscapes [5] and the gradient field heatmaps [12]. This leads to a far more informative visualization than any of these approaches taken by themselves offered before. We name this method *Plot of Landscapes with Optimal Trade-offs (PLOT)*. Besides locality, global relations of local optima and respective basins can now be captured in a single PLOT. For two-dimensional problems, PLOT delivers a complete picture of the problem landscape that can be interpreted almost as seamless as a single-objective landscape, merely relating to the multi-objective gradient.

The remainder of this work is organized as follows. Section 2 summarizes the background by first providing fundamental notations and definitions that are needed later, and afterwards discussing the status quo on the visualization of MO landscapes. Section 3 describes the new methodology to determine locally efficient points, while Sect. 4 describes the concept of merging the cost landscape approach and the gradient field heatmaps into PLOT. Finally, Sect. 5 evaluates our proposed PLOT approach by visualizing examples from current benchmark problems before Sect. 6 concludes the paper.

## 2 Background

### 2.1 Preliminaries on Multi-objective Optimization

For this work, we consider continuous MOPs  $f : \mathbb{R}^p \rightarrow \mathbb{R}^k$  with search space parameter  $p$ ,  $k$  objectives and feasible search space  $\mathcal{X} := [\mathbf{l}, \mathbf{u}] \subseteq \mathbb{R}^p$ :

$$f(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x})) \stackrel{!}{=} \min \quad \text{with} \quad l_i \leq x_i \leq u_i, \quad i = 1, \dots, p. \quad (1)$$

The solution of a MOP is the set  $\mathcal{X}^* \subseteq \mathcal{X}$  of Pareto-optimal trade-offs, i.e., all points  $\mathbf{x}^* \in \mathcal{X}$  for which there exist no  $\mathbf{x}' \in \mathcal{X}$  with  $f_i(\mathbf{x}') \leq f_i(\mathbf{x}^*)$  for all  $i = 1, \dots, k$  and  $f_i(\mathbf{x}') < f_i(\mathbf{x}^*)$  for at least one  $i$  (denoted as Pareto set). Thus, the aim of multi-objective (MO) optimization is to find all points in  $\mathcal{X}$  that are not dominated by other points in the decision space. The image  $f(\mathcal{X}^*)$  is called the Pareto front. *Local efficiency of a point* is defined in analogy to locality in single-objective optimization: given a non-empty  $\varepsilon$ -neighborhood  $B_\varepsilon(\mathbf{x}) \subseteq \mathcal{X}$  around  $\mathbf{x}$ , no point  $\mathbf{y} \in B_\varepsilon(\mathbf{x})$  dominates  $\mathbf{x}$  [4]. Extending this definition to a set of locally efficient points, a *local efficient set* is a set of points, which is not dominated by other points in their  $\varepsilon$ -neighborhood [16]. In a somewhat differentiated view, we can further discriminate different local sets, if we consider connected subsets of points as separate local efficient sets like it is done in [13].

For the remainder of this paper, we will focus on two-dimensional bi-objective problems (i.e.,  $p = 2$  and  $k = 2$ ) to enable visual representations of the MOPs. Although this may seem restrictive at first sight, it should be kept in mind that two-dimensional visualizations have substantially contributed to improving our understanding of the algorithmic search behavior in the single-objective case. Also, we will adopt the notion of locally efficient sets from [13] within this work.

The Fritz-John conditions are well known first-order conditions for continuous MOPs [18]. Given a MO function  $f$  defined as above, as well as inactive constraints, a first-order critical point  $\mathbf{x}^*$  fulfills  $\sum_{i=1}^k \lambda_i \nabla f_i(\mathbf{x}^*) = \mathbf{0}$  with  $\lambda_i \geq 0$  for all  $i = 1, \dots, k$  and  $\sum_{i=1}^k \lambda_i = 1$ . Based on this, a MO gradient  $\nabla f(\mathbf{x})$  can be defined, which is zero if these conditions are satisfied, and which points towards a common descent direction of the objectives otherwise ( $-\nabla f(\mathbf{x})$  for minimization). For two objectives, a definition for the MO gradient is given by the sum of the normalized single-objective gradients:

$$\nabla f(\mathbf{x}) = \nabla f_1(\mathbf{x}) / \|\nabla f_1(\mathbf{x})\| + \nabla f_2(\mathbf{x}) / \|\nabla f_2(\mathbf{x})\| \quad (2)$$

Following the MO gradient in a gradient-descent-like manner eventually leads into a local efficient set [9], i.e., a (possibly connected) set of locally efficient points. Note that, if for a point  $\mathbf{x}$  the length of one of its single-objective gradients is zero, i.e.,  $\|\nabla f_1(\mathbf{x})\| = 0$  or  $\|\nabla f_2(\mathbf{x})\| = 0$ , the point fulfills the necessary condition for a local optimum in the single-objective as well as in the MO case. We therefore define  $\nabla f(\mathbf{x}) := 0$  for such points.

## 2.2 Visualization of Continuous MOPs

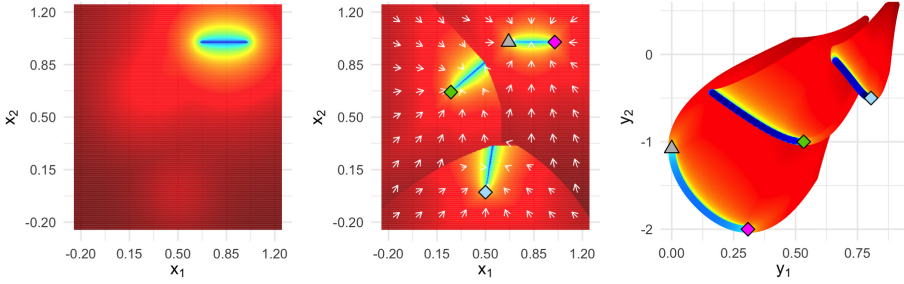
Benchmark problems are usually designed to test the capabilities and limitations of a broad spectrum of (optimization) algorithms [23]. Aside from a pure performance comparison, the insights gained thereof are helpful for designing better algorithms. Here, ‘better’ depends on various aspects such as the application or considered performance measure. Due to the different goals, a variety of test suites have been proposed over the years – ranging from MOP [22], ZDT [25] and DTLZ [7] (which aim at posing challenges for MO evolutionary algorithms), over bi-objective BOB [21] (which extends the gold-standard test suite from single-objective optimization to the bi-objective case), up to more recent benchmark suites like the CEC 2019 test suite [24] (which emphasizes multimodality).

Although most test suites were designed with certain properties in mind, it remains questionable whether the contained MOPs actually meet them. So far, MOPs are predominantly visualized by means of their Pareto sets and/or fronts (see, e.g., [24, 25]). Obviously, this is accompanied by an enormous loss of information, since all non-optimal points, and thus the information contained therein, are ignored. The tools described in [19, 20] offer a slight improvement over the very limited view at Pareto optimal points. However, the approaches described therein, such as the *prosection* method, mainly focus on a dimensionality reduction of the objective space – and thus do not permit drawing conclusions about the effects of the search space parameters on the objectives of the MOP.

To the best of our knowledge, there exist only two visualization techniques which provide a joint view at decision and objective space (and thus are helpful for engineering better algorithms and benchmark problems): the *cost landscapes* by Fonseca [5] and the *gradient field heatmaps (GFH)* by Kerschke and Grimme [12, 14]. Both approaches have in common that they depict the decision space of two-dimensional MOPs and scalarize the information of the MOP’s multiple objectives in a single height value (per point from the decision space). For both approaches, the decision space is discretized into a rectangular grid, and the grid’s resolution naturally impacts the quality of the visualizations.

The height of a cost landscape is given by the so-called Pareto ranking, i.e., an integer value that gives the amount of points from the (discretized) decision space dominating the current point. Due to the usage of the (global) Pareto ranking, cost landscapes focus on *global* optimality and thus are only able to reveal local structures, if the local fronts are close to parts of the Pareto front.

In contrast, the height of the GFHs is based on a MOP’s gradient vector field. More precisely, for each point of the grid, the single-objective gradients (pointing to the closest optimum of the respective objective) are approximated



**Fig. 1.** Exemplary comparison of the cost landscape (left) and the gradient field heatmap (middle) based on the bi-objective SGK function. The right image shows the objective space for the GFH and thus helps identifying the superposition and relationships of the three attraction basins (incl. their associated efficient sets). The colors indicate the respective heights and change gradually from red (max.) to blue (min.). (Color figure online)

using Eq. 3, and afterwards combined into a MO gradient as defined in Eq. 2. Then, for each grid point, the gradient-based path towards the closest local efficient set is computed and the lengths of the MO gradients along the path are cumulated, defining the height of the GFH [12]. Constructing the GFHs based on the paths towards the closest *local* efficient set automatically provides insights into the local structure of the given MOP, as it depicts the attraction basins, as well as the local efficient sets contained therein. However, the focus on locality comes with the drawback that global relationships are hardly visible.

Figure 1 provides an exemplary visual comparison of the cost landscape and the GFH approach based on the bi-objective SGK function<sup>1</sup>, which combines a unimodal and a trimodal sphere function. The single-objective local optima are depicted by a grey triangle (for  $f_1$ ), as well as green, blue, and pink diamonds (for  $f_2$ ), respectively. The left image shows the corresponding cost landscape, in which the MOP's Pareto set is clearly visible – in contrast to the local efficient sets, whose location can only be guessed by the shading of colors. The GFH approach, given in the second image, shows the three attraction basins formed by the three competing optima of  $f_2$ , along with the corresponding vector field of MO gradients (white arrows). Moreover, all three (local) efficient sets are visible, and one can also identify two of them being non-global as their sets – which start in the blue and green diamonds, resp. – are abruptly cut by ridges between the current and the superseding attraction basins. While such a global ranking of the three efficient sets can be derived manually for this simple scenario, it is hard to realize for more complex MOPs (e.g., see bottom row of Fig. 4).

<sup>1</sup>  $f(x_1, x_2)$  with  $f_1(x_1, x_2) = 1 - (1 + 4 \cdot ((x_1 - 2/3)^2 + (x_2 - 1)^2))^{-1}$  and  $f_2(x_1, x_2) = 1 - \max\{g_1, g_2, g_3\}$ , whose subfunctions  $g_1, g_2$  and  $g_3$  are defined as  $g_i(x_1, x_2, h, c_1, c_2) = h / (1 + 4 \cdot ((x_1 - c_1)^2 + (x_2 - c_2)^2))$  with  $h = 1.5, c_1 = 0.5, c_2 = 0$  (for  $g_1$ ),  $h = 2, c_1 = 0.25, c_2 = 2/3$  (for  $g_2$ ), and  $h = 3, c_1 = 1 = c_2$  (for  $g_3$ ).

### 3 Identification of Locally Efficient Points

Locally efficient points are an important part of MOP landscapes as they indicate where local Pareto fronts (or sets) are located, and thus, where local search strategies might get stuck [6]. However, state-of-the-art visualization approaches either do not feature them at all (e.g., the cost landscapes [5]), or only show their locations indirectly (e.g., the gradient field heatmaps [12]). Only when the location of the local efficient sets is known analytically for specific test problems – like for DTLZ [7] or MMF [24] – they are represented in some visualizations.

Here, we present an approach based on the estimated gradients of the MOP and the stability of the MO gradient vector field to locate all locally efficient points for a continuous MOP. We begin by detailing our computational approach for approximating the function and its derivatives, followed by a description of first- and second-order optimality conditions for locally efficient points and how they were implemented.

#### 3.1 Computational Approach

As continuous functions can in principle be evaluated in infinitely many different points, an approximation of the function based on a finite set of evaluated points is required. For this purpose, we evaluate points on a rectangular grid of coordinates  $(x_1^{j_1}, x_2^{j_2})$ . Those coordinates are aligned equidistantly with a number of steps  $n_1, n_2 \in \mathbb{N}$  and step sizes  $s_i = (u_i - l_i) \cdot (n_i - 1)^{-1}$  per dimension, i.e.,  $x_i^{j_i} = l_i + (j_i - 1) \cdot s_i$ , with  $i = 1, 2$  and  $j_i = 1, \dots, n_i$  where  $x_i^{j_i}$  denotes the coordinate of the  $j_i$ -th grid point in the  $i$ -th dimension. Next, the rectangular grid of points is created by taking the cross-product of the one-dimensional coordinates  $x_i^{j_i}$ . The function  $f$  is then evaluated for each of the points from the grid.

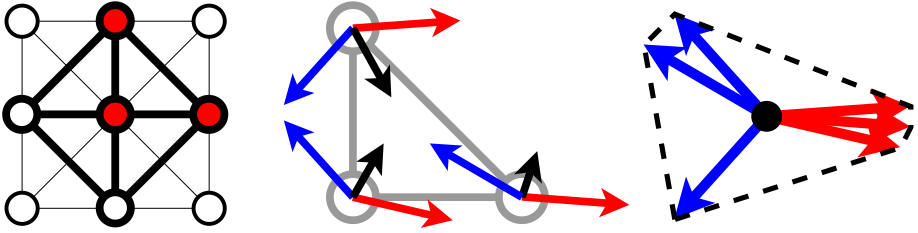
In each grid point, the respective derivative is approximated (per objective) using the finite differences method based on the neighboring coordinates of the point at hand. On the decision boundary, forward- and backward-differences are taken respectively, while the interior points are evaluated using central differences. With the function values of  $f$  at grid point  $(x_1^{j_1}, x_2^{j_2})$  denoted as  $f(x_1^{j_1}, x_2^{j_2})$ , the partial derivative of  $f$  with regard to  $x_1$  is then estimated by:

$$\frac{\partial}{\partial x_1} f(x_1^{j_1}, x_2^{j_2}) \approx \begin{cases} \frac{1}{2s_1} \cdot \left( f(x_1^{j_1+1}, x_2^{j_2}) - f(x_1^{j_1-1}, x_2^{j_2}) \right), & \text{for } 1 < j_1 < n_1 \\ \frac{1}{s_1} \cdot \left( f(x_1^{j_1+1}, x_2^{j_2}) - f(x_1^{j_1}, x_2^{j_2}) \right), & \text{for } j_1 = 1 \\ \frac{1}{s_1} \cdot \left( f(x_1^{j_1}, x_2^{j_2}) - f(x_1^{j_1-1}, x_2^{j_2}) \right), & \text{for } j_1 = n_1 \end{cases} \quad (3)$$

Derivatives for  $x_2$  are calculated analogously. Compared to approximations with smaller step sizes, we did not observe noticeable changes in the visualization.

#### 3.2 First-Order Conditions

Although the MO gradient is capable of capturing the local efficiency of a point in the decision space, it is not sufficient on its own to capture all critical points



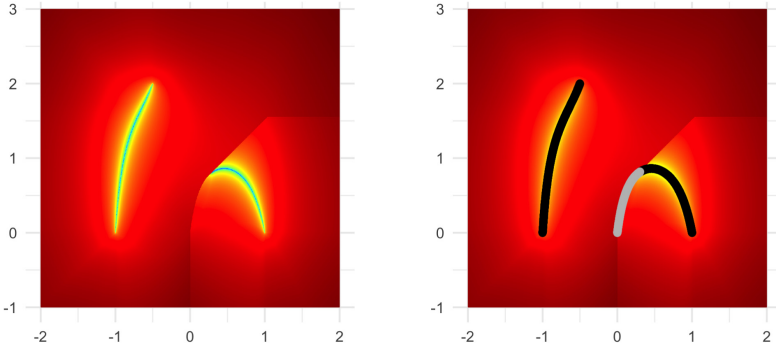
**Fig. 2.** Left: Example of four triangular neighborhoods considered while evaluating for critical points. The red nodes are an example for one particular neighborhood. Middle: For such a triangular neighborhood of points, considering the MO gradients (black), no critical points would be detected, as all of them point to the right as a common descent direction. However, it would be reasonable to expect that due to the changes in direction of the single-objective gradients (red and blue respectively, pointing towards descent directions), there would be a local efficient set in between. Right: Our approach jointly considering the combination of all single-objective gradients from all corner points correctly reports the neighborhood to contain a locally efficient point. (Color figure online)

while approximating them in the grid-based decision space. In particular, if one of the single-objective gradients changes much faster than the other in the neighborhood of a local efficient set, the MO gradient alone may misleadingly fail to recognize some of the critical points (as shown schematically in Fig. 2).

Thus, we present an approach that improves the status quo in detecting all critical points within the MOP's grid. We consider triangular neighborhoods of grid points and jointly consider the gradients of all constituent single-objective functions. If the convex hull of the gradients encloses the origin (see right image of Fig. 2), we presume a critical point *somewhere* in the interior of the triangle.

An illustration of the neighborhood, which is used for detecting the critical points, is given in the left image of Fig. 2. For each grid point, four triangular neighborhoods are evaluated. Each of those triangles consists of the point itself, as well as one horizontal and one vertical neighbor of that point. If a critical point is detected for a triangle, all of its corner points are considered being critical.

In addition, points along the decision boundary require special attention. So far, in the gradient field heatmaps, all (boundary) points for which the MO gradient points “out” of the feasible decision space were considered locally efficient. However, even in that case, it might still be possible to follow a descent direction for all objectives when moving along the decision boundary. We resolve this issue by considering boundary points, their adjacent boundary points and their common descent directions along the decision boundary all together. If no common descent direction for all objectives is found, the respective pair of boundary points is considered critical. Further, we rotate the MO gradient at these points to point into the common descent direction along the boundary, which is helpful for further visualizations with the gradient field heatmap (see Sect. 4).



**Fig. 3.** GFH visualization and location of all interior critical points for the Aspar function  $f(x_1, x_2) = (x_1^4 - 2x_1^2 + 2x_2^2 + 1, (x_1 + 0.5)^2 + (x_2 - 2)^2)$ . Even in this very simple problem, some critical points (gray) do not belong to the efficient sets (black) but are part of the landscape's ridges, emphasizing the need for a second-order criterion.

When only aiming at the identification of locally efficient points in the decision space, one needs to be aware that not all critical points are necessarily locally efficient. An example of this is given in Fig. 3, which shows the GFH of a simple MOP and all of its critical points. Note that some of the critical points do not belong to a local efficient set, but are rather part of the ridges between two adjacent basins of attractions. To reliably extract the locally efficient points from the set of critical points, a second-order condition is required.

### 3.3 Second-Order Conditions

Analogously to single-objective functions satisfying first-order optimality conditions, critical points in the multi-objective sense cannot just be local minima, i.e., locally efficient points, but also local maxima, as well as saddle points.

This highlights the need for the consideration of a second-order condition to distinguish identified critical points into locally efficient points and others. There are second-order conditions for the continuous case [18], however, using the grid approximation required for our approach, these proved to be too unstable for efficient use to discern between the different types of critical points. Motivated by the gradient field heatmaps, which indicate that the MO gradient captures the local search behaviour well, we derive our second-order condition based on properties of the MO gradient vector field.

The MO gradient defines a vector field over the decision space that can be analyzed w.r.t. its stability at the critical points. A point is considered asymptotically stable, if after a small perturbation, following the vector field to the closest critical point, one stays within a small neighborhood of the original point. This property is well studied in the field of autonomous differential equations and can be analyzed by a linear approximation of the vector field at the critical



point using its Jacobian [2].<sup>2</sup> If the real part of all (potentially complex-valued) eigenvalues of the Jacobian is negative, the point is considered stable.

In the MO gradient field, however, we generally deal with degenerated critical points, for which (at least) one eigenvalue is zero, associated with the eigenvectors pointing along the local efficient set. This can pose problems with the numeric approximation that we require for our approach. Luckily, for the 2D case it is sufficient to consider the trace of the Jacobian, also known as the divergence in the context of vector fields, to determine asymptotic stability. Intuitively, the divergence is a measure for the “ingoingness” or “outgoingness” of the vector field at a given point, and it is numerically more stable and efficient to calculate than computing all eigenvalues of the Jacobian. Thus, the divergence of a 2D vector field  $\mathbf{V} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  with  $\mathbf{V}(\mathbf{x}) = (V_{x_1}(\mathbf{x}), V_{x_2}(\mathbf{x}))$  is given by:

$$\mathbf{div}(\mathbf{V}(\mathbf{x})) = \frac{\partial}{\partial x_1} V_{x_1}(\mathbf{x}) + \frac{\partial}{\partial x_2} V_{x_2}(\mathbf{x}). \quad (4)$$

In summary, for minimization as defined in Eq. 1, if an interior critical point  $\mathbf{x}$  fulfills  $\mathbf{div}(-\nabla f(\mathbf{x})) < 0$ , it is a stable critical point in the MO gradient field, and thus locally efficient. Note that in the degenerated case, where all points in a given neighborhood are locally efficient, the divergence is zero.

Thus, to assess whether a critical point is locally efficient, we consider the divergence of each set of points that were jointly considered critical w.r.t. the first-order conditions (see Sect. 3.2). Only if all three evaluated points have non-positive divergence, we regard them as locally efficient. The divergence for each grid point is estimated using the grid-based finite differences method (see Eq. 3).

Again, the critical points along the decision boundary require special treatment. Here, we do not use the divergence to distinguish different types of critical points. Instead, only if the MO gradients of the considered boundary points are pointing “outwards” or along the decision boundary, a set of critical points is considered locally efficient.

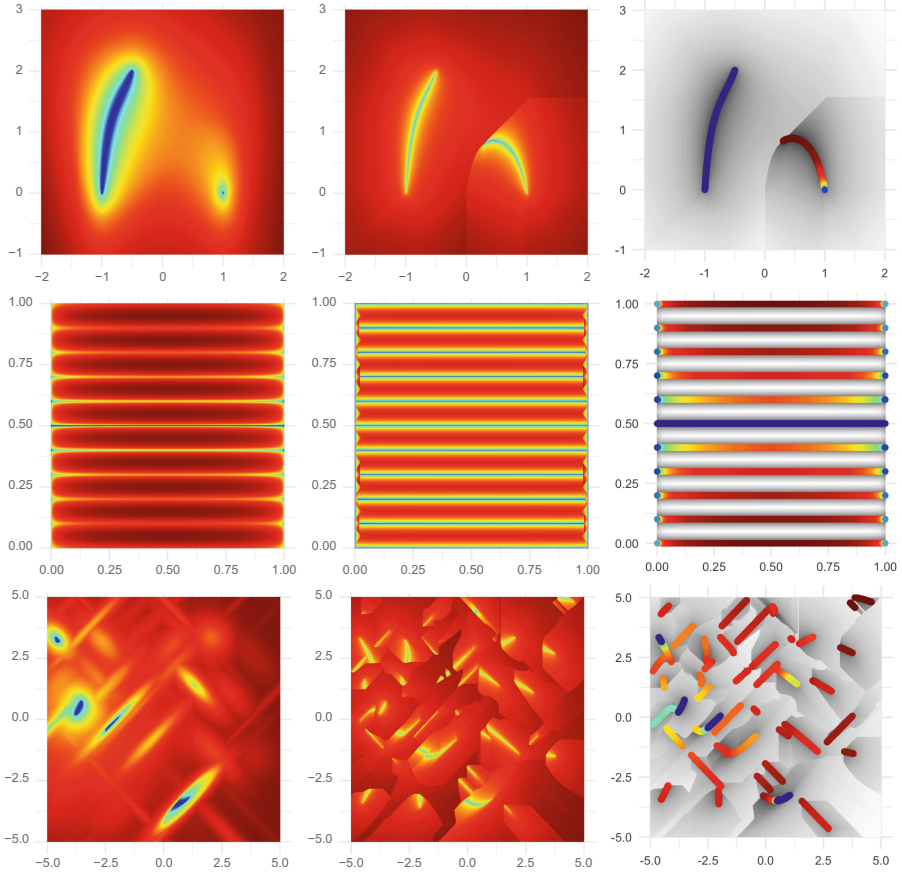
## 4 Visualizing Local and Global Structures of MOPs

The previous section introduced a novel and reliable approach to approximate the location of all locally efficient points within the MOP’s continuous decision space (based on a rectangular grid of evaluated points). This explicit knowledge of the location of the locally efficient points not only allows us to show them in the decision space, but also to extract information about their relative dominance relationship. This means, we utilize Pareto ranking, which also serves as the basis for the cost landscapes [5], but limit ourselves to the locally efficient points – resulting in an enormous speed-up compared to a ranking of *all* grid points.

Ultimately, this leads to a unique visualization that not only shows the locations of locally efficient solutions, but also provides information about their

<sup>2</sup> This presumes that the MO gradient field can be approximated by a linear function in the considered point. For differentiable MOPs this is a reasonable assumption, but we observed that it works well for our approach in general.

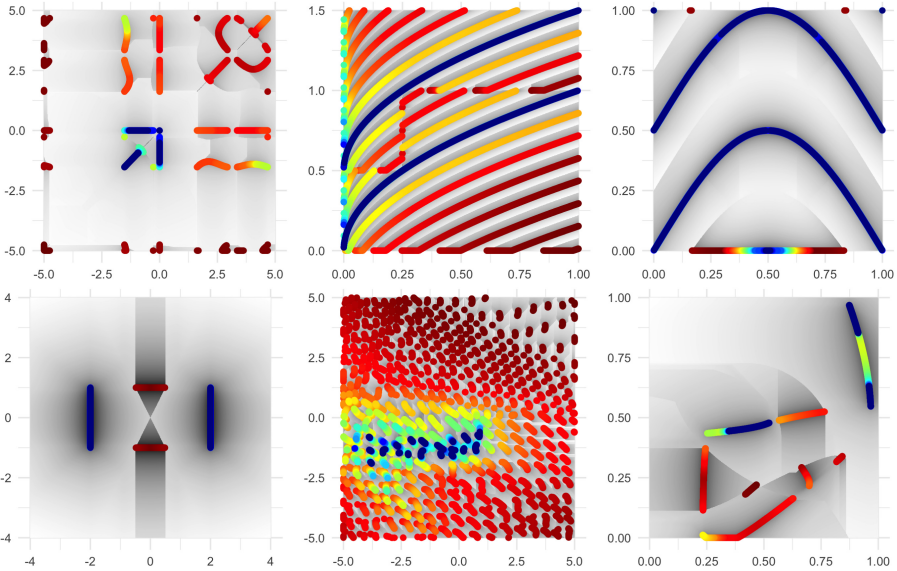




**Fig. 4.** Comparison of the cost landscape, GFH and PLOT (left to right) on the two-dimensional bi-objective Aspar, DTLZ1 and bi-objective BBOB (FID = 10, IID = 1) functions (top to bottom). Due to the computational overhead involved in computing the domination counts, the cost landscape is calculated with only 500 grid points per dimensions, while the GFH and PLOT use a resolution of 1,000 points per dimension.

global optimality. In addition, we enhance our visualization with a gray-scaled version of the corresponding GFH in the background, which preserves additional information about the basins of attraction (e.g., their shapes and sizes). Along the boundary points, we modify the MO gradient as described in Sect. 3.2. Also, all locally efficient points determined by our more stable detection method (see Sect. 3) can automatically be reused “for free” within the generation of the GFHs. Both modifications further improved the visualization quality of the GFHs.

Figure 4 provides a visual comparison of the current state-of-the-art visualization techniques – cost landscapes (left column) and GFHs (middle column) – and our proposed PLOT approach (right column) based on three exemplary MOPs: the simple Aspar function (top row) from Fig. 3, the well-established



**Fig. 5.** Exemplary PLOTs for various continuous MOPs (left to right, top to bottom): Kursawe [15], MMF3 and MMF14a from the CEC 2019 test suite [24], a MinDist function with centers  $(-2, -1)$ ,  $(2, 1)$  and  $(-2, 1)$ ,  $(2, -1)$  [17], a bi-objective BBOB (FID: 42, IID: 1) [21], and a bi-objective function generated using the MPM2 generator (with parameters  $(3, 2, \text{random}, 4)$  and  $(3, 2, \text{random}, 8)$ ) [13]. All plots were generated using an equidistant grid in the decision space with a resolution of  $1,000 \times 1,000$  points.

DTLZ 1 [7] (middle row), and the 10th function from the rather recent bi-objective BBOB test suite [21]. Noticeably, for the latter two problems, GFH has problems in identifying some critical points correctly and mistakenly shows some points along the boundary as locally efficient. On the other hand, the cost landscape approach has problems with *local* efficient sets and can at most identify the *location* of some of them – as long as their fronts are close to the global Pareto front(s). PLOT combines the global view of the cost landscapes with the local information of the GFH, and thus provides a much more informative depiction of the locally efficient solutions.

Ultimately, this results in a single *Plot of the Landscape with Optimal Trade-offs* (PLOT), which jointly illustrates three types of landscape characteristics: (1) *local efficient sets*, (2) the *global optimality* of their respective solutions, and (3) the *basins of attraction* associated with the respective efficient sets.

Our R-package `moPLOT`, which has been used to generate all visualizations in this paper, is available on GitHub: <https://github.com/kerschke/moPLOT>. Further resources and information can be found on our project’s website on multimodal multi-objective optimization: <https://mo-opt.github.io>.

## 5 Observations

We provide PLOT visualizations for a selection of further benchmark functions in Fig. 5. Many MOPs that were designed with multimodality in mind reveal very simple structures in the decision space. Notably, the PLOTs show peculiarities in the definition of some functions that were designed with a focus on multiple global Pareto sets. These can lead to unintended locally and globally efficient solutions along the boundary (MMF14a) and glaring cuts in the landscape of the local efficient sets (MMF3). Otherwise, many MOPs have even simpler landscape structures only containing few local efficient sets in general (MinDist). Only few of the MOPs show interactions between the objectives, which lead to a disconnected global Pareto set (i.e., it is distributed over multiple local efficient sets). This can, e.g., be seen in the bi-objective BBOB and MPM2 functions.

Further note that the location of locally efficient points along the decision boundary implies that in general an unconstrained locally efficient set would be found outside of the feasible decision space. This can be observed in the depicted Kursawe, MMF and MPM2 functions.

The bi-objective BBOB function shows a very complex landscape with many locally efficient solutions. In fact, its sets cover the majority of the decision space and thereby reveal the limitations of PLOT. However, such extremely multimodal MOPs are challenging for any visualization method. Also, even for that very extreme problem, PLOT reliably visualized the MOP's global structure.

## 6 Conclusions

Visualizing an optimization problem's landscape is very useful when studying its properties, or the search behavior of the optimizers operating on it. In MO continuous optimization, however, there exist hardly any meaningful visualization methods, with the consequence that MOPs are primarily treated as black-boxes.

We present a novel approach for the numerical approximation of locally efficient points in the decision space of continuous MOPs. This new information was then integrated into PLOT – our new method for the visualization of bi-objective two-dimensional MOPs. Our approach can visualize local and global efficient sets, as well as their basins of attraction. Thereby, it enables a visualization of MOPs that encompasses information comparable to visualizations available for single-objective functions. We successfully apply our approach to a wide variety of benchmarking functions and often reveal very simple landscape properties. As with previous MO visualization techniques, we hope to inspire further progress in understanding the landscapes of existing benchmarking functions, the design of new benchmarks, as well as the development of novel algorithmic ideas.

It should be noted that the definitions for the MO gradient can be extended to an arbitrary number of dimensions and objectives [8]. Likewise, our approach for identifying critical points as well as our second-order criterion, which is based on the stability of the MO gradient field, can easily be adapted to cope well with

higher-dimensional MOPs. Thus, our proposed approach provides the fundamentals for an extension towards visualizing decision spaces of 3-dimensional MOPs. To the best of our knowledge, this is not yet available beyond the visualization of Pareto sets, or analytically known local efficient sets. An extension to 3D decision spaces would also enable more detailed investigations of the properties of 3-objective MOPs. This is currently not yet feasible, as their counterparts with 2D decision spaces in general contain degenerated critical points.

Further, it can be noted that our approach supports studying selected regions of interest in the landscape. This can be effectively achieved by *zooming* into the PLOT and supports the visualization of particularly complex MOPs. Another possible extension that aims at improving the visualization quality of our PLOT would be a dynamic resampling strategy around the identified critical points, increasing the accuracy of the approximation of the locally efficient points.

## References

1. Beyer, H.G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001). <https://doi.org/10.1007/978-3-662-04378-3>
2. Blanchard, P., Devaney, R., Hall, G.: Differential Equations. Cengage Learning, Boston (2012)
3. Coello Coello, C.A., van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, Boston (2007). <https://doi.org/10.1007/978-0-387-36797-2>
4. Custódio, A.L., Madeira, J.F.A.: MultiGLODS: global and local multiobjective optimization using direct search. *J. Global Optim.* **72**(2), 323–345 (2018)
5. da Fonseca, C.M.M.: Multiobjective genetic algorithms with application to control engineering problems. Ph.D. Thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, September 1995
6. Deb, K.: Multi-objective genetic algorithms: problem difficulties and construction of test problems. *Evol. Comput. (ECJ)* **7**(3), 205–230 (1999)
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. Advanced Information and Knowledge Processing*, 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)
8. Désidéri, J.A.: Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathématique* **350**(5–6), 313–318 (2012)
9. Grimme, C., Kerschke, P., Emmerich, M.T.M., Preuss, M., Deutz, A.H., Trautmann, H.: Sliding to the global optimum: how to benefit from non-global optima in multimodal multi-objective optimization. In: *AIP Conference Proceedings*, pp. 020052-1-020052-4. AIP Publishing (2019)
10. Grimme, C., Kerschke, P., Trautmann, H.: Multimodality in multi-objective optimization – more boon than bane? In: Deb, K., et al. (eds.) *EMO 2019. LNCS*, vol. 11411, pp. 126–138. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-12598-1\\_11](https://doi.org/10.1007/978-3-030-12598-1_11)
11. John, F.: Extremum problems with inequalities as subsidiary conditions. In: Giorgi, G., Kjeldsen, T.H. (eds.) *Traces and Emergence of Nonlinear Programming*, pp. 197–215. Springer, Basel (2014). [https://doi.org/10.1007/978-3-0348-0439-4\\_9](https://doi.org/10.1007/978-3-0348-0439-4_9)

12. Kerschke, P., Grimme, C.: An expedition to multimodal multi-objective optimization landscapes. In: Trautmann, H., et al. (eds.) EMO 2017. LNCS, vol. 10173, pp. 329–343. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-54157-0\\_23](https://doi.org/10.1007/978-3-319-54157-0_23)
13. Kerschke, P., et al.: Towards analyzing multimodality of continuous multiobjective landscapes. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 962–972. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_90](https://doi.org/10.1007/978-3-319-45823-6_90)
14. Kerschke, P., et al.: Search dynamics on multimodal multi-objective problems. *Evol. Comput. (ECJ)* **27**, 577–609 (2019)
15. Kursawe, F.: A variant of evolution strategies for vector optimization. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 193–197. Springer, Heidelberg (1991). <https://doi.org/10.1007/BFb0029752>
16. Liefvooghe, A., López-Ibáñez, M., Paquete, L., Verel, S.: Dominance, epsilon, and hypervolume local optimal sets in multi-objective optimization, and how to tell the difference. In: Proceedings of the 20th Annual Conference on Genetic and Evolutionary Computation (GECCO), vol. 18, pp. 324–331. ACM, Kyoto (2018)
17. Maree, S.C., Alderliesten, T., Bosman, P.A.N.: Real-valued evolutionary multimodal multi-objective optimization by hill-valley clustering. In: Proceedings of the 21st Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 568–576. ACM (2019). <https://doi.org/10.1145/3321707.3321759>
18. Miettinen, K.: *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management Science, vol. 12. Springer, Boston (1998)
19. Tušar, T.: *Visualizing Solution Sets in Multiobjective Optimization*. Ph.D. thesis, Jožef Stefan International Postgrad. School (2014)
20. Tušar, T., Filipič, B.: Visualization of pareto front approximations in evolutionary multiobjective optimization: a critical review and the projection method. *IEEE Trans. Evol. Comput. (TEVC)* **19**(2), 225–245 (2015)
21. Tušar, T., Brockhoff, D., Hansen, N., Auger, A.: COCO: the bi-objective black box optimization benchmarking (bbob-biobj) test suite. arXiv preprint [abs/1604.00359](https://arxiv.org/abs/1604.00359) (2016)
22. van Veldhuizen, D.A.: *Multiobjective evolutionary algorithms: classifications, analyzes, and new innovations*. Ph.D. thesis, Faculty of the Graduate School of Engineering of the Air Force Institute of Technology, Air University, June 1999
23. Whitley, L.D., Mathias, K.E., Rana, S.B., Dzubera, J.: Building better test functions. In: Proceedings of the 6th International Conference on Genetic Algorithms (ICGA), pp. 239–247 (1995)
24. Yue, C., Qu, B., Yu, K., Liang, J., Li, X.: *A Novel Scalable Test Problem Suite for Multimodal Multiobjective Optimization*. Swarm and Evolutionary Computation (2019)
25. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. *Evol. Comput. (ECJ)* **8**(2), 173–195 (2000). <https://doi.org/10.1162/106365600568202>

# **Multi-objective Optimization**



# On Sharing Information Between Sub-populations in MOEA/S

Lucas de Almeida Ribeiro<sup>1,2(✉)</sup>, Michael Emmerich<sup>3</sup>,  
Anderson da Silva Soares<sup>1</sup>, and Telma Woerle de Lima<sup>1</sup>

<sup>1</sup> Instituto de Informática, Universidade Federal de Goiás, Goiânia, Brazil

<sup>2</sup> Instituto Federal de Goiás, Goiânia, Brazil

lucas.ribeiro@ifg.edu.br

<sup>3</sup> Multiobjective Optimization and Decision Analysis Group, LIACS,  
Leiden University, Niels Bohrweg 1, 2363CA Leiden, The Netherlands

**Abstract.** This work investigates the effect of information exchange in decomposition methods that work with multi-membered populations as sub-problems. As an algorithm framework, we use the *Multi-objective Evolutionary Algorithm based on Sub-populations* (MOEA/S). This algorithm uses parallel sub-populations that can exchange information via migration and/or recombination. For this work, each sub-population is constructed by a few weighted utility functions, grouped by distance between their weighting vectors. The question investigated in this paper is: How is the distance between sub-populations and the mechanism of information exchange influencing the performance of MOEA/S? The study considers two ways of transferring information: (1) migration of individuals, (2) recombination using parents from two different sub-populations. A matrix describing the linkage patterns between sub-populations governs migration and recombination mechanisms. This work conducts a systematic study using the multi-objective knapsack problem (MOKP) and multi-objective traveling salesperson (MOTSP) for two and three objectives test problems. The results motivated a restriction policy for sharing information. We compare an algorithm using this policy with other state-of-the-art MOEAs, including NSGA III, MOEA/D, and the previous version of MOEA/S.

**Keywords:** Decomposition-based multi-objective optimization · Cellular genetic algorithm · Sub-population based MOEAs · Migration operator · MOEA/S

## 1 Introduction

Multi-objective optimization is the task of finding solutions in a search space with the best quality concerning multiple objective functions. Decomposition-based multi-objective evolutionary optimization deals with these problems by defining a collective, population-based, search. The main idea of decomposition-based methods is to decompose the problem into sub-problems targeting

different regions on the Pareto front. The search is done simultaneously and while continuously exchanging information between the sub-populations [7, 10, 16].

One crucial matter in decomposition-based methods for multi-objective problems is how to exchange information among sub-populations to speed up the convergence to the Pareto front based on shared search and to improve Pareto front coverage. The mechanism used for this purpose covers mating individuals from different sub-populations and migrating individuals among sub-populations.

Zhang et al. propose a straightforward and commonly used implementation of decomposition-based methods. [29], called the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D). MOEA/D explicitly decomposes the multi-objective optimization problem into  $N$  (size of the population) scalar optimization sub-problems. Then, MOEA/D optimizes the  $N$  sub-problems simultaneously. Each sub-problem is optimized by only using information from its neighboring sub-problems, where the neighborhood is defined a priori based on the neighborhood of reference directions. MOEA/D uses the same aggregation function for all sub-problems; a unique combination of weighting vectors defines each sub-problem. Thus, the neighborhood of a sub-problem is assumed as the neighborhood of its weighting vector.

Another family of decomposition-based methods, used in the literature, splits the population into several sub-populations where each one of them can use a distinct multi-objective strategy [2–5, 9, 11, 22–25]. In this work, we formulate this approach as MOEA/S (Multi-objective Evolutionary Algorithm based on Sub-populations). Although this method has obtained good results, only a very general rule is defined for exchanging information among sub-populations. In contrast to MOEA/D, which uses a neighborhood definition, the simple rule is that every sub-population is exchanging information with every other sub-population at the same rate.

A commonly applied decomposition-based algorithm is the new version of the Non-dominated Sorting Genetic Algorithm (NSGA-III) [7], which places reference points on a simplex the size of which is adapted according to the current best information on the boundaries of the true Pareto front. NSGA-III does not take into account neighborhood among sub-populations in exchanging information, although it uses a niching method for selection.

Murata et al. [17] studies the behavior of restricting mating and replacement based on the neighborhood in a cellular version of Multi-Objective Genetic Algorithm (MOGA). Using different neighborhood sizes, they conclude that neither the closest neighbor nor the farthest sub-population is the best option in sharing information (using mating and replacement). Whether such strategies are better than not sharing information remains open in their analysis.

Ishibuchi et al. [12] studies the use of different size of neighborhoods ( $T_s$ ) for mating and replacement selection in many-objective problems in MOEA/D. They obtain as a result that an appropriate specification of the two neighborhoods is problem-dependent. But in all the cases, a small neighborhood for replacement might lead to a well-distributed Pareto front, in many-objective problems. Wang et al. [28] suggest new replacement strategies where the solution is compared in all weight vectors and replace the solutions in the neighborhood of its best suitable weight vector.



These previous works have shown good results in exploring the relation between neighborhood size and performance. Thus, the exploitation of information obtained in similar sub-problems is useful to improve the speed of convergence to the Pareto front. Nevertheless, these results are not extended for non-cellular approaches, since the neighborhood of one sub-population with multiple individuals is not as easy to define as the neighborhood of a singleton sub-population as it is used in MOEA/D and c-MOGA.

Our work will investigate the effectiveness of exchanging information between sub-populations based on their distance using the non-cellular decomposition method, MOEA/S. MOEA/S decomposes a problem into  $N$  scalar optimization sub-problems. Each sub-problem is solved simultaneously using a population-based multi-objective evolutionary algorithms (MOEA) - in accordance with previous research, the  $N$  populations used by these MOEAs will be called sub-populations. In contrast to MOEA/D, each sub-population can consist, in general, of more than one individual. In order to exchange information, a connection between sub-populations must be established. The connection definition is based on the distance between the centroids of the sub-populations. Different operators for information exchange will be compared for MOEA/S in this paper.

This paper is organized as follows: Sect. 2 introduces MOEA/S and explains the main conceptual ideas of the method. Section 3 shows experimental setup; Sect. 4 explores the results on test problems and in Sect. 5 the paper is concluded with a summary of our main findings<sup>1</sup>.

## 2 Methods

### 2.1 MOEA/S Algorithm

The Multi-objective Evolutionary Algorithm based on Sub-populations (MOEA/S) is a decomposition-based MOEA which supports non-singleton sub-population based MOEAs to solve, simultaneously, the sub-problems of a problem decomposition. In principle, each sub-problem can be solved by a different MOEA, in terms of the selection processes. A global ‘master algorithm’ controls the interplay and information exchange between the MOEAs that address sub-problems.

MOEA/S splits the (global) population into a constant number of  $\mu$  sub-populations, which are managed by different selection processes. One can design a process using Pareto based strategies, indicator-based methods, scalarization based algorithms, and so on. To be eligible as a MOEA, for solving a sub-problem in MOEA/S, the selection operator must obey a particular framework. The framework interface requires: limited population size of at most  $N_{limit}^i$  or  $N^i$  ( $i = 1 \dots \mu$ ) individuals; moreover, it must define a method for mating selection; and a method for environmental (or truncation) selection.

In MOEA/S, it is an essential principle that sub-problems are not solved independently, but in general, it is possible to exchange information between

---

<sup>1</sup> Additional data is made available in the web-repository <http://moda.liacs.nl>.

sub-populations. The idea is, roughly speaking, to exploit synergies between different sub-problem solution processes.

Sub-populations can exchange information in two stages: the first stage, the mating stage, is using the mating operator. Via a mating matrix, a coupling between the sub-populations is established. The rows (index  $i \in \{1, \dots, \mu\}$ ) indicate the populations in the mating pool (deme) of the  $i$ -th sub-population  $P_i$ . Secondly, in the migration stage, a migration matrix (destination matrix) is set up to decide to which other populations, individuals of sub-population  $P_i$  can migrate (for each  $i = 1, \dots, \mu$ ).

In summary, MOEA/S contains a list of sub-populations,  $(P_1, \dots, P_\mu)$ , each of which containing a limited number  $N_i, i = 1, \dots, \mu$  of individuals, a method for selecting parents (mating selection), and a method for discarding or selecting individuals; a population  $(P = \bigcup_{i=1}^{\mu} P_i)$ ; a structure which stores the connections between sub-populations for mating (deme -  $M \in \mathbb{B}^{\mu \times \mu}$ ); and for destination sub-populations (destination matrix -  $D \in \mathbb{B}^{\mu \times \mu}$ ) used in the environmental selection; environmental selection maintains an adjacency matrix to associate each individual with its sub-populations (adjacency matrix -  $A \in \mathbb{B}^{\xi \times \mu}$ , where  $\xi = |P|$ ); a method for creating new individuals; and a method for initialization.

The MOEA/S procedure starts with the initialization of the sub-population structure. This phase distributes all individuals from the initial population into the sub-populations. The evolutionary loop consists of: (1) *selection of the parents* (or mating pool), (2) *creating new individuals by mutation and crossover operators*, (3) *environmental selection*, (4) *migration*. In more detail:

- (1) select  $p1$  as the first sub-population. Then, the first parent ( $s^{p1}$ ) is selected from a designated sub-population ( $P_{p1}$ ) and the second one ( $s^{p2}$ ) is selected from a population  $P_{p2}$ . Index  $p2$  is chosen according to the mating pool of  $P_{p1}$  defined by the mating matrix.
- (2) generate new individuals ( $s^{new1}$  and  $s^{new2}$ ) by crossover and mutation operators from  $s^{p1}$  and  $s^{p2}$
- (3) evaluates the new individuals in the sub-populations of their parents.
- (4) migrate the new individuals to the destination sub-populations of their parents' sub-populations, according to the migration matrix.

An individual  $s^{new}$  is accepted in the sub-population  $P_i$  if the size of  $P_i$  satisfies  $|P_i| < N_i$  or in case  $|P_i| = N_i$  it can be chosen by the selection of the destination sub-population, for instance, a tournament selection. In the latter case, an individual of the destination sub-population  $s^{new}$  replaces  $s^{old}$  in  $P_i$ .

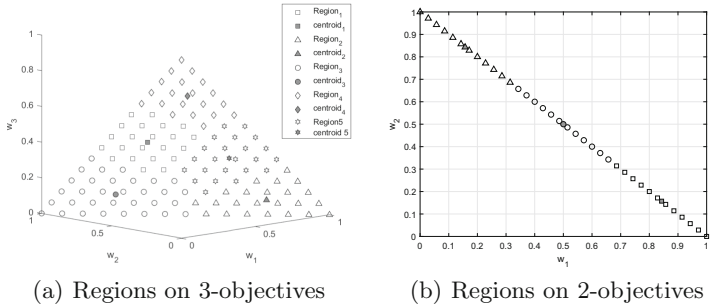
## 2.2 MOEA/S Instance

Next, we will discuss the specific instance of MOEA/S used in this paper, which targets different regions on the Pareto front by different sub-populations:

First,  $N$  (size of the population) scalar optimization functions are defined differed by their weighting vector, as in MOEA/D. Each one of these functions is associated with one sub-problem ( $R^i, 1, \dots, N$ ). Then, sub-problems are clustered

into sub-populations, further called Region sub-populations ( $R^i, 1, \dots, \mu$ ), using their weighting vectors. Thus, each individual in the population is associated with one sub-problem, and its fitness is assessed by one utility function.

Figure 1 exemplifies how weighting vectors distributions for two and three-objective spaces are spatially located. Figure 1(a) presents the regions on 3-objective spaces with 120 points and 5 Regions. Figure 1(b) presents the regions on 2-objective spaces, with 20 points and 3 regions. The clustering method is  $k$ -means clustering [1] which groups each sample around a centroid, where here the number of clusters corresponds to the number of regions.



**Fig. 1.** Distribution of solution targets based on their weighting vectors.

As weighting or utility function Chebychev scalarization is used: Given a solution  $s$ , an lower bound point  $s^{lb}$  (where  $s_i^{lb} = \min(f_i(X'))$ , being  $X'$  the explored search space so far), and a weight vector  $(w_1^1, \dots, w_k^1)$  associated to a problem with  $k$  objective functions  $(f_1, \dots, f_k)$ , then the Chebychev utility function reads  $u_{Cheb} = \max_{i=1}^k w_i(f_i(s) - s_i^{lb})$ .

The mating selection in a Region Sub-population selects one individual based on binary tournament selection. This selection method picks two individuals randomly and chooses the one with the best fitness, where the fitness of a solution is its fitness-value regarding the sub-problem which it is associated.

The environmental selection in a Region Sub-population ( $R^i$ ) for a new solution ( $s^{new}$ ) is accomplished by evaluating  $s^{new}$  in each one of the sub-problems from this region. This process goes until all sub-problems ( $r^j \in R^i$ ) have been visited, or  $s^{new}$  is accepted by some sub-problem in  $R^i$ .

For each Region Sub-population (say  $i$ ) a list of all other regions (say  $j, j \neq i$ ) is created and the list is sorted by the distance to the centroid of  $R^i$ . This way we establish nearest neighboring region, second neighboring regions of different radius. This collection of lists will be treated as a matrix  $L \in \mathbb{B}^{\mu \times \mu}$  where the  $L(i, j)$  equals to the ranking distance between  $i$  and  $J$ . This structure is used to construct matrices  $D$  and  $M$ .

### 2.3 Sharing Information by Migration

Locality is a fundamental resource when using guided search methods, assuming that the structure of the fitness landscape leads search algorithms to high quality solutions [19]. Where low locality degenerate the performance of the search algorithm in a random search [21]. This principle provides two results: small changes in solutions cause small changes in fitness values; solutions with high fitness values are spatially localized.

The creation of a new offspring consists of the subsequent application of the recombination operator and the mutation operator:

- Recombination operators perform search exchanging information among solutions. In this operator, the information content of multiple individuals (normally two) are combined in order to generate a new individual with mutual information from its parents. Recombination operators generate offspring, where the distances between offspring and parents are usually equal to or smaller than the distance between parents.
- A mutation operator generates a solution  $s^{new}$  from  $s^{old}$  by a small random change in  $s^{old}$ . Mutation operators do not use the neighborhood lists.

Together, a search step combining recombination and subsequent mutation produces an offspring in a neighbourhood, which encloses the parents. A search step is useful if it generates a solution in an area of interest regarding a sub-population. Once two sub-populations can overlap a common area of interest, sharing offspring solutions can be beneficial. The process of sharing offspring between sub-populations is known as migration.

This migration operator demands a topology defining links between source and target sub-populations. Sprave [26] presents a formal model of population structures in evolutionary algorithms based on hypergraphs. This model allows using an individual hypergraph matrix as migration topology or mating selection topology. Here, we use this idea defining the migration topology by means of destination matrix ( $D$ ) where  $d_{ij} = 1$  if  $i$  is a source sub-population and  $j$  is a target sub-population, and  $d_{ij} = 0$  otherwise. In this work only newly created individuals are submitted for migration.

### 2.4 Sharing Information by Recombination

Next to migration, MOEA/S also allows sharing information by mating parents from different sub-populations. An essential step in recombination methods is selecting the right combination of mates in order to generate useful descendants. This step is called mating selection. The problem in mating selection can be stated as: given a first parent solution  $s^{p1}$  from  $P_{p1}$ , which other sub-population  $P_{p2}$  should be selected in order to find a good matching ( $s^{p2}$ ) for the first parent?

Multi-objective problems deal with highly conflicting objectives, and hence the search in each sub-population leads to different specialized region. Therefore, combining solutions from sub-populations located on extremely different parts of the Pareto front becomes unreasonable as they evolve. In contrast, solutions in

similar sub-problems have similar information content; combining these solutions leads to exploiting small regions, degrading the search in the first generations.

### 3 Experiments

The multi-objective 0-1 knapsack problem based on [31] is defined for 2 and 3 objectives with 500 items, we call it MOKP-2 or MOKP-3 according to the number of objectives. The second problem is a multi-objective formulation of traveling salesperson problem (MOTSP) [6] defined for 2 and 3 objectives with 30 cities, we call it MOTSP-2 or MOTSP-3 according to the number of objectives. These problems were chosen because in these problems the similarity of solutions reflects to some extent the similarity of the obtained results. As opposed to many problems in continuous multi-objective optimization, such as ZDT [30] and DTLZ [8], spread and convergence are both influenced by all variables. In other words, there is no separation of variables that influence only spread or only convergence. Moreover, the problems have practical relevance and are structurally similar to real world problems.

The hypervolume indicator and R2-indicator were used to assess the performance of the population. The hypervolume indicator has been the most used quality indicator in the performance assessment of Pareto front approximations [18,20]. The hypervolume indicator measures the size of the region dominated by an approximation set [31], and bound from above by a reference point. R2-indicator is defined as an integral over a weight space for a family of distance to a reference point utility function (typically weighted Chebychev distance to the ideal point). Thus, R2-indicator is very suitable for decomposition problems (which uses also utility functions).

In this paper we also propose two metrics: number of useful migrations ( $\alpha$ ) and number of useful mating ( $\beta$ ).  $\alpha$  results from: given distance rank<sup>2</sup>, denoted with  $(\rho \in 1, \dots, \mu)$ ,  $\alpha_\rho$  counts how many useful migrations occurs between sub-populations in distance  $\rho$ ; thus, for instance,  $\alpha_1$  counts how many individuals from  $R^i$  are accepted by  $R^j$ , with  $R^j$  being the nearest neighboring sub-population from  $R^i$ . Second metric,  $\beta$  states: given a distance rank  $\rho (\in 1, \dots, \mu)$ ,  $\beta_\rho$  counts useful offspring resulted from mating between sub-populations in this distance; thus,  $\beta_1$  counts how many individuals are accepted (in any sub-population) from matches between  $R^i$  and  $R^j$ , once  $R^j$  is the nearest neighboring sub-population from  $R^i$ ;  $\beta_0$  counts how many individuals are accepted (in any sub-population) from a mating of parents from the same sub-population.

#### 3.1 Experiment Settings

The MOEA/S setting in the research study on the benefits of sharing information (Experiments 1 and 2) is given by:  $\mu$  - *number of sub-populations* equals

---

<sup>2</sup> Note, as a detail, that in the case of ties, that is two sub-populations share the same distance, the distance rank will be randomly assigned.

to 6;  $N$  - number of individuals set as 36; set of problems defined as MOKP-2, MOKP-3, MOTSP-2, MOTSP-3; and number of generations is  $(120000/N)-2$ .

For MOKP problems, we used binary representation, one-point crossover as in [31], and 2/500 bit-flip mutation rate. For MOTSP problems, we used permutation representation with order crossover and swap mutation. Crossover and mutation rates of 1. The presented results are average performance metrics obtained by the populations at a given time; this average considers 20 runs of the algorithm (generation vs. quality measure). Thus, we can study the sharing process between the sub-populations according to the time (generation number).

**Experiment 1: Sharing Information by Migration Between Independent Sub-populations.** First experiment explores the relationship between neighborhood of a sub-population and the effectiveness of sharing its descendants by **migration**. Destination Matrix:  $D(i, j) = 1$ , for all  $i$  and  $j$ . The mating matrix now reads  $M(i, j) = 1$  if  $i = j$ ; 0, otherwise. We compare the different  $\alpha$  over the generations. That is, we assess the success that is attributed to migration of different radius. For statistical smoothing purposes, we report cumulative values of  $\alpha$  over ranges of distance ranks.

**Experiment 2: Sharing Information by Recombination.** In the second experiment there is no restriction on mating or migration selection process. The destination matrix is set to  $D(i, j) = 1$ , for all  $i$  and  $j$ . Thus, the mating matrix reads  $M(i, j) = 1$ , for all  $i$  and  $j$ . The other parameters are set as in Sect. 3.1. This experiment aims at understanding how the distance between sub-populations of parents is related with producing successful offspring. We used a scheme selection that guarantees all  $\beta$  range are assessed. For statistical smoothing purposes, we report cumulative values of  $\beta$  over ranges of distance ranks.

**Experiment 3: Using Local vs. Global Sharing in MOEA/S.** Last experiment compares three MOEA/S designs ( $MOEAS_{can}$ ,  $MOEAS_0$  and  $MOEAS_1$ ) with MOEA/D [29] and NSGA-III [7] implementations found in PlatEMO [27]. Here we compare approaches with global and local sharing policies. All algorithms in this experiment use the same maximum size of population and search operators. Here:  $N = 120$  is the (maximum) population size. As specific parameters MOEA/D uses neighborhood size  $T = N/10$ ; NSGA-III uses  $N$  accumulation points; and MOEA/S implementations work with  $\mu = 10$  sub-populations.

The MOEA/S implementations are detailed as follows:

- **Global Sharing**  $MOEAS_{can}$  ('can' stands for canonical) defines no restriction over mating parents from different sub-populations. A new solution can migrate to all sub-populations.  $M$  and  $D$  are set as Experiment 2.
- **No Sharing**  $MOEAS_0$  each sub-population works independently and there is no sharing, i.e. the sub-populations work in parallel without migration;  $M$  is as in Experiment 1 (no mating across sub-populations) and  $D = M$ .

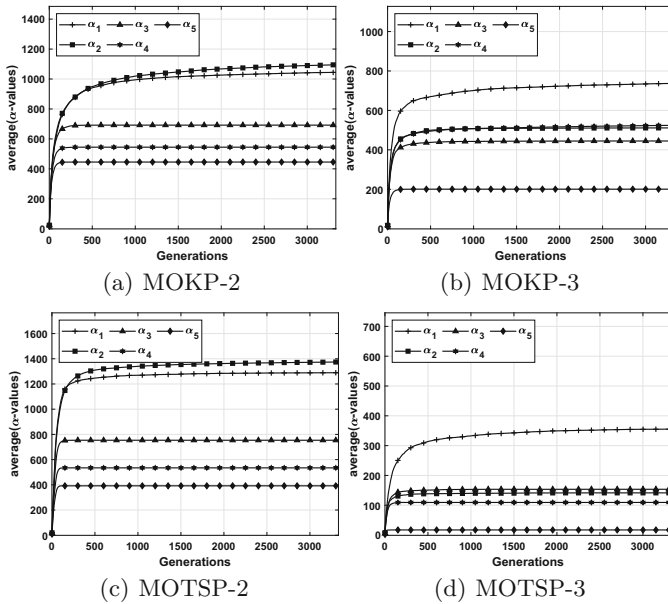
– **Local Sharing** In  $MOEA_S_1$  the distance between two sources of parents is restricted to 1 and solutions can migrate only for the three closest neighborhoods; thus,  $M(i, j) = 1$  if  $L(i, j) \leq 1$ , 0 otherwise; and  $D(i, j) = 1$  if  $L(i, j) \leq 3$ , 0 otherwise. Here  $L$  is the sorting matrix defined in Sect. 2.2.

The reference points for Hypervolume-indicator was set as (26098, 28367) and (27576, 27483, 27367) for MOKP-2 and MOKP-3 test problem, respectively; for MOTSP-2 and MOTSP-3 problems it was set as (296.88, 295.32) and (288.8, 288.54, 284.15), respectively. R2 was implemented using Chebychev scalarization based utility function with the  $N$  (maximum size of the population) number of points. As Hypervolume-based and R2 indicators have obtained similar results (same ranking position when comparing the algorithms), therefore, we just show Hypervolume-indicator.

## 4 Results and Discussion

### 4.1 Sharing Information by Migration Between Sub-populations

Figure 2 illustrates the success rate of sharing information, by migration operator, between sub-populations based on their distance.



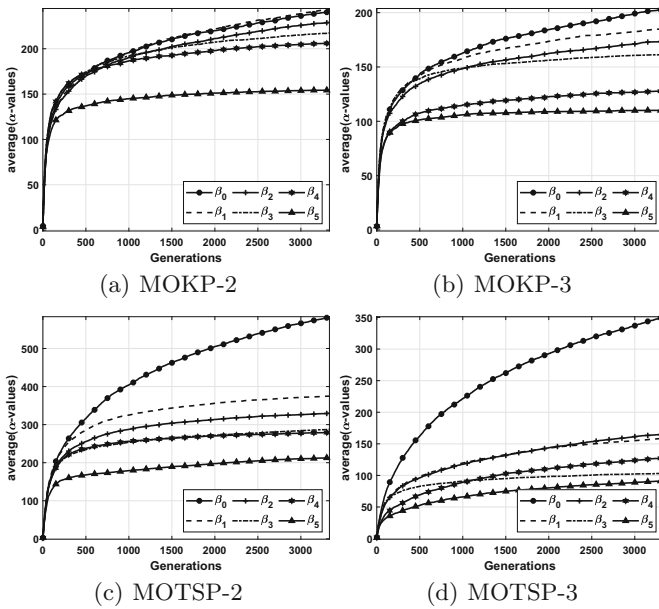
**Fig. 2.** Migration rate success grouped by distance between source and destination. Average  $\alpha$  is the cumulative counting of accepted offsprings in a given distance.

Migrating information with the first neighbour is the best option for 3-objective problems, and it is the second best option for 2-objective problems

(the best option is in the 40% closest neighboring sub-populations). In all cases, sharing information with the furthest sub-population is unlikely to be successful, in particular in later stages of the search when it becomes specialized. Although the success rate is low, migration does not affect the generation, i.e., a bad migration try is not a waste in execution count. Thus, without taking care the effort of validating a solution, sharing with all sub-populations is the best option.

### 4.2 Sharing Information by Recombination Between Sub-populations

The second study (Fig. 3) analyzes the behavior of the population quality during the evolutionary process when applying both of the operators, recombination and migration. This results reinforce the idea from Ishibuchi and Shibata [13–15] about using similarity indicators in mating selection. Crossing individuals between sub-populations can be as useful as crossing neighboring individuals. However, as the search progresses, the probability of generating good offspring by crossbreeding sub-populations decreases. Crossbreeding with the nearest neighbouring sub-populations remains successful also in the later stage of search.



**Fig. 3.** Mating rate effectiveness grouped by distance between source sub-population of parent 1 and source of parent 2. Average  $\beta$  is a cumulative value during the search.

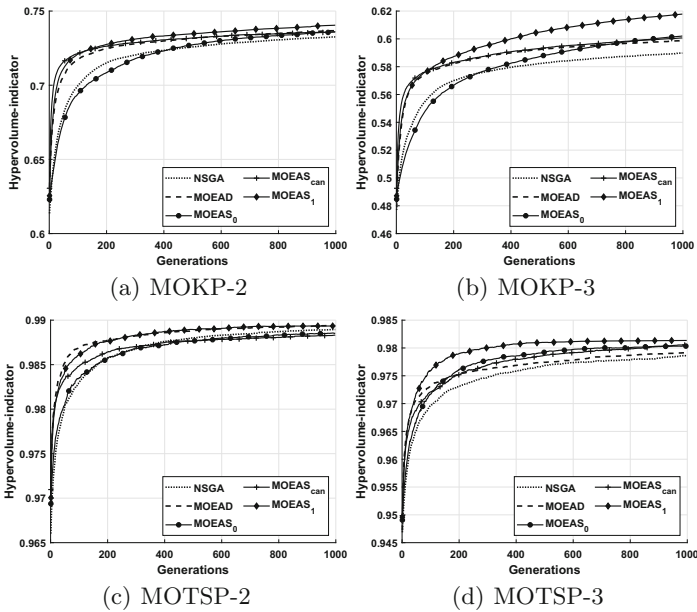
As result from Fig. 3, the highest probability of generating useful offspring is obtained by crossing individuals from the same region. This result is shown



in both problems. The only exception is Fig. 3(a) where the recombination with the first neighboring region has similar (to better) behavior. As the search progresses the probability of finding useful individuals from apart sub-populations decreases. The best mating selection (between sub-populations) scenario occurs in the first neighborhood. Only  $\beta_1$  and  $\beta_2$  continue increasing over time in all test cases.

### 4.3 Using Local vs. Global Sharing in MOEA/S

From previous results (Sects. 4.1 and 4.2), most of information needed for improving search performance in a sub-population comes from the nearest sub-populations. Thus, the last experiment studies the behavior of the evolutionary process when defining mating and migration rules, by comparing global sharing ( $MOEAS_{can}$ ), no sharing or independent sub-populations ( $MOEAS_0$ ), and local sharing ( $MOEAS_1$ ) versions of MOEA/S. We also compare its behavior with MOEA/D and NSGA III implementations. Figure 4 presents the search behavior of these algorithms regarding Hypervolume indicator.



**Fig. 4.** Performance of MOEA/S using mating and migration restriction compared with other MOEAs implementations including NSGA III and MOEA/D.

As result, from Fig. 4, sharing information has been demonstrated as the right choice for improving the convergence rate on MOEA/S. Sharing information with all sub-populations is one of the best options in the beginning of the

search, where  $MOEAS_{can}$  is the best option for MOKP test problems (Fig. 4(a) and 4(b)) until ca. generation 100. However, as the search progresses  $MOEAS_0$  and  $MOEAS_1$  continue improving search performance, while MOEA/D and  $MOEAS_{can}$  prematurely converge. Only in Fig. 4(c) MOEA/D has a similar performance when compared to  $MOEAS_1$  even after generation 500 (no significant difference by Wilcoxon rank sum test,  $p = 0.05$ ).

Once MOTSP-3 takes advantage on neighboring recombination (as shown in experiments Sects. 4.1 and 4.2), Fig. 4(d),  $MOEAS_0$  shows its best ranking performance. NSGA-III performs better with two-objective problems as compared to three-objective test problems. *Thus, sharing information can be considered beneficial for mating and migration selection.*

## 5 Conclusion and Outlook

Our study has investigated sharing in multi-objective optimization across sub-populations that explore different regions of the Pareto front. Both, sharing by migration and by mating has found to be useful tool for improvement of combinatorial multi-objective optimization. Diversity is achieved by exchanging information between dissimilar sub-populations, which influences the performance of the firsts generations. On the other hand, focusing on similar sub-populations can improve exploitation in the search. Consequently, mating neighboring parents leads to better final results. Another important finding is, that the radius of sharing and the type of sharing has a crucial influence on its beneficial effect. Moreover, long radii have found to more benefit early stages of search, whereas in later stages short, but non-zero, radii for sharing are more beneficial.

The study points out and confirms some interesting phenomena regarding sharing and paves the way to future work taking these novel findings into account: Adaptive selection schemes could be considered once the effectiveness of distance-based migration, and mating selection depends on the stage of the search. In the final stages, there is no need to migrate solutions or mating solutions between sub-populations. The selection scheme proposed by Ishibuchi [15] can be the right choice if we consider the panmictic population. However, this selection scheme is not extensible for parallel populations.

Since our study suggests that mating selection is highly related with neighborhood of solutions, future work on designing MOEA/S should take neighborhood adaptation measures for mating into account. Moreover, there is room for discussion on neighborhood/deme representations using hypergraphs (see Sprave [26]). In particular, such considerations might be of relevance for theoretical analysis using Markov chain techniques.

**Acknowledgments.** This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. We thank LIACS for hosting and promoting the collaboration, which resulted in this paper. Particularly, we thank NACO and MODA groups at LIACS and Laboratory of Modern Heuristics at INF/UFG by the discussions and background.

## References


1. Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. In: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (2007)
2. Brasil, C.R.S., Delbem, A.C.B., da Silva, F.L.B.: Multiobjective evolutionary algorithm with many tables for purely ab initio protein structure prediction. *J. Comput. Chem.* **34**(20), 1719–1734 (2013)
3. Camillo, M.H.M., et al.: Combining exhaustive search and multi-objective evolutionary algorithm for service restoration in large-scale distribution systems. *Electric Power Syst. Res.* **134**, 1–8 (2016)
4. Camillo, M.H.M., et al.: Validation of a methodology for service restoration on a real Brazilian distribution system. In: 2014 IEEE PES Transmission & Distribution Conference and Exposition-Latin America (PES T&D-LA), pp. 1–6. IEEE (2014)
5. Camillo, M.H.M., et al.: A multi-objective evolutionary algorithm with efficient data structure and heuristic initialization for fault service restoration. *Procedia Comput. Sci.* **80**, 2367–2371 (2016)
6. Corne, D.W., Knowles, J.D.: Techniques for highly multiobjective optimisation: some nondominated points are better than others. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007, pp. 773–780. Association for Computing Machinery, New York (2007)
7. 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)
8. 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)
9. Delbem, A.C.B., de Carvalho, A.C.P.D.L.F., Bretas, N.G.: Main chain representation for evolutionary algorithms applied to distribution system reconfiguration. *IEEE Trans. Power Syst.* **20**(1), 425–436 (2005)
10. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: formulation: discussion and generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms, San Francisco, CA, USA, pp. 416–423. Morgan Kaufmann Publishers Inc. (1993)
11. Gois, M.M., Sanches, D.S., Martins, J., Junior, J.B.A.L., Delbem, A.C.B.: Multi-objective evolutionary algorithm with node-depth encoding and strength pareto for service restoration in large-scale distribution systems. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) *EMO 2013. LNCS*, vol. 7811, pp. 771–786. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-37140-0\\_57](https://doi.org/10.1007/978-3-642-37140-0_57)
12. Ishibuchi, H., Akedo, N., Nojima, Y.: Relation between neighborhood size and MOEA/D performance on many-objective problems. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) *EMO 2013. LNCS*, vol. 7811, pp. 459–474. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-37140-0\\_35](https://doi.org/10.1007/978-3-642-37140-0_35)
13. Ishibuchi, H., Shibata, Y.: An empirical study on the effect of mating restriction on the search ability of EMO algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L., Deb, K. (eds.) *EMO 2003. LNCS*, vol. 2632, pp. 433–447. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36970-8\\_31](https://doi.org/10.1007/3-540-36970-8_31)

14. Ishibuchi, H., Shibata, Y.: A similarity-based mating scheme for evolutionary multiobjective optimization. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 1065–1076. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-45105-6\\_116](https://doi.org/10.1007/3-540-45105-6_116)
15. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Choosing extreme parents for diversity improvement in evolutionary multiobjective optimization algorithms. In: 2007 IEEE International Conference on Systems, Man and Cybernetics, ISIC, pp. 1946–1951. IEEE (2007)
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. Murata, T., Ishibuchi, H., Gen, M.: Specification of genetic search directions in cellular multi-objective genetic algorithms. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 82–95. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44719-9\\_6](https://doi.org/10.1007/3-540-44719-9_6)
18. Okabe, T., Jin, Y., Sendhoff, B.: A critical survey of performance indices for multi-objective optimisation. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 2, pp. 878–885. IEEE (2003)
19. Raidl, G.R., Gottlieb, J.R.: Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: a case study for the multidimensional knapsack problem. *Evol. Comput.* **13**(4), 441–475 (2005)
20. Riquelme, N., Von Lüken, C., Baran, B.: Performance metrics in multi-objective optimization. In: 2015 Latin American Computing Conference (CLEI), pp. 1–11. IEEE (2015)
21. Rothlauf, F.: Design of Modern Heuristics: Principles and Application. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-3-540-72962-4>
22. Sanches, D.S., Mazucato, S.C., Castoldi, M.F., Delbem, A., London Jr., J.B.: Combining subpopulation tables, non-dominated solutions and strength pareto of MOEAs to treat service restoration problem in large-scale distribution systems. In: IECON 2013–39th Annual Conference of the IEEE Industrial Electronics Society, pp. 1986–1991, November 2013
23. Sanches, D.S., et al.: Multiobjective evolutionary algorithm with a discrete differential mutation operator developed for service restoration in distribution systems. *Int. J. Electr. Power Energy Syst.* **62**, 700–711 (2014)
24. Sanches, D.S., Mansour, M., London Jr., J.B., Delbem, A., Santos, A.C.: Integrating relevant aspects of MOEAs to solve loss reduction problem in large-scale distribution systems. In: 2011 IEEE PES Trondheim PowerTech: The Power of Technology for a Sustainable Society, POWERTECH 2011, June 2011
25. Santos, A., Delbem, A., London, J.B., Bretas, N.: Node-depth encoding and multi-objective evolutionary algorithm applied to large-scale distribution system reconfiguration. *IEEE Trans. Power Syst.* **25**(3), 1254–1265 (2010)
26. Sprave, J.: A unified model of non-panmictic population structures in evolutionary algorithms. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), vol. 2, pp. 1384–1391 (1999)
27. 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)
28. Wang, Z., Zhang, Q., Zhou, A., Gong, M., Jiao, L.: Adaptive replacement strategies for MOEA/D. *IEEE Trans. Cybern.* **46**(2), 474–486 (2016)

29. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
30. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. *Evol. Comput.* **8**(2), 173–195 (2000)
31. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans. Evol. Comput.* **3**(4), 257–271 (1999)



# Multi-objective Optimization by Uncrowded Hypervolume Gradient Ascent

Timo M. Deist<sup>1</sup>(✉) , Stefanus C. Maree<sup>1</sup>, Tanja Alderliesten<sup>2</sup>,  
and Peter A. N. Bosman<sup>1</sup>

<sup>1</sup> Centrum Wiskunde and Informatica, Life Sciences and Health Research Group,  
Amsterdam, The Netherlands

{timo.deist,maree,peter.bosman}@cwi.nl

<sup>2</sup> Department of Radiation Oncology, Leiden University Medical Center, Leiden,  
The Netherlands

t.alderliesten@lumc.nl

**Abstract.** Evolutionary algorithms (EAs) are the preferred method for solving black-box multi-objective optimization problems, but when gradients of the objective functions are available, it is not straightforward to exploit these efficiently. By contrast, gradient-based optimization is well-established for single-objective optimization. A single-objective reformulation of the multi-objective problem could therefore offer a solution. Of particular interest to this end is the recently introduced uncrowded hypervolume (UHV) indicator, which is Pareto compliant and also takes into account dominated solutions. In this work, we show that the gradient of the UHV can often be computed, which allows for a direct application of gradient ascent algorithms. We compare this new approach with two EAs for UHV optimization as well as with one gradient-based algorithm for optimizing the well-established hypervolume. On several bi-objective benchmarks, we find that gradient-based algorithms outperform the tested EAs by obtaining a better hypervolume with fewer evaluations whenever exact gradients of the multiple objective functions are available and in case of small evaluation budgets. For larger budgets, however, EAs perform similarly or better. We further find that, when finite differences are used to approximate the gradients of the multiple objectives, our new gradient-based algorithm is still competitive with EAs in most considered benchmarks. Implementations are available at <https://github.com/scmaree/uncrowded-hypervolume>.

**Keywords:** Multi-objective optimization · Uncrowded hypervolume · Gradient search

## 1 Introduction

Evolutionary algorithms (EAs) are the preferred method for solving black-box multi-objective (MO) optimization problems, when assuming the underlying

---

T. M. Deist and S. C. Maree—These authors contributed equally.

© Springer Nature Switzerland AG 2020

T. Bäck et al. (Eds.): PPSN 2020, LNCS 12270, pp. 186–200, 2020.

[https://doi.org/10.1007/978-3-030-58115-2\\_13](https://doi.org/10.1007/978-3-030-58115-2_13)

details of the problem are unknown [5]. However, when gradient information of the objective functions is available, it is not straightforward to exploit this information efficiently in the optimization process. This can be mainly attributed to the two-sided goal of multi-objective optimization, which is to obtain a set of solutions, known as an approximation set, on the one hand containing solutions that are (near) Pareto optimal, and on the other hand representing a diverse set of trade-offs between the objectives [3].

When considering a to-be-minimized bi-objective function  $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^2$ , the Karush-Kuhn-Tucker (KKT) [17, 20] conditions can be used to identify a descent direction  $d(\mathbf{x})$  for a solution  $\mathbf{x} \in \mathbb{R}^n$  for which all objectives are non-worsening, by taking a weighted convex combination of the gradients of the individual objectives  $\nabla f_0$  and  $\nabla f_1$ ,

$$d(\mathbf{x}) = w_0 \cdot \nabla f_0(\mathbf{x}) + w_1 \cdot \nabla f_1(\mathbf{x}), \quad (1)$$

with  $w_0, w_1 \geq 0$ . In general, there exist infinitely many search directions  $d(\mathbf{x})$  for which all objectives are non-worsening, and different methods have been developed in which a single descent direction is computed [6, 10, 21]. While this provides an approach to converge to Pareto optimal solutions, it does not tell us directly how to take solution diversity into account, which has shown to be non-trivial [4, 23]. We therefore consider a different avenue to handle gradients for MO optimization in this work, which is to cast the MO problem as a single-objective (SO) optimization problem, in which a quality indicator is used to quantify the quality of an approximation set [7, 18]. One popular quality indicator is the hypervolume indicator [30], which measures the volume in objective space that is dominated by an approximation set. The hypervolume indicator is currently the only known indicator that is Pareto-compliant, meaning that solutions in a set with maximal hypervolume are Pareto optimal [9], and it furthermore takes diversity intrinsically into account [1]. Additionally, the hypervolume indicator is differentiable with respect to a problem's objective functions in strictly non-dominated points which allows determining gradient weights via the chain rule [8].

A limitation of the hypervolume indicator however is that it ignores dominated solutions. This prevents the use of the hypervolume indicator directly in indicator-based MO optimization, as it cannot be used to steer dominated solutions to a non-dominated area in the search space [24]. SMS-EMOA [2] overcomes this limitation by using non-dominated sorting to create subsets of solutions such that solutions within a subset are non-dominated. Consequently, each solution's hypervolume contribution with respect to its subset can be computed and used to steer the solution towards the Pareto front. The hypervolume indicator gradient ascent multi-objective optimization (HIGA-MO) algorithm [26] computes hypervolume gradients for solutions in subsets created by non-dominated sorting and thus achieves gradient-based steering for dominated solutions. An approach to incorporate dominated solutions into a hypervolume-based indicator is the uncrowded hypervolume improvement [24] which was combined with the newly presented Sofomore framework to perform optimization by interleaving

single-objective optimizers. In [19], this quality indicator for single solutions was recently converted into a quality measure for solution sets, called the uncrowded hypervolume (UHV), which is directly suitable for indicator-based MO optimization. The resulting UHV problem was then efficiently solved with the gene-pool optimal mixing evolutionary algorithm by exploiting UHV-specific properties (UHV-GOMEA).

In this work, we formulate gradient expressions for the UHV, such that it can be used directly in SO gradient ascent schemes. (Note that the UHV needs to be maximized, independent of whether the underlying MO problem is a minimization or maximization problem.) To demonstrate this, we solve it with the same scheme as used by HIGA-MO, and with *Adam*, a preminent method for efficient stochastic optimization [16]. We further compare UHV gradient ascent to HIGA-MO, and the EAs UHV-GOMEA and Sofomore-GOMEA [19]. For the experimental comparison, we employ simple quadratic benchmark functions similar to benchmarks used in [19] and also the Walking Fish Group (WFG) benchmark set [14]. Additionally, for a fair comparison to EAs, we study the performance of the gradient-based methods in a black-box setting, where gradient information of the MO problem is not available, by using a finite difference gradient approximation. The remainder of this paper is organized as follows. In Sect. 2, we introduce preliminaries of the (uncrowded) hypervolume indicator. In Sect. 3, we introduce our UHV gradient ascent algorithm. Experimental comparisons are described in Sect. 4, followed by a discussion in Sect. 5.

## 2 Uncrowded Hypervolume Optimization

We consider MO problems given by a to-be-minimized  $m$ -dimensional objective function  $\mathbf{f} : \mathcal{X} \rightarrow \mathbb{R}^m$ , where  $\mathcal{X} \subseteq \mathbb{R}^n$  is the  $n$ -dimensional (box) constrained decision space. We focus on the bi-objective case  $m = 2$  in this work. Let  $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$  be a solution of the MO problem, which we from now on refer to as an *MO-solution*. The goal of MO optimization is to obtain a set of (near-)Pareto-optimal MO-solutions  $\mathcal{S} \subset \mathcal{X}$  of manageable size. To evaluate the quality of  $\mathcal{S}$ , we use the uncrowded hypervolume (UHV) indicator [19], which measures the area in objective space enclosed by the non-dominated MO-solutions in  $\mathcal{S}$  and a reference point  $r = (r_0, r_1)$  (as the hypervolume indicator [30]), and uses the uncrowded distance [24] (explained below) to steer dominated MO-solutions. As  $\mathcal{S}$  can contain dominated MO-solutions, let  $\mathcal{A}$  be the approximation set of  $\mathcal{S}$ , i.e., the largest subset of  $\mathcal{S}$  that contains only non-dominated MO-solutions.

In order to search the space of solution sets,  $\mathcal{P}(\mathcal{X})$ , a parameterization of solution sets is required [2, 19, 26]. For this, we consider sets  $\mathcal{S}_p$  of a fixed size of  $p$  MO-solutions, and simply concatenate the decision variables of all MO-solutions into a single vector  $X \in \mathbb{R}^{np}$ , i.e.,  $X = [\mathbf{x}_0 \cdots \mathbf{x}_{p-1}]$ , similar to notation used in [8]. Additionally, let  $Y \in \mathbb{R}^{p \times m}$  be the matrix of concatenated objectives values corresponding to  $X$ , i.e.,  $Y_{i,0:m-1} = \mathbf{y}_i = \mathbf{f}(\mathbf{x}_i)$ . Finally, let  $F : \mathbb{R}^{np} \rightarrow \mathbb{R}^{m \times p}$  be the operator that evaluates the entire solution set given by  $X$ , i.e.,  $Y = F(X)$ . This implies that an evaluation of  $F$  consists of  $p$  evaluations of the MO problem



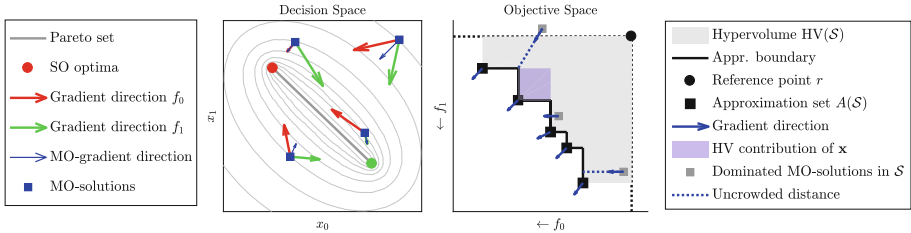
(MO-evaluations). The resulting SO UHV-based optimization problem can then be formulated as,

$$\begin{aligned} & \text{maximize} && g(X) = \text{UHV}(F(X)) = \text{HV}(F(X)) - \text{UD}(F(X)), \\ & \text{with} && \mathbf{f} : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad F : \mathbb{R}^{np} \rightarrow \mathbb{R}^{m \times p}, \quad X \in \mathbb{R}^{np}, \end{aligned} \quad (2)$$

where  $\text{HV} : \mathbb{R}^{m \times p} \rightarrow \mathbb{R}_{\geq 0}$  is the hypervolume indicator [30] and  $\text{UD} : \mathbb{R}^{m \times p} \rightarrow \mathbb{R}_{\geq 0}$  is the mean of the uncrowded distances  $\text{ud}(\mathbf{y}, Y)$  [24], which measure the shortest distance of a point  $\mathbf{y}$  towards the domination boundary of  $Y$  in objective space. It is called the uncrowded distance as the nearest point on the boundary of  $Y$  is generally away from points in  $Y$ . The UD is then given by,

$$\text{UD}(Y) = \frac{1}{p} \sum_{i=0}^{p-1} \text{ud}(\mathbf{y}_i, Y)^m. \quad (3)$$

We refrain from repeating a mathematical definition here, but provide an illustration in Fig. 1. Note that, in contrast to [19], we only consider the interior boundary of  $Y$  here, which was found to improve performance in preliminary experiments, as the extreme points of  $Y$  are often already well-positioned close to the extremes of the approximation front (i.e. the approximation set in objective space), and steering additional points into the same location causes undesired computational overhead. Finally, note that the UHV is equivalent to the hypervolume indicator when all MO-solutions in  $\mathcal{S}$  are non-dominated, which implies that the UHV is still Pareto-compliant on the space of approximation sets.



**Fig. 1.** Illustration of UHV gradient ascent on a bi-objective problem. The MO-gradient direction in decision space (left subfigure) is a weighted linear combination of the SO gradients, where the weights are determined based on the UHV gradient direction in objective space (right subfigure).

### 3 UHV Gradient Ascent

We apply a gradient ascent scheme to  $g(X) = \text{UHV}(F(X))$  in Eq. (2). For this, we use the gradient of the hypervolume indicator as was derived in [8]. We

briefly describe the concept here, but refer the reader to [8] for a rigorous mathematical derivation and analysis. The gradient  $\nabla g(X) = \nabla \text{UHV}(F(X))$  can be split up into subvectors corresponding to different MO-solutions by using that  $X = [\mathbf{x}_0 \cdots \mathbf{x}_{p-1}] \in \mathbb{R}^{np}$ ,

$$\nabla g(X) = \frac{\partial \text{UHV}(F(X))}{\partial X} = \left[ \frac{\partial \text{UHV}(F(X))}{\partial \mathbf{x}_0} \cdots \frac{\partial \text{UHV}(F(X))}{\partial \mathbf{x}_{p-1}} \right]. \quad (4)$$

We now apply the chain rule to each of the subvectors  $i$  by using  $\mathbf{y}_i = \mathbf{f}(\mathbf{x}_i)$ ,

$$\frac{\partial \text{UHV}(F(X))}{\partial \mathbf{x}_i} = \frac{\partial \text{UHV}(F(X))}{\partial F(X)} \cdot \frac{\partial F(X)}{\partial \mathbf{x}_i} = \sum_{j=0}^{p-1} \frac{\partial \text{UHV}(F(X))}{\partial \mathbf{y}_j} \cdot \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_i}, \quad (5)$$

where we can now use that  $\frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_i} = \mathbf{0}$  for  $j \neq i$ , as the fitness values of  $\mathbf{y}_j = \mathbf{f}(\mathbf{x}_j)$  do not depend on  $\mathbf{x}_i$ . For  $j = i$ , we have  $\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_i} = [\nabla f_0(\mathbf{x}_i) \ \nabla f_1(\mathbf{x}_i)]$ , which are simply the gradients of the MO problem. This gives,

$$\frac{\partial \text{UHV}(F(X))}{\partial \mathbf{x}_i} = \frac{\partial \text{UHV}(F(X))}{\partial f_0(\mathbf{x}_i)} \cdot \nabla f_0(\mathbf{x}_i) + \frac{\partial \text{UHV}(F(X))}{\partial f_1(\mathbf{x}_i)} \cdot \nabla f_1(\mathbf{x}_i). \quad (6)$$

Note the correspondence of this expression with the weighted search direction in Eq. (1). Directly using the objective space gradients to determine the search direction would cause MO-solutions that contribute more to the UHV to make big steps, and MO-solutions that contribute little to slowly *creep*, which was noted earlier [12, 26]. To overcome this, we normalize the objective gradients by setting  $W = \left\| \left[ \frac{\partial \text{UHV}}{\partial f_0(\mathbf{x}_i)} \ \frac{\partial \text{UHV}}{\partial f_1(\mathbf{x}_i)} \right] \right\|$ , which gives us the desired search direction,

$$\frac{1}{W} \frac{\partial \text{UHV}(F(X))}{\partial \mathbf{x}_i} = \frac{1}{W} \frac{\partial \text{UHV}(F(X))}{\partial f_0(\mathbf{x}_i)} \cdot \nabla f_0(\mathbf{x}_i) + \frac{1}{W} \frac{\partial \text{UHV}(F(X))}{\partial f_1(\mathbf{x}_i)} \cdot \nabla f_1(\mathbf{x}_i). \quad (7)$$

It now remains to find an expression for the objective space gradients. We now use that  $\text{UHV} = \text{HV} - \text{UD}$ . For both objectives  $k = \{0, 1\}$ , this gives,

$$\frac{\partial \text{UHV}(F(X))}{\partial f_k(\mathbf{x}_i)} = \frac{\partial \text{HV}(F(X))}{\partial f_k(\mathbf{x}_i)} - \frac{\partial \text{UD}(F(X))}{\partial f_k(\mathbf{x}_i)}.$$

Whenever  $\mathbf{x}_i$  is a dominated MO-solution, it has no contribution to the hyper-volume, and the first term is therefore equal to zero. For the second term, let  $\mathbf{s}(\mathbf{f}(\mathbf{x}_i)) \in \mathbb{R}^m$  be the point towards which the uncrowded distance is computed, i.e., the nearest point to  $\mathbf{f}(\mathbf{x}_i)$  on the approximation boundary given, as illustrated in Fig. 1. Using the definition of UD in Eq. (3), we obtain the final expression for objective-space derivative for dominated MO-solutions,

$$\frac{\partial \text{UD}(F(X))}{\partial f_k(\mathbf{x}_i)} = \frac{1}{p} \frac{\partial}{\partial f_k(\mathbf{x}_i)} \|\mathbf{f}(\mathbf{x}_i) - \mathbf{s}(\mathbf{f}(\mathbf{x}_i))\|^m.$$

Whenever  $\mathbf{x}_i$  is a non-dominated MO-solution, the objective-space hyper-volume gradient can be computed by the approach described in [8]. Conceptually, the computation can be reduced to the objective-space gradient of the

**Table 1.** UHV gradient ascent schemes for maximizing  $g(X)$ .

Adam [16]	GA-MO [25]
<b>Initial values:</b> $\gamma^0 = \ \mathcal{X}_{\text{init}}\  \cdot 10^{-2}$ , $b_0 = 0.9$ , $b_1 = 0.999$ , $b_2 = 0.99$ , $\varepsilon = 10^{-16}$ , $\mathbf{m}_{-1} = \mathbf{v}_{-1} = \mathbf{0}$ .	<b>Initial values:</b> $c = 0.1$ , $\alpha = 0.7$ , $\beta = 0.7$ , and for $i = 0, \dots, (p-1)$ : $\gamma_i^{-1} = \ \mathcal{X}_{\text{init}}\  \cdot 10^{-2}$ , $\mathbf{n}_i^{-1} = \mathbf{0}$ , $m_i^{-1} = 0$ .
For $t = 0, 1, \dots$ , $\mathbf{m}^t = b_0 \mathbf{m}^{t-1} + (1 - b_0) \nabla g(X^t)$ , $\mathbf{v}^t = b_1 \mathbf{v}^{t-1} + (1 - b_1) \nabla g^2(X^t)$ , $X^{t+1} = X^t + \gamma^t \frac{\mathbf{m}^t / (1 - (b_0)^{t+1})}{\sqrt{\mathbf{v}^t / (1 - (b_1)^{t+1})} + \varepsilon}$ , $\gamma^{t+1} = \begin{cases} b_2 \gamma^t, & \text{if } g(X^{t+1}) \leq g(X^t), \\ \gamma^t, & \text{else.} \end{cases}$	For $t = 0, 1, \dots$ , $d^- = \min_{l, k \in \{0, \dots, (p-1)\}, l \neq k} \ \mathbf{x}_l^t - \mathbf{x}_k^t\ $ , $d^+ = \max_{l, k \in \{0, \dots, (p-1)\}, l \neq k} \ \mathbf{x}_l^t - \mathbf{x}_k^t\ $ , $\gamma^{\text{UB}} = \beta (d^+ + d^-) / 2$ For $i = 0, \dots, (p-1)$ , $\mathbf{n}_i^t = \nabla g(\mathbf{x}_i^t) / \ \nabla g(\mathbf{x}_i^t)\ $ , $m_i^t = (1 - c) m_i^{t-1} + c (\mathbf{n}_i^{t-1}, \mathbf{n}_i^t)$ , $\gamma_i^t = \min\{\gamma^{\text{UB}}, \gamma_i^{t-1} e^{\alpha m_i^t}\}$ , $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \gamma_i^t \mathbf{n}_i^t$ .

hypervolume contribution of that MO-solution, which is easily computed when the neighbouring MO-solutions on the approximation front are known (Fig. 1). Additionally, whenever  $\mathbf{x}_i$  is a non-dominated MO-solution, it determines the approximation boundary, which is used in the computation of the UD for other MO-solutions. Therefore,  $\frac{\partial \text{UD}(F(X))}{\partial f_k(\mathbf{x}_i)}$  is potentially non-zero. In that case, the UD can be improved at the cost of worsening non-dominated MO-solutions, as this reduces the uncrowded distance of dominated MO-solutions. This is undesirable, and we therefore explicitly set  $\frac{\partial \text{UD}(F(X))}{\partial f_k(\mathbf{x}_i)} = 0$  for non-dominated  $\mathbf{x}_i$ , although preliminary experiments showed that performance is largely unaffected by this. Finally, we consider the case in which  $\mathbf{x}_i$  is *weakly dominated*, which occurs for pairs of MO-solutions with at least one coinciding objective value. In this case, the objective-space gradient of the UHV is undefined [8, Proposition 3]. To prevent such case, we consider these points to be strongly dominated, and (temporarily) worsen the objective value(s) that are shared with other MO-solutions by a small value  $\varepsilon$ , which allows us to compute the uncrowded distance as before. Since objective space gradients are normalized, the actual choice of  $\varepsilon$  is irrelevant as long as it is small enough so that the weakly dominated MO-solution does not get dominated by other MO-solutions.

### 3.1 Gradient Ascent Schemes

We use two gradient ascent schemes for UHV gradient ascent, as listed in Table 1. The first scheme we consider is *Adam* [16] (UHV-Adam), which is a popular method for stochastic gradient descent. Adam uses a variance-corrected weighted average of current and previous gradients. In contrast to the original formulation, we set  $\varepsilon$  to machine precision, and we add a very simple step size shrinking scheme in which the step size is reduced if no improvement was found. The second scheme is the GA-MO scheme (UHV-GA-MO) used in the

Python implementation of HIGA-MO [25]. GA-MO updates the step size for each MO-solution separately using a weighted average of search directions’ inner products as input for an exponential cooling scheme. We adapted the weight used for averaging inner products  $c = 0.1$  (from  $c = 0.2$ ) as both HIGA-MO and UHV-GA-MO showed stagnation in preliminary experiments with  $c = 0.2$ . Additionally, we changed the upper bound on the step size  $\gamma^{\text{UB}}$  to be also based on  $d^+$ , the maximum distance between two MO-solutions in decision space, as for the UHV objective function, two dominated MO-solutions could be steered to the same point on the front, and only basing it on the minimum distance  $d^-$  could shrink  $\gamma$  prematurely. For both schemes, we use projected gradients (i.e., boundary repair) to handle box-constrained search spaces. Initial MO-solutions are initialized uniformly random in a box  $\mathcal{X}_{\text{init}} \subseteq \mathcal{X}$ , and the initial step size is based on the maximum initialization range in any dimension, which we denote by  $\|\mathcal{X}_{\text{init}}\|$ . Implementations of UHV-Adam and UHV-GA-MO are available at <https://github.com/scmaree/uncrowded-hypervolume>.

### 3.2 Finite Difference Gradient Approximation

To assess the performance of gradient-based algorithms in a black-box scenario, where exact gradients are not known, finite forward difference gradient approximations (FD) are used. The FD step size is set to  $h = 10^{-6} \cdot \bar{\gamma}_t$ , where  $\bar{\gamma} = \sum_{i=0}^{p-1} \gamma_i^t$  for UHV-GA-MO, and  $\bar{\gamma} = \gamma^t$  for UHV-Adam. In this way,  $h$  is always smaller than the mean step size. If the FD step violates the search space’s box-constraints, backward differences are used. Estimating both objectives’ gradients in one MO-solution requires  $n$  additional MO-evaluations, the number of MO-evaluations thus increases from  $p$  to  $(1 + n) \cdot p$  per iteration. When using FD, we refer to our methods as UHV-Adam-FD and UHV-GA-MO-FD.

## 4 Experiments

Experiments are conducted on several bi-objective problems: four bi-objective problems with known gradients as defined in Table 2 and nine box-constrained problems from the WFG benchmark suite [13, 14, 28]. For each algorithm, the *best* approximation set obtained so far is recorded over the run of that algorithm, where quality is measured by the algorithm itself, i.e., based on the HV or UHV. Performance is measured by the number of MO function evaluations (MO-evaluations), where we define one MO-evaluation as the computation of  $f_0, f_1, \nabla f_0$ , and  $\nabla f_1$  at once. Note that the evaluation of  $X$ , which models a solution set  $\mathcal{S}_p$  of size  $p$ , therefore costs  $p$  MO-evaluations. All problems are run with a fixed hypervolume reference point  $r = (11, 11)$ , which is rather far away from the Pareto front, as this puts additional importance towards obtaining the end points of the front [1]. However, even with this choice of reference point, the endpoints are not always included in the approximation set with optimal hypervolume, depending on the shape of the front [1].

We compare the two UHV gradient ascent schemes, UHV-Adam and UHV-GA-MO, to the EAs UHV-GOMEA-Lt and Sofomore-GOMEA from [19]. We

furthermore consider the gradient-based HIGA-MO. UHV-GOMEA-Lt uses a linkage tree in which at most a few MO-solutions are updated simultaneously. A full description of UHV-GOMEA-Lt and Sofomore-GOMEA can be found here [19]. We used the Python implementation of HIGA-MO [25], but extended it with a dynamic reference point so that hypervolume gradients can also be computed for solution sets with  $f_0^+ > r_0$  or  $f_1^+ > r_1$  (which is not an issue for the UHV-based algorithms), where  $f_i^+$  is the worst value for the  $i^{\text{th}}$  objective in the set. The dynamic reference point is set to  $\hat{r} = (\max\{1.1f_0^+, r_0\}, \max\{1.1f_1^+, r_1\})$ . Algorithmic performance is always evaluated with respect to  $r$ .

As performance indicators, we consider the difference with the optimal hypervolume (for  $p$  MO-solutions)  $\Delta\text{HV}(\mathcal{A}_p) = \text{HV}(\mathcal{A}_p^*) - \text{HV}(\mathcal{A}_p)$ , where  $\mathcal{A}_p = A(\mathcal{S}_p)$  is the approximation set given by  $\mathcal{S}_p$ , and  $\mathcal{A}_p^*$  is the approximation set with optimal hypervolume. The second measure we consider is the generational distance (GD) [29]. The GD for problems 0 and 2 is computed analytically from their known Pareto set [19]. For problems 1 and 3, the GD is computed based on a sample of 5000 MO-solutions from a reference set. The GD is not Pareto compliant, but it is a useful tool to measure proximity to the Pareto set. Finally, we consider  $|\mathcal{A}_p|$ , i.e., the number of non-dominated MO-solutions in  $\mathcal{S}_p$ , which we use to measure how well different mechanisms for handling dominated MO-solutions perform. Unless mentioned otherwise, all experiments are repeated 10 times, and medians and inter-quartile ranges (IQR) are shown.

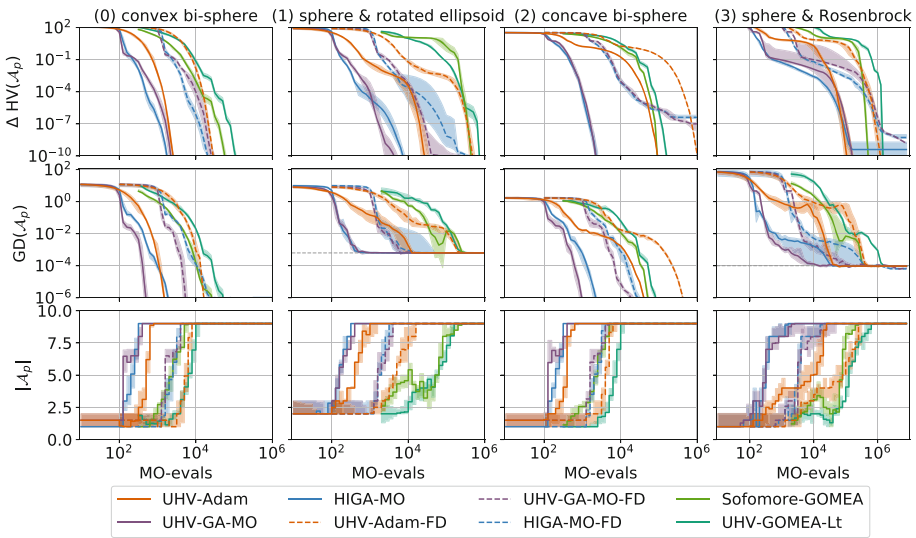
**Table 2.** Quadratic bi-objective benchmark problems.  $R$  applies a  $45^\circ$  rotation along all axes. All MO-solutions are initialized in  $[-2, 2]^{n \times p}$  for problems 0–2 and  $[0, 2]^{n \times p}$  for Problem 3.

#	Problem	$f_0$	$f_1$	Properties
0	Convex Bi-sphere	$f(\mathbf{x}) = \sum_{i=0}^{n-1} (x_i)^2$	$f(\mathbf{x} - \mathbf{c})$ , with $\mathbf{c} = [1 \ 0 \ \dots \ 0]$	Decomposable
1	Sphere & Rotated ellipsoid	$\frac{1}{n} f(\mathbf{x})$	$(\sqrt{W}R\mathbf{x} - \sqrt{W}\mathbf{c})^\top (\sqrt{W}R\mathbf{x} - \sqrt{W}\mathbf{c})$ , $W_{i,i} = 10^{\frac{-6i}{n-1}}$	Non- decomposable, Ill-conditioned
2	Concave Bi-sphere	$f(\mathbf{x})^{\frac{1}{4}}$	$f(\mathbf{x} - \mathbf{c})^{\frac{1}{4}}$	Decomposable, Concave front
3	Sphere & Rosenbrock	$f(\mathbf{x})$	$\frac{1}{(n-1)} \sum_{i=0}^{(n-2)} 100 (x_{(i+1)} - x_i^2)^2 + (1 - x_i)^2$	Bimodal, Chained dependencies

#### 4.1 Convergence in Hypervolume on the Quadratic Functions

For the first experiment, we consider the quadratic bi-objective functions from Table 2, with problem dimensionality  $n = 10$ . We consider solution sets  $\mathcal{S}_p$  of size  $p = 9$ . As we do not know  $\text{HV}(\mathcal{A}_p^*)$  analytically, the target HV is set to maximal HVs obtained from lower dimensional instances. For UHV-Adam, UHV-GA-MO, and HIGA-MO, the initial step size  $\gamma^0$  is set to one percent of the mean initialization range. As in [19], UHV-GOMEA-Lt and Sofomore-GOMEA

are run with population size  $N = 31$  for the decomposable problems 0 and 2, and  $N = 200$  otherwise. All optimizers are run for  $10^6$  MO-evaluations or until convergence criteria are met. Results are shown in Fig. 2. All gradient-based algorithms reach the target hypervolume in all problems except for HIGA-MO on Problem 3 which converges close to the target HV. HIGA-MO’s performance is more volatile across runs which is especially visible in problems 1 and 3. Both EAs, UHV-GOMEA-Lt and Sofomore-GOMEA, always obtain the target hypervolume, but require substantially more MO-evaluations. The IQRs of gradient-based and EA-based algorithms only rarely intersect, indicating that the faster convergence of gradient-based algorithms is robust to random initialization. These differences in performance are also reflected in  $GD(\mathcal{A}_p)$ . All gradient-based algorithms obtain  $|\mathcal{A}_p| = 9$  non-dominated MO-solutions faster than EAs. UHV-GA-MO and HIGA-MO reach  $|\mathcal{A}_p| = 9$  sooner than UHV-Adam which indicates that the gradient ascent scheme (i.e., GA-MO vs. Adam) has a larger effect on quickly finding non-dominated MO-solutions than the strategy for handling dominated MO-solutions (i.e., UD vs. non-dominated sorting of MO-solutions into multiple fronts in HIGA-MO).



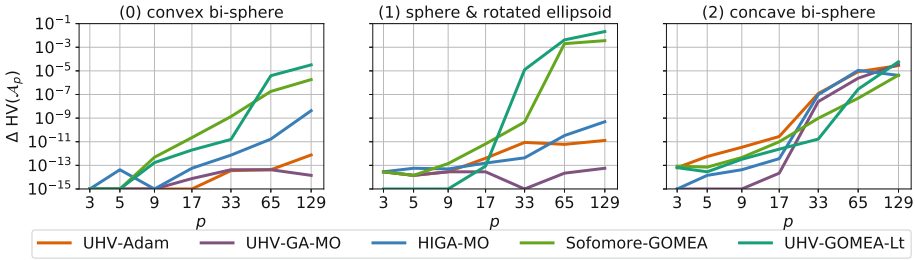
**Fig. 2.** Results for the different algorithms with  $p = 9$  on the benchmark problems in Table 2 with  $n = 10$ . Lines indicate median values and shaded areas represent the IQR. Dashed lines correspond to gradient-based algorithms with finite difference gradient approximations. MO-evals: MO-evaluations.

**Finite Difference Gradient Approximation.** In practice it may well happen that analytic gradients are not available. In Fig. 2, it can be seen that the

gradient-based algorithms lose much of their advantage over EA-based algorithms when relying on finite difference gradient approximations. Their convergence is only slightly faster (Problem 0), similar (Problem 3), or EAs now clearly outperform them (Problem 2). In Problem 2, HIGA-MO-FD converges prematurely. Only in Problem 1, there is still evidence of advantages for gradient-based algorithms: UHV-GA-MO-FD still converges more than 10 times faster than EA-based algorithms. This problem is highly dependent and ill-conditioned, and a large population is required for the EAs to solve this problem, while gradient-based algorithms directly capture these dependencies. These results also show that finite difference gradient approximations not only increase the computational cost per iteration, but could also worsen convergence rates or even cause stagnation. This is especially true for the GA-MO scheme. UHV-Adam-FD does however not show a deterioration in the rate of convergence (besides the expected shift of a factor  $1+n$ ). Adam was developed for stochastic gradient descent, and uses a weighted average of current and past gradients instead of the gradient itself, enabling it to handle the imprecise gradient approximations.

## 4.2 Effect of the Number of MO-Solutions $p$

When  $p$  is increased, the size of the to-be-optimized approximation set  $\mathcal{A}_p$  is also increased. This makes the resulting UHV optimization problem more difficult, and dependency modelling becomes essential in order to obtain the optimal distribution of MO-solutions along the front with UHV-GOMEA-Lt [19]. To investigate the dependence of convergence speed on the number of MO-solutions  $p$ , all gradient-based optimizers are applied on problems 0–2 with  $p = 2^j + 1$  for  $j = 1, \dots, 7$  and  $n = 10$ . Problem 3 is excluded as premature convergence to its local optimum would obfuscate the comparison. All optimizers are run for  $10^7$  MO-evaluations or until convergence criteria are met. The target HV is set to the maximal HV found across all algorithms. Parameter settings ( $N$  for UHV-GOMEA-Lt,  $\gamma^0$  otherwise) were tuned experimentally across all  $p = 2^j + 1$ :  $\gamma^0$  is set to  $4 \cdot 10^{-2}$  for UHV-Adam,  $4 \cdot 10^{-4}$  for UHV-GA-MO, and to  $4 \cdot 10^{-3}$  for HIGA-MO. UHV-GOMEA-Lt’s population sizes are scaled in  $p$  as the larger parameter spaces require larger populations:  $N(p) = \lceil 0.76p^{\frac{1}{3}}N_{\text{base}} \rceil$ , where  $\lceil \cdot \rceil$  is the rounding operator.  $N_{\text{base}} = 31$  for problems 0 and 2 and  $N_{\text{base}} = 200$  for Problem 1 as in Sect. 4.1. Sofomore-GOMEA interleaves optimizations of individual MO-solutions, therefore  $N$  does not need to be scaled in  $p$  and  $N$  is set to  $N_{\text{base}}$ . The median  $\Delta\text{HV}(\mathcal{A}_p)$  in problems 0–2 with varying  $p$  is shown in Fig. 3. All algorithms always reach the target HV with  $10^{-10}$  accuracy for  $p \leq 17$ . As  $p$  increases, the  $\Delta\text{HV}(\mathcal{A}_p)$  of UHV-GOMEA-Lt and Sofomore-GOMEA increases across problems. All gradient-based algorithms obtain lower  $\Delta\text{HV}(\mathcal{A}_p)$  values than both EAs as  $p$  increases with the exception of Problem 2, in which UHV-GOMEA-Lt and Sofomore-GOMEA scale better in  $p$ .



**Fig. 3.** The median distance to the target HV after  $10^7$  MO-evaluations of all algorithms for problems 0–2 with varying  $p$  and  $n = 10$  over 10 repetitions.

### 4.3 WFG Benchmark

The WFG test suite [14] consists of 9 benchmark functions with different properties. WFG1 is decomposable, but has a flat region in the decision space, which could cause stagnation. WFG2, WFG4, and WFG9 have one or more multimodal objectives, which are expected to be difficult for gradient-based algorithms. Problems WFG4–9 have concave fronts, WFG1 has a convex front, WFG2 has a disconnected convex front, and WFG3 has a linear front. We use finite difference approximations for the gradient-based algorithms. We again consider bi-objective problems, and use  $k_{\text{WFG}} = 4$  position variables and  $l_{\text{WFG}} = 20$  distance variables, resulting in a total of  $n = 24$  decision variables as originally chosen in [15]. We solve these benchmarks with approximation sets of size  $p = 9$  and a limited computational budget of  $10^5$  MO-evaluations. All experiments are repeated 30 times. Differences are tested for statistical significance (up to 4 decimals) by a Wilcoxon rank sum test with  $\alpha = 0.05$ , pairwise to the best. Ranks (in brackets) are computed based on the mean hypervolume. All statistics are computed per table. For the gradient-based algorithms, we set  $\gamma^0 = \|\mathcal{X}_{\text{init}}\| \cdot 10^{-2}$ , and a population size of  $N = 200$  was used for the population-based algorithms. Results on the WFG benchmark are shown in Table 3. UHV-Adam-FD performs best overall, while UHV-GA-MO-FD has worse performance on most problems. As expected, the gradient-based algorithms perform worse on the multi-modal problems WFG4 and WFG9. WFG2 has only one multimodal objective which does not seem to be a problem for UHV-Adam-FD. All algorithms have difficulties with the flat region in WFG1, and the worst overall hypervolume values are obtained for this problem. The only algorithm that has an explicit mechanism for handling flatness is HIGA-MO-FD, which consequently performs best for WFG1. HIGA-MO-FD re-initializes MO-solutions if the gradient is zero, which does not help traversing the plateau, but increases diversity. Note that we did not add any mechanism to handle flatness in the other algorithms. Especially with gradient-based algorithms, flatness is easily detected and a mechanism could be added to improve performance. On the WFG problems with concave fronts, UHV-Adam-FD performs well, in contrast to the previous results on the concave bi-sphere, showing that it does not have difficulties with concavity in general.



**Table 3.** Results on the WFG Benchmark. Hypervolume values are shown (mean,  $\pm$  standard deviation (rank)). Finite differences (FD) are used for the gradient-based algorithms. Bold are best scores per problem, or those not statistically different from it.

Problem	Sofomore-GOMEA	UHV-GOMEA-Lt	UHV-ADAM-FD	UHV-GA-MO-FD	HIGA-MO-FD
WFG1	86.82 $\pm$ 0.68(4)	85.50 $\pm$ 0.24(5)	96.83 $\pm$ 0.24(3)	97.12 $\pm$ 0.22(2)	<b>97.91</b> $\pm$ 0.56(1)
WFG2	109.60 $\pm$ 0.26(2)	109.38 $\pm$ 0.18(3)	<b>114.13</b> $\pm$ 3.76(1)	108.79 $\pm$ 7.79(4)	100.13 $\pm$ 3.64(5)
WFG3	115.50 $\pm$ 0.27(2)	115.48 $\pm$ 0.17(3)	<b>116.42</b> $\pm$ 0.01(1)	114.77 $\pm$ 0.34(4)	112.88 $\pm$ 0.67(5)
WFG4	<b>110.95</b> $\pm$ 0.26(1)	109.17 $\pm$ 0.48(2)	105.99 $\pm$ 1.66(5)	107.09 $\pm$ 0.74(3)	106.07 $\pm$ 1.84(4)
WFG5	108.42 $\pm$ 0.99(3)	103.45 $\pm$ 1.21(5)	<b>110.33</b> $\pm$ 0.97(1)	109.65 $\pm$ 1.29(2)	105.73 $\pm$ 1.55(4)
WFG6	113.15 $\pm$ 0.25(2)	109.64 $\pm$ 0.73(3)	<b>114.28</b> $\pm$ 0.04(1)	109.58 $\pm$ 2.00(4)	109.49 $\pm$ 2.93(5)
WFG7	112.93 $\pm$ 0.48(4)	113.06 $\pm$ 0.36(3)	<b>114.33</b> $\pm$ 0.03(1)	113.19 $\pm$ 0.59(2)	112.41 $\pm$ 0.51(5)
WFG8	109.72 $\pm$ 0.29(2)	109.27 $\pm$ 0.26(3)	<b>111.22</b> $\pm$ 0.22(1)	109.17 $\pm$ 1.29(4)	105.98 $\pm$ 2.27(5)
WFG9	<b>110.70</b> $\pm$ 1.71(1)	108.58 $\pm$ 0.57(3)	109.27 $\pm$ 0.66(2)	106.95 $\pm$ 2.02(4)	101.46 $\pm$ 2.87(5)
Rank	2.33 (2)	3.33 (4)	<b>1.78</b> (1)	3.22 (3)	4.33 (5)

## 5 Discussion

We performed gradient-based multi-objective (MO) optimization by formulating the problem as a high-dimensional single-objective optimization problem based on the uncrowded hypervolume (UHV). We presented how the gradient of the UHV can be computed from the gradients of the MO function using the chain rule. We further showed that UHV gradient optimization can be solved with existing gradient ascent schemes, obtaining results competitive to or better than EAs and another gradient-based algorithm that performs hypervolume optimization. Future studies should additionally compare the presented UHV gradient-based algorithms to popular dominance-based EAs for HV optimization and investigate scalability also in  $n$ , the dimensionality of the underlying MO problem.

We have shown that the UHV is an effective and efficient approach for obtaining a set of non-dominated MO-solutions, that requires little to no extra care during the optimization process. In [27], different techniques for steering dominated points are compared, and the uncrowded distance (UD) we use here is reminiscent of dominated point handling techniques such as secant slope weighting or gap-filling. However, a diversity loss is noted there as a possible disadvantage of gap-filling over the domination ranking technique employed in HIGA-MO, but we did not observe this in our results (Fig. 2).

UHV gradient ascent is not very sensitive to the initial step size, scales better when  $p$  is large (Fig. 3), and achieves a better hypervolume than EA-based UHV-optimization while requiring significantly fewer function evaluations (Fig. 3). When gradient information of the MO problem is missing, finite difference gradient approximation can be used, requiring  $n + 1$  MO-evaluations. Even with such approximations, UHV gradient ascent is competitive or even outperforms EAs on smaller computational budgets (Table 3), although ultimately EAs often outperform the gradient-based algorithms. The effect of approximation errors in the finite difference approximation in UHV gradient ascent is negligible when

using the Adam gradient scheme [16], as it was developed for stochastic gradient descent in which exact gradients are unavailable (or too expensive to compute).

Any algorithm based on the hypervolume is limited by the hypervolume's computational complexity increasing in the number of objectives  $m > 2$ , e.g.,  $O(p^{m-2} \log(p))$  [11]. However, the approximation sets we consider are rather small (e.g.,  $p = 9$ ), and UHV gradient ascent shows good scalability in  $p$ , which encourages investigating cases with  $m > 2$  objectives.

Finally, as any gradient-based algorithm, UHV gradient ascent suffers from the risk of ending up in local optima. A future research direction therefore is to hybridize UHV gradient ascent and EAs. Hybridization is however not trivial [4, 22] as both EAs and gradient-based algorithms rely on information of preceding iterations, and interleaving different algorithms might disrupt these mechanisms.

**Acknowledgments.** This work was supported by the Dutch Research Council (NWO) through Gravitation Programme Networks 024.002.003 and is part of the research programme Open Technology Programme with project number 15586, which is financed by NWO, Elekta, and Xomnia. Further, this project is co-funded by the public-private partnership allowance for top consortia for knowledge and innovation (TKIs) from the Ministry of Economic Affairs.

## 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 - FOGA 2009, pp. 87–102. ACM Press, New York (2009)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
3. Bosman, P.A.N., Thierens, D.: The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Trans. Evol. Comput.* **7**(2), 174–188 (2003)
4. Bosman, P.A.N.: On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization. *IEEE Trans. Evol. Comput.* **16**(1), 51–69 (2011)
5. Deb, K.: *Multi-Objective Optimization*. Wiley, Chichester (2001)
6. Désidéri, J.A.: Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *C.R. Math.* **350**(5–6), 313–318 (2012)
7. Emmerich, M.T.M., Deutz, A.H.: A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Nat. Comput.* **17**(3), 585–609 (2018). <https://doi.org/10.1007/s11047-018-9685-y>
8. Emmerich, M., Deutz, A.: Time complexity and zeros of the hypervolume indicator gradient field. In: Schuetze, O., et al. (eds.) *EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation III*, pp. 169–193. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-319-01460-9\\_8](https://doi.org/10.1007/978-3-319-01460-9_8)
9. Fleischer, M.: The measure of pareto optima applications to multi-objective etaheristics. 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)

10. Fliege, J., Svaiter, B.F.: Steepest descent methods for multicriteria optimization. *Math. Methods Oper. Res.* **51**(3), 479–494 (2000)
11. Fonseca, C.M., Paquete, L., López-Ibáñez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: 2006 IEEE International Conference on Evolutionary Computation, pp. 1157–1163. IEEE (2006)
12. Sosa Hernández, V.A., Schütze, O., Emmerich, M.: Hypervolume maximization via set based Newton’s method. In: Tantar, A.A., et al. (eds.) *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*. AISC, vol. 288, pp. 15–28. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-07494-8\\_2](https://doi.org/10.1007/978-3-319-07494-8_2)
13. Huband, S., Barone, L., While, L., Hingston, P.: Walking fish group toolkit: C++ source code. <http://www.wfg.csse.uwa.edu.au/toolkit/>. Accessed 06 Apr 2020
14. 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)
15. 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)
16. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
17. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: *Proceedings of the 2nd Berkeley Symposium on Mathematical and Statistical Probability*, pp. 481–492. University of California Press (1951)
18. Li, M., Yao, X.: Quality evaluation of solution sets in multiobjective optimisation: a survey. *ACM Comput. Surv. (CSUR)* **52**(2), 1–38 (2019)
19. Maree, S.C., Alderliesten, T., Bosman, P.A.N.: Uncrowded hypervolume-based multi-objective optimization with gene-pool optimal mixing. arXiv preprint [arXiv:2004.05068](https://arxiv.org/abs/2004.05068) (2020)
20. Peitz, S., Dellnitz, M.: Gradient-based multiobjective optimization with uncertainties. arXiv preprint [arXiv:1612.03815v2](https://arxiv.org/abs/1612.03815v2) (2017)
21. Schäffler, S., Schultz, R., Weinzierl, K.: Stochastic method for the solution of unconstrained vector optimization problems. *J. Optim. Theory Appl.* **114**(1), 209–222 (2002)
22. Schütze, O., Hernández, V.A.S., Trautmann, H., Rudolph, G.: The hypervolume based directed search method for multi-objective optimization problems. *J. Heuristics* **22**(3), 273–300 (2016). <https://doi.org/10.1007/s10732-016-9310-0>
23. Schütze, O., Martín, A., Lara, A., Alvarado, S., Salinas, E., Coello, C.A.C.: The directed search method for multi-objective memetic algorithms. *Comput. Optim. Appl.* **63**(2), 305–332 (2015). <https://doi.org/10.1007/s10589-015-9774-0>
24. Touré, C., Hansen, N., Auger, A., Brockhoff, D.: Uncrowded hypervolume improvement: COMO-CMA-ES and the sofomore framework. In: *Proceedings of the Genetic and Evolutionary Computation Conference, New York, NY, USA*, pp. 638–646 (2019)
25. Wang, H.: Hypervolume indicator gradient ascent multi-objective optimization. <https://github.com/wangronin/HIGA-MO>. Accessed 11 April 2020
26. Wang, H., Deutz, A., Bäck, T., Emmerich, M.: Hypervolume indicator gradient ascent multi-objective optimization. In: Trautmann, H., et al. (eds.) *EMO 2017*. LNCS, vol. 10173, pp. 654–669. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-54157-0\\_44](https://doi.org/10.1007/978-3-319-54157-0_44)

27. Wang, H., Ren, Y., Deutz, A., Emmerich, M.: On steering dominated points in hypervolume indicator gradient ascent for bi-objective optimization. In: Schütze, O., Trujillo, L., Legrand, P., Maldonado, Y. (eds.) NEO 2015. SCI, vol. 663, pp. 175–203. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-44003-3\\_8](https://doi.org/10.1007/978-3-319-44003-3_8)
28. Wessing, S.: Optproblems: Infrastructure to define optimization problems and some test problems for black-box optimization. Python package version 1.2 (2018)
29. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Trans. Evol. Comput.* **7**(2), 117–132 (2003)
30. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms — a case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P. (eds.) *Parallel Problem Solving from Nature – PPSN V*. PPSN 1998. Lecture Notes in Computer Science, vol. 1498, pp. 292–301. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0056872>



# An Ensemble Indicator-Based Density Estimator for Evolutionary Multi-objective Optimization

Jesús Guillermo Falcón-Cardona<sup>1</sup>, Arnaud Liefooghe<sup>2</sup>,  
and Carlos A. Coello Coello<sup>1</sup>

<sup>1</sup> Computer Science Department, CINVESTAV-IPN, 07300 Mexico City, Mexico

[jfalcon@computacion.cs.cinvestav.mx](mailto:jfalcon@computacion.cs.cinvestav.mx), [ccoello@cs.cinvestav.mx](mailto:ccoello@cs.cinvestav.mx)

<sup>2</sup> JFLI – CNRS IRL 3527, University of Tokyo, Tokyo 113-0033, Japan  
[arnaud.liefooghe@univ-lille.fr](mailto:arnaud.liefooghe@univ-lille.fr)

**Abstract.** Ensemble learning is one of the most employed methods in machine learning. Its main ground is the construction of stronger mechanisms based on the combination of elementary ones. In this paper, we employ AdaBoost, which is one of the most well-known ensemble methods, to generate an ensemble indicator-based density estimator for multi-objective optimization. It combines the search properties of five density estimators, based on the hypervolume, R2, IGD<sup>+</sup>,  $\epsilon^+$ , and  $\Delta_p$  quality indicators. Through the multi-objective evolutionary search process, the proposed ensemble mechanism adapts itself using a learning process that takes the preferences of the underlying quality indicators into account. The proposed method gives rise to the ensemble indicator-based multi-objective evolutionary algorithm (EIB-MOEA) that shows a robust performance on different multi-objective optimization problems when compared with respect to several existing indicator-based multi-objective evolutionary algorithms.

**Keywords:** Multi-objective optimization · Quality indicators · Ensemble learning · AdaBoost

## 1 Introduction

In many scientific and industrial fields arise the so-called multi-objective optimization problems (MOPs), that involve the simultaneous optimization of two or more conflicting objective functions. Mathematically, an MOP is defined as follows:

$$\min_{\mathbf{x} \in \Omega} \{\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))\}, \quad (1)$$

---

The first author acknowledges support from CONACyT and CINVESTAV-IPN to pursue graduate studies in Computer Science. The third author gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (*Investigación en Fronteras de la Ciencia 2016*) and from a SEP-Cinvestav grant (application no. 4).

© Springer Nature Switzerland AG 2020

T. Bäck et al. (Eds.): PPSN 2020, LNCS 12270, pp. 201–214, 2020.

[https://doi.org/10.1007/978-3-030-58115-2\\_14](https://doi.org/10.1007/978-3-030-58115-2_14)

where  $\mathbf{x}$  is the vector of decision variables,  $\Omega \subseteq \mathbb{R}^n$  is the decision space and  $\mathbf{F}(\mathbf{x})$  is the vector of  $m \geq 2$  objective functions such that  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$  for  $i \in \{1, 2, \dots, m\}$ . Unlike single-objective optimization problems which have a single global optimal solution, the solution of an MOP is a set of solutions that represents the best possible trade-offs among the objective functions. Given  $\mathbf{x}, \mathbf{y} \in \Omega$  and  $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , we say that  $\mathbf{x}$  dominates  $\mathbf{y}$ , denoted as  $\mathbf{F}(\mathbf{x}) \prec \mathbf{F}(\mathbf{y})$ , if and only if  $\forall i \in \{1, \dots, m\}, f_i(\mathbf{x}) \leq f_i(\mathbf{y})$  and there exists at least an index  $j \in \{1, \dots, m\}$  such that  $f_j(\mathbf{x}) < f_j(\mathbf{y})$ . The particular set that yields the optimum values, according to the Pareto dominance relation, is the Pareto set; its image is known as the Pareto front.

Multi-objective evolutionary algorithms (MOEAs) constitute a popular choice to tackle complex MOPs [1]. MOEAs are stochastic black-box optimizers based on the principles of Darwin's natural selection. MOEAs are population-based metaheuristics that can generate a Pareto front approximation (or approximation set) in a single execution. Ideally, an MOEA should produce solutions as close as possible to the Pareto front, covering it all and with good diversity. There exist four main design methodologies for MOEAs [1]: (1) MOEAs using the Pareto dominance relation or any of its relaxed forms, (2) decomposition-based MOEAs, (3) reference set-based MOEAs, and (4) indicator-based MOEAs (IB-MOEAs). In the last fifteen years, IB-MOEAs have attracted considerable attention due to their ability to solve MOPs having more than three objective functions (i.e., the so-called many-objective optimization problems) [2]. The underlying idea of IB-MOEAs is the use of a quality indicator (QI) [3], which is a set function that evaluates the quality of an approximation set based on specific preferences, in order to guide the evolutionary search process by focusing on the selection mechanisms. Currently, there exist several QIs, such as the hypervolume indicator (HV) [4], R2 [5], the inverted generational distance plus (IGD<sup>+</sup>) [6], the additive epsilon indicator ( $\epsilon^+$ ) [7], and the averaged Hausdorff distance ( $\Delta_p$ ) [8], being these ones the most popular within the currently available IB-MOEAs [2].

An IB-MOEA produces a Pareto front approximation exhibiting the preferences of its underlying QI [9]. As such, different IB-MOEAs yield different results in terms of the distribution of solutions in the approximation set, due to the underlying properties of the QI they employ. Moreover, there are MOPs where a specific IB-MOEA performs well, but there are others on which it does not. As a consequence, it is not clear which QI to consider beforehand, and an open question is whether a set of existing indicator-based selection mechanisms can create a single operator that reaches a consensus that outperforms the existing ones. In 2011, Phan and Suzuki were the first to investigate this question by boosting a set of indicator-based mating selection operators [10]. The boosted indicator-based mating selection operator uses 15 quality indicators from which it ensembles the best suited ones using the AdaBoost algorithm [11] with an offline training. The proposed mechanism was embedded into the non-dominated sorting genetic algorithm II (NSGA-II) [12], giving rise to the boosted indicator-based evolutionary algorithm (BIBEA). According to the reported results, BIBEA was able

to outperform NSGA-II, exhibiting robustness in MOPs with different characteristics. Later on, Phan *et al.* [13] proposed BIBEA-P which allows BIBEA to use an additional ensemble indicator-based mechanism for environmental selection. Moreover, BIBEA-P uses Pdi-Boosting instead of AdaBoost. Similarly to BIBEA, the ensemble operators of BIBEA-P needs to be trained using a given MOP in an offline fashion. The experimental results showed that BIBEA-P is better than BIBEA, NSGA-II, and SMS-EMOA (which is a HV-based MOEA) [14] when using MOPs with different Pareto front shapes.

In this paper, we propose an ensemble indicator-based density estimator using the HV, R2, IGD<sup>+</sup>, ε<sup>+</sup>, and Δ<sub>p</sub> indicators. Unlike BIBEA and BIBEA-P, our mechanism adapts the combination of the indicator-based density estimators (IB-DEs) in an online fashion. The underlying reason to use an online ensemble mechanism is to allow our approach to produce high-quality results for problems with different characteristics, and to reach a robust performance with respect to the multiple indicators being considered. This approach allows our proposed ensemble indicator-based MOEA (EIB-MOEA) to tackle problems with different Pareto front geometries from the test suites DTLZ, DTLZ<sup>-1</sup>, WFG, and WFG<sup>-1</sup>. Furthermore, the proposed approach consistently obtains competitive results with respect to other IB-MOEA.

The remainder of this paper is organized as follows. Section 2 provides the mathematical definitions of the QIs that we consider in our analysis. Section 3 describes the algorithmic design of EIB-MOEA. Section 4 shows our experimental results and Sect. 5 provides our final conclusions and some possible paths for future work.

## 2 Background

In this section, we describe a selection of five QIs, corresponding to those which are most frequently used in the specialized literature. They will be considered as constituent QIs in our proposed ensemble indicator-based density estimator. In the following,  $\mathcal{A}$  denotes an approximation set.

**Definition 1 (Hypervolume indicator [4]).** *Given an anti-optimal reference point  $\mathbf{r} \in \mathbb{R}^m$ , the hypervolume is defined as follows:*

$$HV(\mathcal{A}, \mathbf{r}) = \mathcal{L} \left( \bigcup_{\mathbf{a} \in \mathcal{A}} \{\mathbf{b} \mid \mathbf{a} \prec \mathbf{b} \prec \mathbf{r}\} \right), \tag{2}$$

where  $\mathcal{L}(\cdot)$  denotes the Lebesgue measure in  $\mathbb{R}^m$ .

**Definition 2 (Unary R2 indicator [5]).** *The unary R2 indicator is defined as follows:*

$$R2(\mathcal{A}, W) = -\frac{1}{|W|} \sum_{\mathbf{w} \in W} \max_{\mathbf{a} \in \mathcal{A}} \{u_{\mathbf{w}}(\mathbf{a})\}, \tag{3}$$

where  $W$  is a set of weight vectors and  $u_{\mathbf{w}} : \mathbb{R}^m \rightarrow \mathbb{R}$  is a scalarizing function defined by  $\mathbf{w} \in W$  that assigns a real value to each  $m$ -dimensional vector.

**Definition 3 (IGD<sup>+</sup> indicator [6]).** *The IGD<sup>+</sup>, for minimization, is defined as follows:*

$$IGD^+(\mathcal{A}, Z) = \frac{1}{|Z|} \sum_{z \in Z} \min_{\mathbf{a} \in \mathcal{A}} d^+(\mathbf{a}, z), \quad (4)$$

where  $d^+(\mathbf{a}, z) = \sqrt{\sum_{k=1}^m (\max\{a_k - z_k, 0\})^2}$ .

**Definition 4 (Unary  $\epsilon^+$  indicator [7]).** *The unary  $\epsilon^+$ -indicator gives the minimum distance by which a Pareto front approximation needs to or can be translated in each dimension in the objective space such that a reference set is weakly dominated. Mathematically, it is defined as follows:*

$$\epsilon^+(\mathcal{A}, Z) = \max_{z \in Z} \min_{\mathbf{a} \in \mathcal{A}} \max_{1 \leq i \leq m} \{a_i - z_i\}. \quad (5)$$

**Definition 5 ( $\Delta_p$  indicator [8]).** *For a given  $p > 0$ , the  $\Delta_p$  is defined as follows:*

$$\Delta_p(\mathcal{A}, Z) = \max \{GD_p(\mathcal{A}, Z), IGD_p(\mathcal{A}, Z)\}. \quad (6)$$

$\Delta_p$  is defined on the basis of two indicators:  $GD_p$  and  $IGD_p$  which are slight modifications of the indicators Generational Distance (GD) and Inverted Generational Distance (IGD) [3], respectively. These are defined in the following.

**Definition 6 ( $GD_p$  indicator [8])**

$$GD_p(\mathcal{A}, Z) = \left( \frac{1}{|\mathcal{A}|} \sum_{\mathbf{a} \in \mathcal{A}} d(\mathbf{a}, Z)^p \right)^{1/p}, \quad (7)$$

where  $d(\mathbf{a}, Z) = \min_{z \in Z} \sqrt{\sum_{i=1}^m (a_i - z_i)^2}$ .

**Definition 7 ( $IGD_p$  indicator [8]).** *It is defined as follows:  $IGD_p(\mathcal{A}, Z) = GD_p(Z, \mathcal{A})$ .*

**Definition 8 (Indicator contribution).** *Let  $\mathcal{I}$  be any indicator in the set  $\{HV, R2, IGD^+, \epsilon^+, \Delta_p\}$ . The individual contribution  $C$  of a solution  $\mathbf{a} \in \mathcal{A}$  to the indicator value is given as follows:*

$$C_{\mathcal{I}}(\mathbf{a}, \mathcal{A}) = |\mathcal{I}(\mathcal{A}) - \mathcal{I}(\mathcal{A} \setminus \{\mathbf{a}\})|. \quad (8)$$

Interestingly, the QIs presented above have different properties, and express different preferences in terms of set approximation quality [7]. Moreover, they do not always agree with each other [15], so that good-quality approximation sets for a given QI typically contain different solutions than for other QIs. This motivates the ensemble indicator-based approach introduced below.

### 3 The Proposed EIB-MOEA Approach

In this section, we first give the general description of EIB-MOEA, then we detail the learning model and the adaptive strategy considered to update the relative importance given to each QI at different iterations.



**Algorithm 1.** EIB-MOEA's general framework

---

**Require:** Set of indicators  $\{I_1, \dots, I_k\}$ ; time window size  $T_w$   
**Ensure:** Pareto front approximation

- 1: Randomly initialize population  $\mathcal{A}$
- 2:  $w_i = 1/k, i \in \{1, \dots, k\}$
- 3: Initialize performance matrix  $P \in \mathbb{R}^{k \times T_w}$
- 4: Initialize learning matrix  $\Psi \in \{0, 1\}^{k \times T_w}$
- 5:  $g = 0$
- 6: **while** stopping criterion is not fulfilled **do**
- 7:   Create an offspring solution  $q$  based on  $\mathcal{A}$
- 8:    $Q = \mathcal{A} \cup \{q\}$
- 9:    $\{R_1, \dots, R_\ell\} = \text{NondominatedSorting}(Q)$
- 10:   **if**  $|R_\ell| > 1$  **then**
- 11:      $z_i^{\min} = \min_{a \in \mathcal{A}} f_i(a), i \in \{1, \dots, m\}$
- 12:      $z_i^{\max} = \max_{a \in \mathcal{A}} f_i(a), i \in \{1, \dots, m\}$
- 13:     Normalize  $\{R_1, \dots, R_\ell\}$  using  $z^{\min}$  and  $z^{\max}$
- 14:     **for**  $j = 1$  to  $k$  **do**
- 15:        $C_{I_j}(r, R_\ell) = |I_j(R_\ell) - I_j(R_\ell \setminus \{\mathbf{F}(r)\})|, \forall r \in R_\ell$
- 16:       Sort  $C_{I_j}$  in ascending order
- 17:        $\forall z \in R_\ell$ , compute  $\text{rank}_{I_j}(\mathbf{F}(r))$ , using the sorted  $C_{I_j}$
- 18:     **end for**
- 19:      $\mathbf{a}_{\text{worst}} = \arg \min_{r \in R_\ell} \left\{ H(z = \mathbf{F}(r)) = \sum_{j=1}^k w_j \text{rank}_{I_j}(z) \right\}$
- 20:     Learning( $Q, R_\ell, \{I_1, \dots, I_k\}, g, \mathbf{a}_{\text{worst}}, P, \Psi$ )
- 21:      $g = g + 1$
- 22:   **else**
- 23:     Let  $\mathbf{a}_{\text{worst}}$  be the sole solution in  $R_\ell$
- 24:   **end if**
- 25:    $\mathcal{A} = Q \setminus \{\mathbf{a}_{\text{worst}}\}$
- 26:   **if**  $g = T_w$  **then**
- 27:     UpdateWeights( $\mathbf{w}, P, \Psi, T_w, k$ )
- 28:      $g = 0$
- 29:   **end if**
- 30: **end while**
- 31: **return**  $\mathcal{A}$

---

### 3.1 General Description

The proposed EIB-MOEA is a steady-state MOEA based on SMS-EMOA [14]. Its general framework is outlined in Algorithm 1. EIB-MOEA requires a set of  $k$  indicators  $\{I_1, \dots, I_k\}$  and a time window frame  $T_w$  as input parameters. In Line 2, all the components of the weight vector  $\mathbf{w}$  are set to  $1/k$ . This weight vector is employed in the ensemble indicator-based density estimator (EIB-DE), and contains the relative importance given to each indicator at the current iteration. Lines 6 to 30 describe the main loop of EIB-MOEA. At each iteration, a single offspring solution  $q$  is created using variation operators. This newly created solution is added to the population  $\mathcal{A}$  to create the temporary population  $Q$ . The non-dominated sorting algorithm [12] processes  $Q$  to create a set of layers  $\{R_1, \dots, R_\ell\}$ . If  $R_\ell$  contains more than one solution, EIB-DE is executed. First, the population is normalized in Line 13. Then, for each indicator  $I_j, j \in \{1, \dots, k\}$ , the individual indicator contributions of all solutions in  $R_\ell$  are computed and stored in the vector  $C_{I_j}$ . By sorting this vector in ascending order, for each  $r \in R_\ell$  we obtain  $\text{rank}_{I_j}(\mathbf{F}(r)) \in \{1, 2, \dots, |R_\ell|\}$  that returns the ranking of the solution in the sorted  $C_{I_j}$ , where rank 1 corresponds to the worst-contributing solution to  $I_j$ . In Line 19, the worst-contributing solution,

---

**Algorithm 2.** Learning

---

**Require:** Population  $\mathcal{A}$ ; worst set  $R$ ; set of indicators  $\{I_1, \dots, I_k\}$ ; index  $t$ ; selected solution  $\mathbf{a}_{\text{worst}}$ ; performance matrix  $P$ ; learning matrix  $\Psi$

**Ensure:** Updated  $\Psi$

- 1: **for**  $j = 1$  to  $k$  **do**
- 2:      $\mathbf{a}_{\text{worst}}^j = \arg \min_{r \in R} |I_j(R) - I_j(R \setminus \{\mathbf{F}(r)\})|$
- 3:      $\mathcal{A}^j = \mathcal{A} \setminus \{\mathbf{a}_{\text{worst}}^j\}$
- 4:      $P_{jt} = I_j(\mathcal{A}^j)$
- 5:     **if**  $P_{jt} > P_{j,t-1 \bmod T_w} \wedge \mathbf{a}_{\text{worst}}^j = \mathbf{a}_{\text{worst}}$  **then**
- 6:          $\Psi_{jt} = 0$
- 7:     **else**
- 8:          $\Psi_{jt} = 1$
- 9:     **end if**
- 10: **end for**
- 11: **return**  $\Psi$

---

using EIB-DE, is obtained. The learning process (see Algorithm 2), which is a fundamental part to update the weight vector  $\mathbf{w}$ , is performed in Line 20, and then, the counter  $g$  is incremented by one. In Line 25,  $\mathbf{a}_{\text{worst}}$  is eliminated from  $Q$  to shape the population for the next generation. In case  $g$  is equal to  $T_w$ ,  $\mathbf{w}$  is updated following Algorithm 3 and  $g$  is set to zero. Finally, once the stopping condition is satisfied,  $\mathcal{A}$  is returned as the Pareto front approximation.

### 3.2 Learning Process

The learning process, described in Algorithm 2, is based on analyzing the behavior of the population using all indicators. For each indicator  $I_j, j \in \{1, \dots, k\}$ , we obtain its worst-contributing solution  $\mathbf{a}_{\text{worst}}^j$ , where  $R$  represents the last layer of solutions with respect to non-dominated sorting. In Line 3, we simulate the elimination of  $\mathbf{a}_{\text{worst}}^j$  from the population  $\mathcal{A}$  to generate the set  $\mathcal{A}^j$  that is assessed by  $I_j$ . This indicator value is stored in the performance matrix at position  $(j, t)$ , i.e.,  $P_{jt} = I_j(\mathcal{A}^j)$ . It is worth noting that each row of  $P$ , represented as  $P_j$ , works as a circular array of size  $T_w$ . If  $P_{jt}$  is greater than the previous sample in  $P_j$  (which implies an increase in quality) and  $\mathbf{a}_{\text{worst}}^j$  is the same as the worst-contributing solution to EIB-DE, the selection is marked as successful and a zero value is stored in the learning matrix  $\Psi$  in the same position  $(j, t)$ . Otherwise, we set  $\Psi_{jt} = 1$ .

### 3.3 Updating the Relative Importance of QIs

After executing EIB-DE and the learning algorithm a total of  $T_w$  times, the weight vector has to be updated. Algorithm 3 sketches the update process which is based on the AdaBoost algorithm [11], whose aim is to minimize the exponential loss. For each indicator  $I_j, j \in \{1, \dots, k\}$ , the selection error  $e_j$  is calculated using the  $j^{\text{th}}$  row of the learning matrix  $\Psi$ , taking into account that  $e_j$  should be in the open interval  $(0, 1)$  to avoid numerical problems in the calculation of the factor  $\alpha_j$ . Using the indicator values in  $P_j$ , a linear model is constructed to obtain its angle  $\theta$ . In Line 7, we set the weight  $w_j = w_j e^{-\alpha_j}$  if  $\theta$  is strictly

**Algorithm 3.** UpdateWeights

---

**Require:** Weight vector  $w$ ; performance matrix  $P$ ; learning matrix  $\Psi$ ; time window size  $T_w$ ; number of indicators  $k$

**Ensure:** Updated  $w$

- 1: **for**  $j = 1$  to  $k$  **do**
- 2:    $e_j = \frac{w_j}{T_w} \sum_{i=1}^{T_w} \Psi_{ji}$
- 3:   Validate that  $e_j \in (0, 1)$
- 4:    $\alpha_j = \frac{1}{2} \ln \left( \frac{1-e_j}{e_j} \right)$
- 5:   Build linear performance model based on  $P_j$
- 6:   Get the angle  $\theta_j$  of the linear model
- 7:    $w_j = \begin{cases} w_j e^{-\alpha_j}, & \theta > 0 \\ w_j e^{\alpha_j}, & \text{otherwise} \end{cases}$
- 8:   Validate that  $w_j > 0$
- 9: **end for**
- 10:  $w_j = \frac{w_j}{\sum_{i=1}^k w_i}, j \in \{1, \dots, k\}$
- 11: **return**  $w$

---

positive, which implies an increasing quality of the population due to the use of the density estimator based on  $I_j$ . Otherwise, we set  $w_j = w_j e^{\alpha_j}$ . To avoid having EIB-DE composed of a single indicator, we do not allow the existence of zero weights. At last, all weights are normalized in Line 10 and the updated weight vector is returned.

## 4 Experimental Analysis

In this section, we analyze the performance of the proposed approach<sup>1</sup>. First, we compare EIB-MOEA with its average ranking version, i.e, an EIB-MOEA where the weights for the ensemble are the same for all indicators (denoted as avgEIB-MOEA) to show that the adaptive mechanism produces better quality results. Then, we perform an exhaustive analysis where we compare EIB-MOEA with SMS-EMOA, R2-EMOA, IGD<sup>+</sup>-MaOEA,  $\epsilon^+$ -MaOEA, and  $\Delta_p$ -MaOEA, which are all steady-state MOEAs using density estimators based on the HV, R2, IGD<sup>+</sup>,  $\epsilon^+$ , and  $\Delta_p$  indicators, respectively. In all test instances, each MOEA is independently executed 30 times.

### 4.1 Parameters Settings

We employ the benchmark functions DTLZ1, DTLZ2, DTLZ5, DTLZ7, WFG1, WFG2, WFG3, and WFG4, together with their corresponding minus versions proposed in [16] for two and three objective functions. We adopted these problems because they all have different search difficulties and Pareto front shapes. The number  $n$  of decision variables was set as follows. For DTLZ instances and their minus versions,  $n = m + K - 1$ , where  $m$  is the number of objective functions and  $K = 5$  for DTLZ1,  $K = 10$  for both DTLZ2 and DTLZ5, and  $K = 20$

<sup>1</sup> The source code of EIB-MOEA is available at <http://computacion.cs.cinvestav.mx/~jfalcon/Ensemble/EIB-MOEA.html>.

for DTLZ7. Regarding the WFG and WFG<sup>-1</sup> test problems,  $n$  was set to 24 and 26, for two- and three-objective instances and in both cases the number of position-related parameters was set to 2. For a fair comparison, all the MOEAs employ the same population size  $\mu = 120$ , and the same variation operators: simulated binary crossover (SBX) and polynomial-based mutation (PBM) [12] for all test instances. The crossover probability is set to 0.9, the mutation probability is  $1/n$  (where  $n$  is the number of decision variables), and both the crossover and mutation distribution indexes are set to 20. We considered 50,000 function evaluations as the stopping criterion for all MOPs. We employ the achievement scalarizing function for the R2-based density estimator. In every generation, we employ the currently population's non-dominated solutions as the reference set required by IGD<sup>+</sup>,  $\epsilon^+$ , and  $\Delta_p$ . Regarding EIB-MOEA and avgEIB-MOEA, we set  $T_w = \mu$ .

## 4.2 Experimental Results

For the performance assessment of EIB-MOEA, avgEIB-MOEA and the other IB-MOEAs, we used eight quality indicators: HV, HV relative deviation (HVRD), R2, IGD<sup>+</sup>,  $\epsilon^+$ ,  $\Delta_p$ , and, for diversity, we employed Riesz  $s$ -energy [17] and the Solow-Polasky Diversity indicator [18]. The indicator values for two- and three-objective instances of the DTLZ and DTLZ<sup>-1</sup> test problems are shown with boxplots in Figs. 1 and 2, respectively. The boxplots for the WFG and WFG<sup>-1</sup> instances with two and three objective functions correspond to Figs. 3 and 4, respectively. Figure 5 shows the statistical ranks obtained by each algorithm over all benchmark functions with respect to each considered indicator. For a given benchmark function, the rank corresponds to the number of algorithms that significantly outperform the algorithm under consideration with respect to a Mann-Whitney non-parametric statistical test with a  $p$ -value of 0.05 and a Bonferroni correction (a lower value is better). The complete numerical results related to Fig. 5 are available at <http://computacion.cs.cinvestav.mx/~jfalco/Ensemble/EIB-MOEA.html> due to space limitations.

Regarding the comparison of EIB-MOEA with avgEIB-MOEA, Fig. 5 shows that the former gets better statistical ranks for all the considered indicators except for  $\Delta_p$ . From these QIs, the increase in quality is more evident for the hypervolume indicator. This means that the online ensemble allows EIB-MOEA to produce approximation sets closer to the Pareto front. This is supported by the other convergence indicators. However, producing better convergent approximation sets is not strictly related to producing higher diversity, as shown by the Riesz  $s$ -energy and SPD values, where EIB-MOEA is hardly better than avgEIB-MOEA. Overall, these results support that EIB-MOEA performs better than avgEIB-MOEA. On the other hand, for the comparison of EIB-MOEA

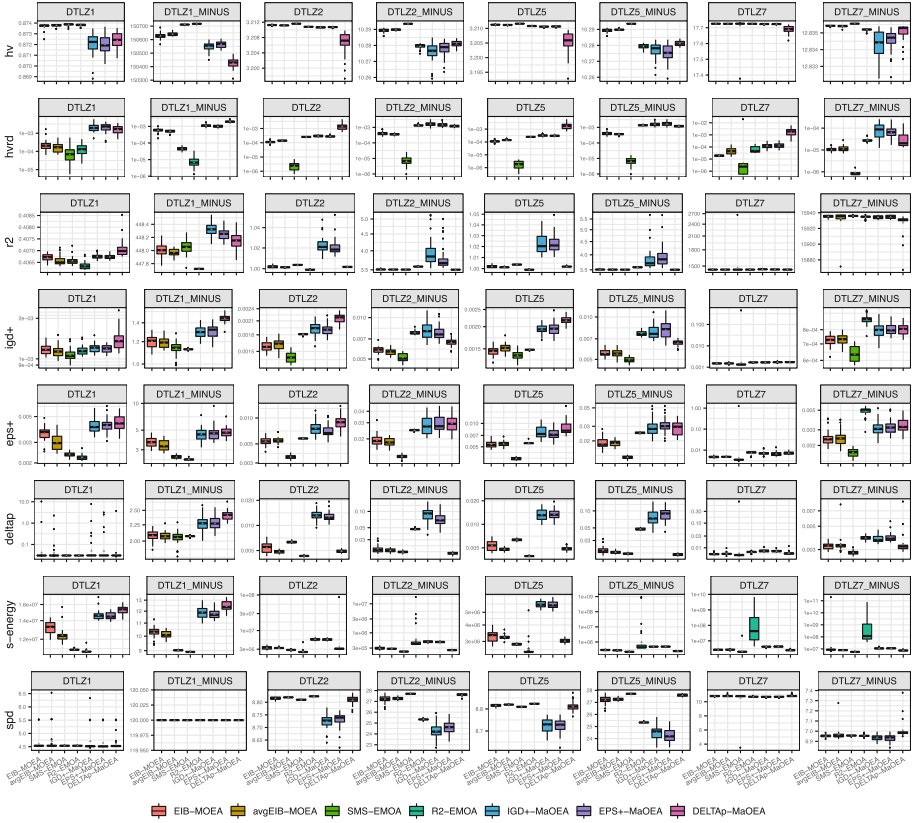
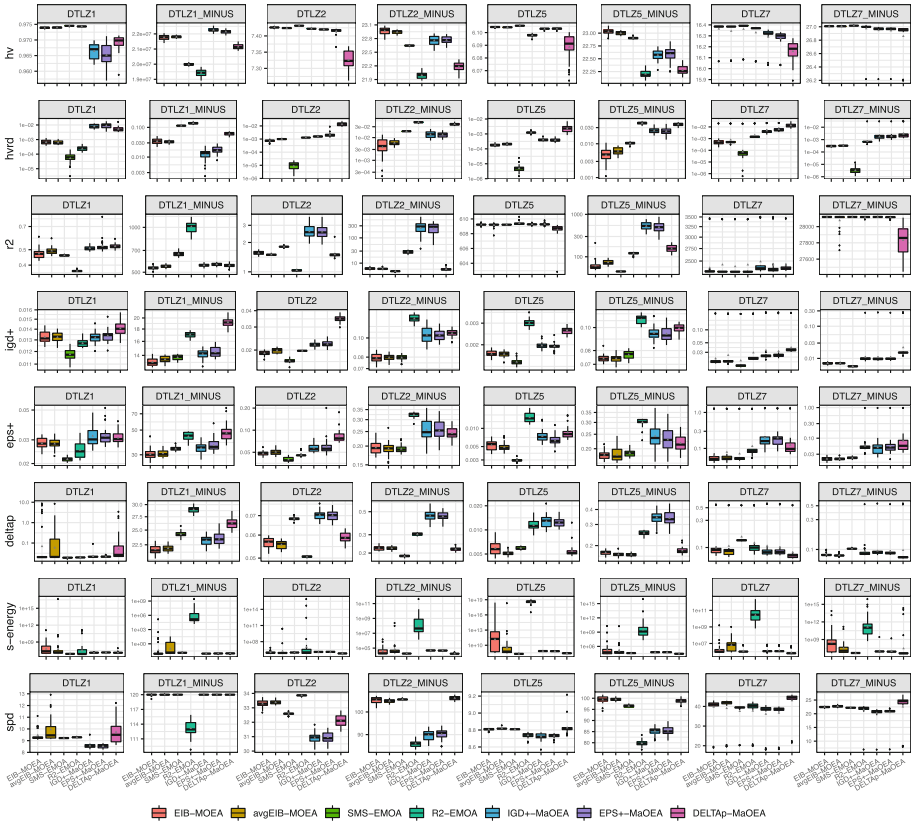


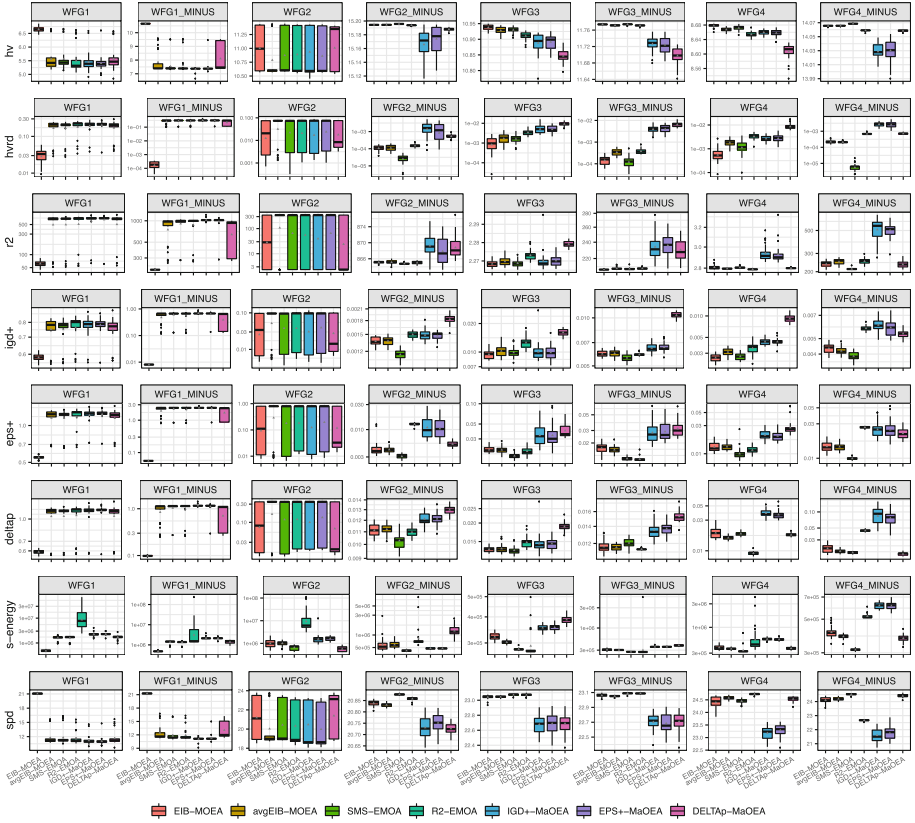
Fig. 1. Indicator values for two-objective DTLZ benchmark functions.

against the steady-state IB-MOEA, Fig. 5 shows that our proposed approach maintains a robust performance over all the considered QIs. Figures 1, 2, 3 and 4 illustrate that EIB-MOEA and SMS-EMOA obtained the best HV values. Overall, SMS-EMOA performs better on the original benchmark problems, but the quality of its approximate Pareto fronts is just slightly better than those produced by EIB-MOEA. In contrast, for the  $DTLZ^{-1}$  and  $WFG^{-1}$  test suites, EIB-MOEA significantly outperforms SMS-EMOA. This is because EIB-MOEA is able to produce Pareto front approximations with better distribution and coverage, due to the influence of all the indicators, for these modified problems, which increases the HV value. Moreover, for the original problems, EIB-MOEA still produces good approximations but with other type of distribution that does



**Fig. 2.** Indicator values for three-objective DTLZ benchmark functions.

not maximize the hypervolume value as SMS-EMOA does. In consequence, EIB-EMOA performs more robustly than SMS-EMOA. Additionally, for  $IGD^+$  and  $\epsilon^+$  which are QIs whose preferences are highly correlated to those of HV, Fig. 5 shows a similar behavior as in the case of HV. This is also supported by the detailed boxplots reported for the different test problems. Regarding the R2 indicator, R2-EMOA presents the best results for MOPs whose Pareto front maps to the simplex shape; e.g., DTLZ1, DTLZ2, and WFG4. This behavior is expected since R2-EMOA uses a set of convex weight vectors [16]. However, for the  $DTLZ^{-1}$  and  $WFG^{-1}$  test suites, R2-EMOA does not perform well and EIB-MOEA presents the best overall results. This indicates that the ensemble mechanism of EIB-MOEA allows to circumvent the weaknesses of the individual indicator-based density estimators, in this case the one based on R2. Finally,



**Fig. 3.** Indicator values for two-objective WFG benchmark functions.

in terms of diversity, Figs. 1, 2, 3 and 4 show that EIB-MOEA generates well-diversified approximation sets when dealing with MOPs whose Pareto front is irregular; i.e., different from the simplex shape. This is the case, for example, of WFG1, WFG1<sup>-1</sup>, DTLZ1<sup>-1</sup>, and DTLZ<sup>-1</sup>. Nevertheless, EIB-MOEA is able to produce competitive results with respect to Riesz *s*-energy and SPD, while SMS-EMOA is the best-ranked algorithm for the former indicator and  $\Delta_p$ -MaOEA is the best for the latter. As such, although EIB-MOEA is able to obtain very good HV values, there is still room for improvement in terms of diversity, e.g. by adding diversity-related indicators into the ensemble controlled by EIB-MOEA.



Fig. 4. Indicator values for three-objective WFG benchmark functions.

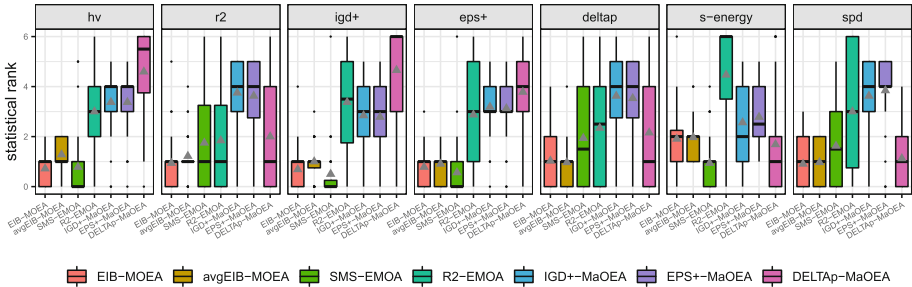


Fig. 5. Statistical ranks obtained by each algorithm over all benchmark functions with respect to each considered indicator.



## 5 Conclusions and Future Work

In this paper, we explored the effectiveness of an ensemble indicator-based density estimator, using the AdaBoost algorithm. The proposed mechanism adapts the ensemble in an online fashion depending on the performance of the underlying density estimators based on the indicators HV, R2, IGD<sup>+</sup>,  $\epsilon^+$ , and  $\Delta_p$ . The adaptive ensemble mechanism was embedded into a steady-state MOEA, giving rise to the EIB-MOEA. First, we showed that EIB-MOEA outperforms an average ranking EIB-MOEA that sets all the weights to the same value for the ensemble. Then, we compared EIB-MOEA with respect to SMS-EMOA, R2-EMOA, IGD<sup>+</sup>-MaOEA,  $\epsilon^+$ -MaOEA, and  $\Delta_p$ -MaOEA. The experimental results showed that EIB-MOEA is able to maintain a robust performance with respect to multiple quality indicators. As part of our future work, we aim at studying the performance of a generational EIB-MOEA and at improving the learning mechanism for the ensemble. We would also like to assess the performance of our proposed EIB-MOEA in many-objective optimization problems.

## References

1. Li, B., Li, J., Tang, K., Yao, X.: Many-objective evolutionary algorithms: a survey. *ACM Comput. Surv.* **48**(1), 1–35 (2015)
2. Falcón-Cardona, J.G., Coello, C.A.C.: Indicator-based multi-objective evolutionary algorithms: a comprehensive survey. *ACM Comput. Surv.* **53**(2), 1–35 (2020)
3. Li, M., Yao, X.: Quality evaluation of solution sets in multiobjective optimisation: a survey. *ACM Comput. Surv.* **52**(2), 26:1–26:38 (2019)
4. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms— a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0056872>
5. Brockhoff, D., Wagner, T., Trautmann, H.: On the properties of the *R2* indicator. In: 2012 Genetic and Evolutionary Computation Conference (GECCO 2012), Philadelphia, USA, July 2012, pp. 465–472. ACM Press (2012). (ISBN 978-1-4503-1177-9)
6. 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)
7. 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)
8. Schütze, O., Esquivel, X., Lara, A., Coello, C.A.C.: Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Trans. Evol. Comput.* **16**(4), 504–522 (2012)
9. Falcón-Cardona, J.G., Coello, C.A.C.: Convergence and diversity analysis of indicator-based multi-objective evolutionary algorithms. In: 2019 Genetic and Evolutionary Computation Conference (GECCO 2019), Prague, Czech Republic, pp. 524–531. ACM Press (2019)

10. Phan, D.H., Suzuki, J.: Boosting indicator-based selection operators for evolutionary multiobjective optimization algorithms. In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, pp. 276–281. IEEE Press (2011)
11. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **55**(1), 119–139 (1997)
12. 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)
13. Phan, D.H., Suzuki, J., Hayashi, I.: Leveraging indicator-based ensemble selection in evolutionary multiobjective optimization algorithms. In: 2012 Genetic and Evolutionary Computation Conference (GECCO 2012), Philadelphia, USA, July 2012, pp. 497–504. ACM Press (2012). (ISBN 978-1-4503-1177-9)
14. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
15. Liefoghe, A., Derbel, B.: A correlation analysis of set quality indicator values in multiobjective optimization. In: 2016 Genetic and Evolutionary Computation Conference (GECCO 2016), Denver, Colorado, USA, 20–24 July 2016, pp. 581–588. ACM Press (2016). (ISBN 978-1-4503-4206-3)
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**(2), 169–190 (2017)
17. Hardin, D.P., Saff, E.B.: Discretizing manifolds via minimum energy points. *Not. AMS* **51**(10), 1186–1194 (2004)
18. Emmerich, M.T.M., Deutz, A.H., Krusselbrink, J.W.: On quality indicators for black-box level set approximation. In: Tantar, E., et al. (eds.) *EVOLVE - A Bridge Between Probability, Set Oriented Numerics and Evolutionary Computation*. SCL, vol. 447, pp. 157–185. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-32726-1\\_4](https://doi.org/10.1007/978-3-642-32726-1_4). (978-3-642-32725-4, Chapter 4)



# Ensuring Smoothly Navigable Approximation Sets by Bézier Curve Parameterizations in Evolutionary Bi-objective Optimization

Stefanus C. Maree<sup>1</sup>(✉), Tanja Alderliesten<sup>2</sup>, and Peter A. N. Bosman<sup>1</sup>

<sup>1</sup> Life Sciences and Health Research Group, Centrum Wiskunde and Informatica, Amsterdam, The Netherlands

{maree,peter.bosman}@cwi.nl

<sup>2</sup> Department of Radiation Oncology, Leiden University Medical Center, Leiden, The Netherlands

t.alderliesten@lumc.nl

**Abstract.** The aim of bi-objective optimization is to obtain an approximation set of (near) Pareto optimal solutions. A decision maker then navigates this set to select a final desired solution, often using a visualization of the approximation front. The front provides a navigational ordering of solutions to traverse, but this ordering does not necessarily map to a smooth trajectory through decision space. This forces the decision maker to inspect the decision variables of each solution individually, potentially making navigation of the approximation set unintuitive. In this work, we aim to improve approximation set navigability by enforcing a form of smoothness or continuity between solutions in terms of their decision variables. Imposing smoothness as a restriction upon common domination-based multi-objective evolutionary algorithms is not straightforward. Therefore, we use the recently introduced uncrowded hypervolume (UHV) to reformulate the multi-objective optimization problem as a single-objective problem in which parameterized approximation sets are directly optimized. We study here the case of parameterizing approximation sets as smooth Bézier curves in decision space. We approach the resulting single-objective problem with the genepool optimal mixing evolutionary algorithm (GOMEA), and we call the resulting algorithm BezEA. We analyze the behavior of BezEA and compare it to optimization of the UHV with GOMEA as well as the domination-based multi-objective GOMEA. We show that high-quality approximation sets can be obtained with BezEA, sometimes even outperforming the domination- and UHV-based algorithms, while smoothness of the navigation trajectory through decision space is guaranteed.

**Keywords:** Evolutionary algorithm · Multi-objective optimization · Hypervolume · Bézier curve estimation · Approximation set navigation

## 1 Introduction

The aim of multi-objective optimization is to obtain a set of solutions that is as close as possible to the set of Pareto-optimal solutions, with different trade-offs between the objective functions. A decision maker can then navigate the obtained set, called the *approximation set*, to select a desired solution. The decision maker often incorporates external factors in the selection process that are not taken into account in the optimization objectives. An inspection of the decision variables of individual solutions is therefore required to determine their desirability. To guide the selection in bi-objective optimization, a visualization of the *approximation front* (i.e., the approximation set mapped to objective space) or trade-off curve can be used. The approximation front then intuitively implies a navigational order of solutions by traversing the front from one end to the other. However, solutions with similar objective values could still have completely different decision values. The decision values of all solutions then need to be inspected individually and carefully because they may not change predictably when the approximation front is traversed. This could make navigation of the approximation set unintuitive and un insightful.

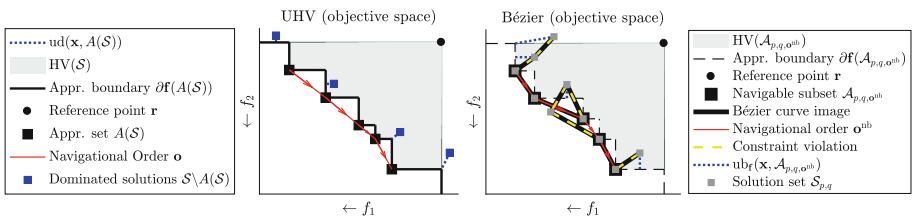
Population-based multi-objective evolutionary algorithms (MOEAs) have successfully been applied to real-world black-box optimization problems, for which the internal structure is unknown, or too complex to exploit efficiently by direct problem-specific design [6, 8, 22]. However, imposing a form of smoothness or continuity in terms of decision variables between solutions in the approximation set as a restriction upon the population of MOEAs is not straightforward. An underlying requirement to do so is that control over approximation sets as a whole is needed. However, typical dominance-based EAs use single-solution-based mechanics. Alternatively, multi-objective optimization problems can be formulated as a higher-dimensional single-objective optimization problem by using a quality indicator that assigns a fitness value to approximation sets. An interesting quality indicator is the hypervolume measure [23], as it is currently the only known Pareto-compliant indicator, meaning that an approximation set of given size with optimal hypervolume is a subset of the Pareto set [9, 13, 24]. However, the hypervolume measure has large drawbacks when used as quality indicator in indicator-based optimization, as it does not take dominated solutions into account. The uncrowded distance has been recently introduced to overcome this [20], which then resulted in the uncrowded hypervolume (UHV) measure [18]. The UHV can be used directly as a quality indicator for indicator-based multi-objective optimization. To be able to optimize approximation sets in this approach, fixed-size approximation sets are parameterized by concatenating the decision variables of a fixed number of solutions [2, 18, 21]. A single-objective optimizer can then be used to directly optimize approximation sets. The resulting single-objective optimization problem is however rather high-dimensional. To efficiently solve it, the UHV gene-pool optimal mixing evolutionary algorithm (UHV-GOMEA) [18], exploits grey-box properties of the UHV problem by only updating a subset of the decision variables corresponding to one (or a few) multi-objective solutions.

In this work, we go beyond an unrestricted concatenation of the decision variables of solutions and we propose to model approximation sets as sets of points that lie on a Bézier curve [10] in decision space. Optimizing only the control points of the Bézier curve, that define its curvature, enforces the decision variables of solutions in the approximation set to vary in a smooth, continuous fashion, thereby likely improving intuitive navigability of the approximation set. Previous work on parameterizations of the approximation set has been applied mainly in a post-processing step after optimization, or was performed in the objective space [3, 15, 19], but this does not aid in the navigability of the approximation set in decision space. Moreover, fitting a smooth curve through an already optimized set of solutions might result in a bad fit, resulting in a lower-quality approximation set. Additionally, we will show that specifying solutions as points on a Bézier curve directly enforces a form of diversity within the approximation set, which can actually aid in the optimization process, and furthermore reduces the problem dimensionality of the single-objective problem.

The remainder of this paper is organized as follows. In Sect. 2, we introduce preliminaries on UHV-based multi-objective optimization. In Sect. 3, we define a measure for navigational smoothness of approximation sets. In Sect. 4, we introduce Bézier curves and the corresponding optimization problem formulation. Empirical benchmarking on a set of benchmark problems is performed in Sect. 5. Finally, we discuss the results and conclude in Sect. 6.

## 2 UHV-Based Multi-objective Optimization

Let  $\mathbf{f} : \mathcal{X} \rightarrow \mathbb{R}^m$  be a to-be-minimized  $m$ -dimensional vector function and  $\mathcal{X} \subseteq \mathbb{R}^n$  be the  $n$ -dimensional (box-constrained) decision space. When the objectives in  $\mathbf{f}$  are conflicting, no single optimal solution exists, but the optimum of  $\mathbf{f}$  can be defined in terms of *Pareto optimality* [14]. A solution  $\mathbf{x} \in \mathcal{X}$  is said to *weakly dominate* another solution  $\mathbf{y} \in \mathcal{X}$ , written as  $\mathbf{x} \preceq \mathbf{y}$ , if and only if  $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$  for all  $i$ . When the latter relation is furthermore strict (i.e.,  $f_i(\mathbf{x}) < f_i(\mathbf{y})$ ) for at least one  $i$ , we say that  $\mathbf{x}$  *dominates*  $\mathbf{y}$ , written as  $\mathbf{x} \prec \mathbf{y}$ . A solution that is not dominated by any other solution in  $\mathcal{X}$  is called *Pareto optimal*. The *Pareto set*  $\mathcal{A}^*$  is the set of all Pareto optimal solutions, i.e.,  $\mathcal{A}^* = \{\mathbf{x} \in \mathcal{X} : \nexists \mathbf{y} \in \mathcal{X} : \mathbf{y} \prec \mathbf{x}\} \subset \mathcal{X}$ . The image of the Pareto set under  $\mathbf{f}$  is called the *Pareto*



**Fig. 1.** Illustration of the uncrowded hypervolume (UHV) [18] (left) for a bi-objective minimization problem, and the Bézier parameterization (right).

front, i.e.,  $\{\mathbf{f}(\mathbf{x}) : \mathbf{x} \in \mathcal{A}^*\} \subset \mathbb{R}^m$ . The aim of multi-objective optimization is to approximate the Pareto set with a set of non-dominated solutions called an *approximation set*  $\mathcal{A}$ . Let  $\mathcal{S} \subseteq \mathcal{X}$  be a solution set, that can contain dominated solutions and let  $A : \wp(\mathcal{X}) \rightarrow \wp(\mathcal{X})$  be the approximation set given by  $\mathcal{S}$ , i.e.,  $A(\mathcal{S}) = \{\mathbf{x} \in \mathcal{S} : \nexists \mathbf{y} \in \mathcal{S} : \mathbf{y} \prec \mathbf{x}\}$ , where  $\wp(\mathcal{X})$  is the powerset of  $\mathcal{X}$ .

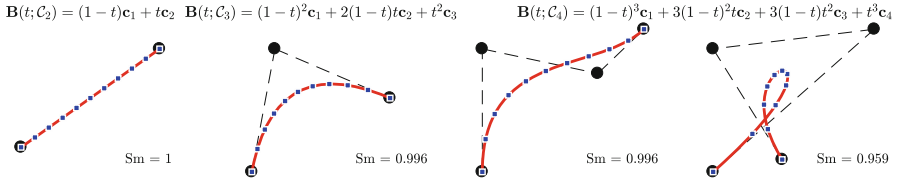
The hypervolume measure  $\text{HV} : \wp(\mathcal{X}) \rightarrow \mathbb{R}$  [1,24] measures the area or volume dominated by all solutions in the approximation set, bounded by a user-defined reference point  $\mathbf{r} \in \mathbb{R}^m$ , as shown in Fig. 1. As the hypervolume ignores dominated solutions, we use the *uncrowded distance* to assign a quality value to dominated solutions [20]. The uncrowded distance  $\text{ud}_{\mathbf{f}}(\mathbf{x}, \mathcal{A})$  measures the shortest Euclidean distance between  $\mathbf{x}$  and the *approximation boundary*  $\partial\mathbf{f}(\mathcal{A})$ , when  $\mathbf{x}$  is dominated by any solution in  $\mathcal{A}$  or outside the region defined by  $\mathbf{r}$ , and is defined  $\text{ud}_{\mathbf{f}}(\mathbf{x}, \mathcal{A}) = 0$  else (Fig. 1). It is called the uncrowded distance as the shortest distance to  $\partial\mathbf{f}(\mathcal{A})$  is obtained for a point on the boundary that is not in  $\mathcal{A}$  itself. Combining the uncrowded distance with the hypervolume measure results in the *uncrowded hypervolume* (UHV) [18],

$$\text{UHV}_{\mathbf{f}}(\mathcal{S}) = \text{HV}_{\mathbf{f}}(\mathcal{S}) - \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x} \in \mathcal{S}} \text{ud}_{\mathbf{f}}(\mathbf{x}, A(\mathcal{S}))^m. \tag{1}$$

We use the subscript  $\mathbf{f}$  to denote that its value is computed with respect to the multi-objective problem  $\mathbf{f}$ . To be able to optimize the UHV of a solution set, a parameterization of solution sets is required. Let  $\phi \in \mathbb{R}^l$  be such a parameterization consisting of  $l$  decision variables, and let  $S(\phi) = \{\mathbf{x}_1, \mathbf{x}_2, \dots\}$  be an operator that transforms  $\phi$  into its corresponding solution set. The resulting UHV-based optimization problem is then given by,

$$\begin{aligned} \text{maximize} \quad & \text{UHV}_{\mathbf{f},S}(\phi) = \text{HV}_{\mathbf{f}}(S(\phi)) - \frac{1}{|S(\phi)|} \sum_{\mathbf{x} \in S(\phi)} \text{ud}_{\mathbf{f}}(\mathbf{x}, A(S(\phi)))^m, \\ \text{with} \quad & \mathbf{f} : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad S : \mathbb{R}^l \rightarrow \wp(\mathcal{X}), \quad \phi \in \mathbb{R}^l. \end{aligned} \tag{2}$$

In a parameterization that is commonly used, solution sets  $\mathcal{S}_p$  of fixed size  $p$  are considered, and the decision variables of the solutions in  $\mathcal{S}_p$  are simply concatenated, i.e.,  $\phi = [\mathbf{x}_1 \cdots \mathbf{x}_p] \in \mathbb{R}^{p \cdot n}$  [2,18,21]. Using this parameterization, the resulting single-objective optimization problem is  $l = p \cdot n$  dimensional. In [18], GOMEA [5] was used to efficiently solve this problem by exploiting the *grey-box* (gb) property that not all solutions  $\mathbf{x}_i$  have to be recomputed when only some decision variables change. The resulting algorithm, which we call UHVEA-gb here (and was called UHV-GOMEA-Lm in [18]), greatly outperformed the mostly similar algorithm UHVEA-bb (called UHV-GOMEA-Lf in [18]) but in which the UHV was considered to be a *black box* (bb). This problem parameterization however does not guarantee any degree of navigational smoothness of the approximation set, which is the key goal in this paper.



**Fig. 2.** Illustration of Bézier curves (red) in decision space with different control points (black). Blue points correspond to  $p = 10$  evenly spread values of  $t$ , and the smoothness ( $S_m$ ) of these  $p$  points is given, computed based on  $\mathbf{o}^{\text{bez}}$ . (Color figure online)

### 3 A Measure for Navigational Smoothness

We introduce a measure for the navigational smoothness of an approximation set. Let  $\mathcal{S}_p = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}$  be an approximation set of size  $p$ . Furthermore, let the *navigational order*  $\mathbf{o}$  be a permutation of (a subset of)  $I = \{1, 2, \dots, p\}$ , representing the indices of the solutions in  $\mathcal{S}_p$  that the decision maker assesses in the order the solutions are inspected. The (*navigational*) *smoothness*  $S_m(\mathcal{S}_p, \mathbf{o})$  is then defined as,

$$S_m(\mathcal{S}_p, \mathbf{o}) = \frac{1}{p-2} \sum_{i=2}^{p-1} \frac{\|\mathbf{x}_{\mathbf{o}_{i-1}} - \mathbf{x}_{\mathbf{o}_{i+1}}\|}{\|\mathbf{x}_{\mathbf{o}_{i-1}} - \mathbf{x}_{\mathbf{o}_i}\| + \|\mathbf{x}_{\mathbf{o}_i} - \mathbf{x}_{\mathbf{o}_{i+1}}\|}. \quad (3)$$

This smoothness measure measures the *detour length*, i.e., the extra distance traveled (in decision space) when going to another solution via an intermediate solution, compared to directly going there.

Throughout this work, we will consider a navigational order  $\mathbf{o}$  for approximation sets  $\mathcal{A}$  such that  $f_1(\mathbf{x}_{\mathbf{o}_i}) < f_1(\mathbf{x}_{\mathbf{o}_j})$  holds whenever  $i < j$  holds, i.e., from left to right in the objective space plot Fig. 1. We therefore simply write  $S_m(\mathcal{A}, \mathbf{o}) = S_m(\mathcal{A})$  from now on. Note that  $S_m(\mathcal{A}) \in [0, 1]$ , and only if all solutions are colinear in decision space,  $S_m(\mathcal{A}) = 1$  holds. This we consider the ideal scenario, where the decision variables of solutions change perfectly predictably. This also implies that any other (continuous) non-linear curve is not considered to be perfectly smooth. Although one could argue for different definitions of smoothness, we will see later that this measure serves our purpose for distinguishing smoothly from non-smoothly navigable approximation sets.

### 4 Bézier Curve Parameterizations of Approximation Sets

A Bézier curve  $\mathbf{B}(t; \mathcal{C}_q)$  is a parametric curve that is commonly used in computer graphics and animations to model smooth curves and trajectories [10]. An  $n$ -dimensional Bézier curve is fully specified by an ordered set of  $q \geq 2$  control points  $\mathcal{C}_q = \{\mathbf{c}_1, \dots, \mathbf{c}_q\}$  with  $\mathbf{c}_j \in \mathbb{R}^n$ , and given by,

$$\mathbf{B}(t; \mathcal{C}_q) = \sum_{j=1}^q b_{j-1, q-1}(t) \mathbf{c}_j, \quad \text{with} \quad b_{j, q}(t) := \binom{q}{j} (1-t)^{q-j} t^j, \quad (4)$$

for  $0 \leq t \leq 1$ , where  $\binom{q}{j}$  are the binomial coefficients. Examples of Bézier curves are shown in Fig. 2. The first and last control points are always the end points of the Bézier curve, while intermediate control points do not generally lie on the curve. We parameterize a solution set  $\mathcal{S}_p = \{\mathbf{x}_1, \dots, \mathbf{x}_p\}$  of fixed size  $p$  using an  $n$ -dimensional Bézier curve  $\mathbf{B}(t; \mathcal{C}_q)$  with  $q$  control points. On this curve,  $p$  points  $\mathbf{x}_i = \mathbf{B}((i - 1)/(p - 1); \mathcal{C}_q)$  are selected, evenly spread in the domain of  $t$ . The resulting solution set  $\mathcal{S}_{p,q}(\phi) = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}$  is then given by,

$$\mathcal{S}_{p,q}(\phi) = \left\{ \mathbf{B} \left( \frac{0}{p-1}; \mathcal{C}_q \right), \mathbf{B} \left( \frac{1}{p-1}; \mathcal{C}_q \right), \dots, \mathbf{B} \left( \frac{p-1}{p-1}; \mathcal{C}_q \right) \right\},$$

with  $\phi = [\mathbf{c}_1 \dots \mathbf{c}_q] \in \mathbb{R}^{q \cdot n}$ . Note that inverting the order of control points does not affect the Bézier curve. To avoid this symmetry in the parameterization, we standardize the curve direction throughout optimization. After a change of the curve, we check if  $f_1(\mathbf{c}_1) < f_1(\mathbf{c}_q)$  holds. If not, the order of the control points is simply inverted.

---

**Algorithm 1:** Navigational order for Bézier parameterizations

---

**function:**  $[\mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}}, (\mathbf{o}^{\text{nb}})] = \text{A}^{\text{nb}}(\mathcal{S}_{p,q}, \mathbf{o}^{\text{bez}})$   
**input** : Bézier solution set  $\mathcal{S}_{p,q} = \{\mathbf{x}_1, \dots, \mathbf{x}_p\}$  with intrinsic ordering  $\mathbf{o}^{\text{bez}}$   
**output** : Approximation (sub)set  $\mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}}$ , (navigational order  $\mathbf{o}^{\text{nb}}$ ),  
 $\eta = \arg \min_{i \in \{1, \dots, p\}} f_1(\mathbf{x}_{\mathbf{o}_i^{\text{bez}}});$   
 $\mathbf{o}^{\text{nb}} = [\mathbf{o}_\eta^{\text{bez}}]$  and  $\mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}} = \{\mathbf{x}_{\mathbf{o}_\eta^{\text{bez}}}\};$   
**for**  $j = \eta, \dots, p$  **do**  
    **if**  $\mathbf{x}_{\mathbf{o}_j^{\text{bez}}} \in A(\mathcal{S}_{p,q})$  **and**  $f_2(\mathbf{x}_{\mathbf{o}_j^{\text{bez}}}) < f_2(\mathbf{x}_{\mathbf{o}_{\text{end}}^{\text{nb}}})$  **then**  
         $\mathbf{o}^{\text{nb}} = [\mathbf{o}^{\text{nb}}; \mathbf{o}_j^{\text{bez}}]$  **and**  $\mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}} = \mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}} \cup \{\mathbf{x}_{\mathbf{o}_j^{\text{bez}}}\};$  // here  
         $\mathbf{o}_{\text{end}}^{\text{nb}} = \mathbf{o}_j^{\text{bez}}$

---

### 4.1 A Navigational Order for Bézier Parameterizations

Solution sets  $\mathcal{S}_{p,q} = \mathcal{S}_{p,q}(\phi)$  parameterized by a Bézier curve introduce an intrinsic order  $\mathbf{o}^{\text{bez}}$  of solutions by following the curve from  $t = 0$  to  $t = 1$ . Even though the solutions in  $\mathcal{S}_{p,q}$  now lie on a smooth curve in decision space, it might very well be that some of these solutions dominate others. We define a navigational-Bézier (nb) order  $\mathbf{o}^{\text{nb}}$  for a solution set  $\mathcal{S}_{p,q}$  that follows the order of solutions  $\mathbf{o}^{\text{bez}}$  along the Bézier curve, but also aligns with the left-to-right ordering described in Sect. 3. Pseudo code for  $\mathbf{o}^{\text{nb}}$  is given in Algorithm 1, and an example is given in Fig. 1. The navigational order  $\mathbf{o}^{\text{nb}}$  starts from the solution with best  $f_1$ -value and continues to follow the Bézier curve (i.e., in the order  $\mathbf{o}^{\text{bez}}$ ) until the solution with best  $f_2$ -value is reached, only improving in  $f_2$  (and thereby worsening in  $f_1$ ) along the way, and skipping solutions that violate this



property. Let  $\mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}} = A^{\text{nb}}(\mathcal{S}_{p,q}, \mathbf{o}^{\text{bez}})$  be the resulting subset of  $\mathcal{S}_{p,q}$  pertaining to exactly the solution indices as specified in  $\mathbf{o}^{\text{nb}}$ , and note that this is an approximation set.

## 4.2 Unfolding the Bézier Curve (in Objective Space)

Smoothly navigable approximation sets can now be obtained by maximizing the hypervolume of  $\mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}}$ . To maximize the number of navigable solutions  $|\mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}}| = |\mathbf{o}^{\text{nb}}|$ , we need to unfold the Bézier curve in objective space. For this, we introduce a constraint violation function  $C(\mathcal{S}_{p,q}, \mathbf{o}^{\text{nb}}) \geq 0$ , as given in Algorithm 2 and illustrated in Fig. 1. It is composed of two parts. The first part is similar to the uncrowded distance term in Eq. (1), but the approximation boundary is now given by  $\mathcal{A}_{p,q,\mathbf{o}^{\text{nb}}}$ . The second part aims to pull solutions that are not in  $\mathcal{S}_{p,q,\mathbf{o}^{\text{nb}}}$  towards neighboring solutions on the Bézier curve.

---

### Algorithm 2: Bézier constraint violation function

---

**function:**  $C(\mathcal{S}_{p,q}, \mathbf{o}^{\text{bez}}) \geq 0$   
**input** : Bézier solution set  $\mathcal{S}_{p,q} = \{\mathbf{x}_1, \dots, \mathbf{x}_p\}$  with intrinsic ordering  $\mathbf{o}^{\text{bez}}$   
**output** : Constraint value  $C \geq 0$

$[\mathcal{A}, \mathbf{o}^{\text{nb}}] = A^{\text{nb}}(\mathcal{S}_{p,q}, \mathbf{o}^{\text{bez}});$  // See Algorithm 1  
 $C = \frac{1}{|\mathcal{S}_{p,q}|} \sum_{\mathbf{x} \in \mathcal{S}_{p,q}} \text{ud}_{\mathbf{f}}(\mathbf{x}, \mathcal{A}^{\text{nb}});$  // Uncrowded distance (ud), see (1)

**for**  $j = 1, \dots, |\mathcal{S}_{p,q}| - 1$  **do**  
    **if**  $\mathbf{o}_j^{\text{bez}} \notin \mathbf{o}^{\text{nb}}$  **or**  $\mathbf{o}_{j+1}^{\text{bez}} \notin \mathbf{o}^{\text{nb}}$  **then**  
         $C = C + \|\mathbf{f}(\mathbf{x}_{\mathbf{o}_j^{\text{bez}}}) - \mathbf{f}(\mathbf{x}_{\mathbf{o}_{j+1}^{\text{bez}}})\|;$  // Euclidean distance in  $\mathbb{R}^m$

---

## 4.3 Bézier Parameterization + GOMEA = BezEA

The resulting Bézier curve optimization problem is given by,

$$\begin{aligned} & \text{maximize} && \text{HV}_{\mathbf{f}, \mathcal{S}_{p,q}}(\phi) = \text{HV}_{\mathbf{f}}(A^{\text{nb}}(\mathcal{S}_{p,q}(\phi))), \\ & \text{with} && C(\mathcal{S}_{p,q}(\phi), \mathbf{o}^{\text{nb}}(\phi)) = 0, \\ & && \mathbf{f} : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad \mathcal{S}_{p,q} : \mathbb{R}^{q \cdot n} \rightarrow \mathcal{P}(\mathcal{X}), \quad \phi \in \mathbb{R}^{q \cdot n}. \end{aligned} \quad (5)$$

We use *constraint domination* to handle constraint violations [7]. With constraint domination, the fitness of a solution is computed regardless of its feasibility. When comparing two solutions, if both are infeasible (i.e.,  $C > 0$ ), the solution with the smallest amount of constraint violation is preferred. If only one solution is infeasible, the solution that is feasible is preferred. Finally, if both solutions are feasible (i.e.,  $C = 0$ ), the original ranking based on fitness is used.

Bézier curves have no local control property, meaning that a change of a control point affects all solutions on the curve. Partial evaluations can therefore

no longer be exploited with this parameterization, and we thus solve this problem with the black-box version of GOMEA. Analogous to the UHV naming, we brand the resulting algorithm Bézier-GOMEA-bb, which we abbreviate to BezEA. A detailed description of GOMEA can be found in [5], and a description of UHV-GOMEA in [18].

## 5 Numerical Experiments

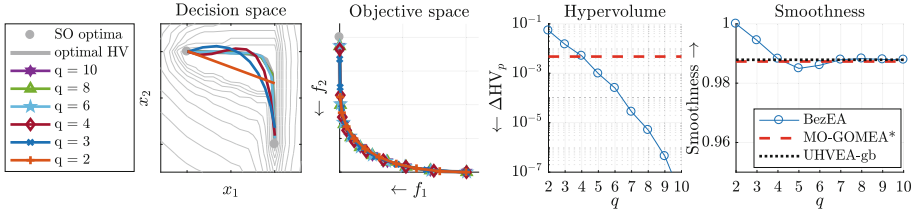
We compare BezEA with UHVEA-gb and UHVEA-bb. These methods use a different hypervolume-based representation of the multi-objective problem, but use very similar variation and selection mechanisms, making the comparison between these methods most fair. We use the guideline setting for the population size  $N$  of GOMEA with full linkage models in a black-box setting [4], which for separable problems yields  $N = \lfloor 10\sqrt{l} \rfloor$  and for non-separable problems  $N = 17 + \lfloor 3l^{1.5} \rfloor$ . BezEA solves a single-objective problem of  $l = qn$  decision variables. UHVEA-bb solves a single objective problem of  $l = pn$  decision variables. UHVEA-gb solves the same problem by not considering all  $pn$  decision variables simultaneously, but by updating only subsets of  $l = n$  decision variables, on which we base the population size guideline for UHVEA-gb.

We furthermore include the domination-based MO-GOMEA [6]. In MO-GOMEA, a population of solutions is aimed to approximate the Pareto front by implicitly balancing diversity and proximity. From a population of  $N_{mo}$  solutions, truncation selection is performed based on domination rank. The resulting selection is clustered into  $K_{mo}$  overlapping clusters that model different parts of the approximation front. For each cluster, a Gaussian distribution is estimated to sample new solutions from, which uses very similar update rules as the single-objective GOMEA, and therefore allows for a most fair comparison to BezEA and UHVEA. MO-GOMEA obtains an elitist archive, aimed to contain 1000 solutions. For a fair comparison to the hypervolume-based methods that obtain an approximation set of at most  $p$  solutions, we reduce the obtained elitist archive of MO-GOMEA to  $p$  solutions using greedy hypervolume subset selection (gHSS) [11], which we denote by MO-GOMEA\*. As described in [18], to align MO-GOMEA with the other algorithms, we set  $N_{mo} = p \cdot N$  and  $K_{mo} = 2p$  such that the overall number of solutions in the populations is the same, and all sample distributions are estimated from the same number of solutions.

As performance measure, we define  $\Delta HV_p = HV_p^* - HV(\mathcal{A}_p)$  as the distance to the optimal hypervolume  $HV_p^*$  obtainable with  $p$  solutions, empirically determined with UHVEA.

### 5.1 Increasing $q$

We illustrate how increasing the number of control points  $q$  of the Bézier curve improves achievable accuracy of BezEA (with  $q = \{2, \dots, 10\}$  and  $p = 10$ ) in case the Pareto set is non-linear. For this, we construct a simple two-dimensional problem *curvePS*, with objective functions  $f_1^{\text{curvePS}}(\mathbf{x}) = (x_1 - 1)^2 + 0.01x_2^2$  and



**Fig. 3.** Bézier curve approximations of the Pareto set of the curvePS problem (left), obtained with BezEA. Contour lines show domination ranks, the corresponding approximation fronts (middle), and  $\Delta\text{HV}_{10}$  together with smoothness (right).

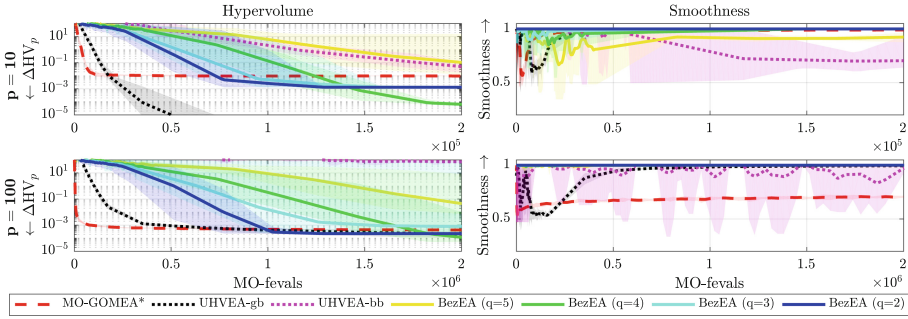
$f_2^{\text{curvePS}}(\mathbf{x}) = x_1^2 + (x_2 - 1)^2$ . A large computational budget was used to show maximally achievable hypervolume, and standard deviations are therefore too small to be visible.

Results are shown in Fig. 3. A larger  $q$  results in a better approximation of the leftmost endpoint of the Pareto front (second subfigure), thereby improving  $\Delta\text{HV}_p$  (third subfigure), but slightly lowering smoothness (fourth subfigure), as the Bézier curve deviates from a straight line. MO-GOMEA\*, UHVEA-gb, and BezEA for large  $q$  all obtain a very similar smoothness. As MO-GOMEA\* does not explicitly optimize the hypervolume of its approximation set, it obtains a slightly different distribution of solutions, which results in a lower hypervolume. Additionally, MO-GOMEA\* does not converge to the Pareto set due to the finite population size and infinitely large Pareto set, as described in more detail in [18]. Even though this is a fundamental limitation of domination-based MOEAs, this level of accuracy is often acceptable in practice.

## 5.2 Comparison with UHV Optimization

Next, we demonstrate the behavior of BezEA compared to UHVEA on the simple *bi-sphere* problem, which is composed of two single-objective sphere problems,  $f_{\text{sphere}}(\mathbf{x}) = \sum_{i=1}^n x_i^2$ , of which one is translated,  $f_1^{\text{bi-sphere}}(\mathbf{x}) = f_{\text{sphere}}(\mathbf{x})$ , and  $f_2^{\text{bi-sphere}} = f_{\text{sphere}}(\mathbf{x} - \mathbf{e}_1)$ , where  $\mathbf{e}_i$  is the  $i^{\text{th}}$  unit vector. We set  $n = 10$ , and initialize all algorithms in  $[-5, 5]^n$ . This is a separable problem and we therefore use the univariate population size guideline (i.e.,  $N = \sqrt{l}$ ). We consider the cases  $p = \{10, 100\}$ . The computational budget is set to  $2p \cdot 10^4$  evaluations of the multi-objective problem given by  $\mathbf{f}$  (MO-fevals). When the desired number of solutions  $p$  along the front is large, neighboring solutions are nearby each other on the approximation front. This introduces a dependency between these solutions, which needs to be taken into account in the optimization process to be able to effectively solve the problem [18].

Results are shown in Fig. 4. This problem is unimodal with a linear Pareto set, and the smoothness of (a subset) of the Pareto set is therefore 1.0. As UHVEA-gb converges to a subset of the Pareto set (see [18]), it ultimately obtains a smoothness of 1.0, even though its smoothness is initially lower. MO-GOMEA\*



**Fig. 4.** Comparison of UHVEA with BezEA and MO-GOMEA\* on the bi-sphere problem with  $n = 10$  and  $p = 10$  (top row) and  $p = 100$  (bottom row). Left two subfigures show mean scores, and the shaded areas represent min/max scores, obtained over 10 runs. Objective and decision space subfigures show results of a single run. Solutions in the decision space projection are sorted based on their  $f_0$ -value, from best to worst.

does not converge to the Pareto set, and its smoothness stagnates close to 1.0 when  $p = 10$ , but stagnates around 0.7 when  $p = 100$ . BezEA with  $q = 2$  has per construction a perfect smoothness of 1.0, and for  $q = 3$  and  $q = 4$ , the obtained smoothness is close to 1. With  $q = 5$  control points, BezEA does not converge within the given budget, resulting in a lower smoothness within the computational budget. UHVEA-gb furthermore shows a better convergence rate, which could be because UHVEA-gb can exploit partial evaluations, while this is not possible with BezEA. However, UHVEA-bb, which also does not perform partial evaluations, is unable to solve the problem for  $p = 100$ . This difference between BezEA and UHVEA-bb could be attributed to the lower degree of freedom that BezEA has due to the rather fixed distribution of solutions. This distribution does however not exactly correspond to the distribution of  $HV_p^*$ . This is why a stagnation in terms of hypervolume convergence can be observed for small values of  $q$ . The solutions of BezEA are equidistantly distributed along the curve in terms of  $t$ . By doing so, intermediate control points can be used to adapt the distribution of solutions (when  $q > 2$ ). This is why BezEA with  $q = 4$  can obtain a better  $\Delta HV_p$  than BezEA with  $q = 2$ , even though the Pareto set is linear. For  $p = 100$ , BezEA obtains a better  $\Delta HV_p$  than UHVEA-gb, which can be explained by the increased problem complexity when the desired number of solutions along the front is large. Increasing the population size  $N$  of UHVEA-gb would (at least partially) overcome this, but we aimed here to show that BezEA does not suffer from this increased complexity as its problem dimensionality depends on  $q$ , not  $p$ .

### 5.3 WFG Benchmark

We benchmark BezEA, UHVEA, and MO-GOMEA on the nine commonly used WFG functions [12]. We consider bi-objective WFG problems with  $n = 24$  decision variables of which  $k_{WFG} = 4$  are WFG-position variables. We furthermore

**Table 1.** Obtained hypervolume  $HV_p$  (mean  $\pm$  standard deviation (rank)) and mean navigational smoothness (Sm) for the 9 WFG problems with  $p = 9$  solutions. Bold are best scores per problems, or those not statistically different from it.

#	MO-GOMEA*		UHVEA-gb		BezEA ( $q = 2$ )		BezEA ( $q = 3$ )	
	$HV_9$	Sm	$HV_9$	Sm	$HV_9$	Sm	$HV_9$	Sm
1	<b>97.60</b> $\pm$ 0.7 (1)	0.76	93.62 $\pm$ 1.7 (2)	0.67	90.35 $\pm$ 1.1 (4)	1.00	90.37 $\pm$ 1.2 (3)	0.99
2	110.09 $\pm$ 0.0 (2)	0.86	<b>110.38</b> $\pm$ 1.0 (1)	0.66	97.74 $\pm$ 0.0 (4)	1.00	97.85 $\pm$ 0.0 (3)	0.98
3	116.11 $\pm$ 0.1 (4)	0.93	116.42 $\pm$ 0.1 (3)	0.71	<b>116.50</b> $\pm$ 0.0 (1)	1.00	<b>116.50</b> $\pm$ 0.0 (2)	1.00
4	111.88 $\pm$ 0.8 (3)	0.75	<b>112.37</b> $\pm$ 0.7 (1)	0.69	111.59 $\pm$ 1.3 (4)	1.00	<b>112.19</b> $\pm$ 1.3 (2)	0.98
5	112.03 $\pm$ 0.1 (3)	0.66	111.86 $\pm$ 0.3 (4)	0.63	112.17 $\pm$ 0.0 (2)	1.00	<b>112.19</b> $\pm$ 0.0 (1)	1.00
6	113.86 $\pm$ 0.3 (3)	0.88	114.23 $\pm$ 0.2 (2)	0.72	<b>114.34</b> $\pm$ 0.1 (1)	1.00	113.02 $\pm$ 0.3 (4)	0.99
7	114.06 $\pm$ 0.1 (4)	0.94	114.32 $\pm$ 0.1 (3)	0.66	114.37 $\pm$ 0.0 (2)	1.00	<b>114.38</b> $\pm$ 0.0 (1)	1.00
8	110.70 $\pm$ 0.2 (4)	0.79	<b>111.24</b> $\pm$ 0.3 (1)	0.67	111.07 $\pm$ 0.1 (3)	1.00	111.14 $\pm$ 0.0 (2)	1.00
9	<b>111.70</b> $\pm$ 0.5 (1)	0.68	111.46 $\pm$ 0.1 (2)	0.68	110.19 $\pm$ 0.7 (3)	1.00	109.36 $\pm$ 2.9 (4)	0.98

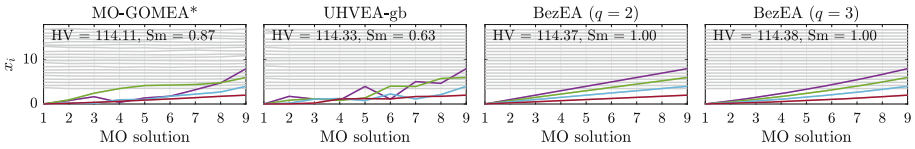
set  $p = 9$  and a computational budget of  $10^7$  MO-fevals. A population size of  $N = 200$  was shown to work well for UHVEA [18], which we use here also for BezEA. We perform 30 runs, and a pair-wise Wilcoxon rank-sum test with  $\alpha = 0.05$  is used to test whether differences with the best obtained result are statistically significant (up to 4 decimals). Ranks (in brackets) are computed based on the mean hypervolume values.

Results are given in Table 1. WFG1 is problematic, as none of the algorithms have an explicit mechanism to deal with its flat region. WFG2 has a disconnected Pareto front. MO-GOMEA\* and UHVEA-gb both obtain solutions in multiple subsets, while BezEA obtains all solutions in a single connected subset, and spreads out well there. The linear front of WFG3 corresponds to the equidistant distribution of solutions along the Bézier curve, and BezEA outperforms the other methods there. Increasing  $q$  generally increases performance of BezEA, except for WFG6 and WFG9. Both these problems are non-separable, and require a larger population size than the currently used  $N = 200$  to be properly solved. However, the guideline for non-separable problems results in a population size that is too large to be of practical relevance here. In terms of smoothness, BezEA with  $q = 3$  is able to obtain a smoothness close to 1, while simultaneously obtaining the best  $HV_9$  for 4/9 problems. MO-GOMEA\* obtains a mean smoothness of 0.81 while UHVEA-gb obtains the worst mean smoothness (0.68). To illustrate the obtained smoothness a parallel coordinate plot for WFG7 is given in Fig. 5. This figure shows a clear pattern in decision variable values along the front (in the order  $\mathbf{o}$ ) for BezEA. This pattern is not obvious for the other two methods, while they achieve only a slightly lower hypervolume, and a lower smoothness.

## 6 Discussion and Outlook

In this work, we parameterized approximation sets as smooth Bézier curves in decision space, thereby explicitly enforcing a form of smoothness between

decision variables of neighboring solutions when the approximation front is traversed, aimed to improve its navigability. We used an UHV-based MO problem formulation that directly allows for the optimization of parameterized approximation sets. Solving this Bézier problem formulation with GOMEA (BezEA), was shown to be competitive to UHV-based optimization and domination-based MOEAs, while smoothness is guaranteed. We showed that approximation sets obtained with BezEA show a more clear pattern in terms of decision variables when traversing the approximation front on a set of benchmark problems, which suggests that this approach will lead to a more intuitive and smooth approximation set navigability for real-world optimization problems.



**Fig. 5.** Parallel coordinate plots shows of decision variables  $x_i$  for WFG7. In color the  $k_{WFG} = 4$  position-type decision variables, in grey the remaining decision variables.

We chose to fix the solution set size  $p$  for BezEA during and after optimization, but since a parametric expression of the approximation set is available, it is straightforward to construct a large approximation set after optimization. This could be exploited to increase performance of BezEA, as it currently show computational overhead on the simple bi-sphere problem in terms of multi-objective function evaluations compared to UHVEA. In contrast to MOEAs, UHVEA and BezEA have the ability to converge to the Pareto set. When the problem is multimodal, UHVEA will spread its search over multiple modes. In that case, even an a posteriori fitting of a smooth curve through the obtained approximation set will result in low-quality solutions. BezEA on the other hand aims to obtain solutions in a single mode, thereby guaranteeing smoothness, even in a multimodal landscape. This form of regularization that is enforced upon approximation sets shows that BezEA can outperform MO-GOMEA\* and UHVEA-gb on multiple problems in the WFG benchmark.

The smoothness measure introduced in this work is a measure for entire solution sets  $\mathcal{S}_p$ , and not for individual solutions  $\mathbf{x}$ . It can therefore not be added directly as an additional objective to the original multi-objective problem  $\mathbf{f}(\mathbf{x})$ . We chose in this work to introduce a parameterization of approximation sets that directly enforces smoothness. Alternatively, smoothness could also be added as a second objective to the UHV-based problem formulation. This then results in the  $pn$ -dimensional bi-objective optimization problem, given by  $h(\mathcal{S}_p) = [\text{UHV}_{\mathbf{f}}(\mathcal{S}_p) ; \text{Sm}(\mathcal{S}_p)]$ . This problem can then be solved with a domination-based MOEA, or even by again formulating it as a (much) higher-dimensional UHV-based single-objective problem. Whether this approach can be

efficient, even when grey-box properties such as partial evaluations are exploited, remains however future work.

The problems in this work were limited to problems involving two objectives. The presented results show that it is an interesting research avenue to extend this work to problems with more objectives. The Pareto front of non-degenerate problems with  $m$  objectives is an  $m-1$ -dimensional manifold. Instead of a one-dimensional Bézier curve, the Pareto set can then be modeled by an  $(m-1)$ -dimensional Bézier simplex [15]. For the navigation of higher-dimensional manifolds, a one-dimensional path through all obtained solutions could still be used. However, navigation would be performed might be problem specific and should be discussed with end-users. BezEA is applied to treatment planning of brachytherapy for prostate cancer, and results can be found in the supplementary of this work (online [17]).

*Source code for the algorithms in this work is made available at [16].*

**Acknowledgments.** This work was supported by the Dutch Research Council (NWO) through Gravitation Programme Networks 024.002.003. We furthermore acknowledge financial support of the Nijbakker-Morra Foundation for a high-performance computing system.

## References

1. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proceedings of the IEEE Congress on Evolutionary Computation - CEC 2005, pp. 1769–1776. IEEE Press (2005)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
3. Bhardwaj, P., Dasgupta, B., Deb, K.: Modelling the Pareto-optimal set using B-spline basis functions for continuous multi-objective optimization problems. *Eng. Optim.* **46**(7), 912–938 (2014)
4. Bosman, P.A.N., Grahl, J., Thierens, D.: Benchmarking parameter-free AMaLGaM on functions with and without noise. *Evol. Comput.* **21**(3), 445–469 (2013)
5. Bouter, A., Alderliesten, T., Witteveen, C., Bosman, P.A.N.: 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 - GECCO 2017, pp. 705–712. ACM Press, New York (2017)
6. Bouter, A., Luong, N.H., Alderliesten, T., Witteveen, C., Bosman, P.A.N.: The multi-objective real-valued gene-pool optimal mixing evolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 2017, pp. 537–544. ACM Press, New York (2017)
7. Deb, K.: An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **186**(2), 311–338 (2000)
8. Deb, K.: *Multi-objective Optimization*. Wiley, Chichester (2001)
9. 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)



10. Gallier, J.: *Curves and Surfaces in Geometric Modeling: Theory and Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco (1999)
11. Guerreiro, A., Fonseca, C., Paquete, L.: Greedy hypervolume subset selection in low dimensions. *Evol. Comput.* **24**(3), 521–544 (2016)
12. 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)
13. Knowles, J.: Local-search and hybrid evolutionary algorithms for Pareto optimization. Technical report, Ph.D. thesis, University of Reading (2002)
14. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimization. Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich - TIK report 214 (2006)
15. Kobayashi, K., Hamada, N., Sannai, A., Tanaka, A., Bannai, K., Sugiyama, M.: Bezier simplex fitting: describing Pareto fronts of simplicial problems with small samples in multi-objective optimization. Preprint [arXiv:1812.05222](https://arxiv.org/abs/1812.05222) (2018)
16. Maree, S.C.: Uncrowded-hypervolume multi-objective optimization C++ source code on Github (2019). <https://github.com/scmaree/uncrowded-hypervolume>
17. Maree, S.C., Alderliesten, T., Bosman, P.A.N.: Ensuring smoothly navigable approximation sets by Bézier curve parameterizations in evolutionary bi-objective optimization - applied to brachytherapy treatment planning for prostate cancer. Preprint [arXiv:2006.06449](https://arxiv.org/abs/2006.06449) (2020)
18. Maree, S.C., Alderliesten, T., Bosman, P.A.N.: Uncrowded hypervolume-based multi-objective optimization with gene-pool optimal mixing. Preprint [arXiv:2004.05068](https://arxiv.org/abs/2004.05068) (2020)
19. Mehta, V.K., Dasgupta, B.: Parametric approximation of the Pareto set in multi-objective optimization problems. *J. Multi-Crit. Decis. Anal.* **21**, 335–362 (2014)
20. Touré, C., Hansen, N., Auger, A., Brockhoff, D.: Uncrowded hypervolume improvement: COMO-CMA-ES and the sofomore framework. In: *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 2019*, pp. 638–646. ACM Press, New York (2019)
21. Wang, H., Deutz, A., Bäck, T., Emmerich, M.: Hypervolume indicator gradient ascent multi-objective optimization. In: Trautmann, H., et al. (eds.) *EMO 2017*. LNCS, vol. 10173, pp. 654–669. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-54157-0\\_44](https://doi.org/10.1007/978-3-319-54157-0_44)
22. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems - EUROGEN 2001*, pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE) (2001)
23. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* **3**(4), 257–271 (1999)
24. 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)





# Many-Objective Test Database Generation for SQL

Zhilei Ren<sup>1</sup>(✉), Shaozheng Dong<sup>1</sup>, Xiaochen Li<sup>2</sup>, Zongzheng Chi<sup>1</sup>, and He Jiang<sup>1</sup>

<sup>1</sup> School of Software, Dalian University of Technology, Dalian, China  
{zren, jianghe}@dlut.edu.cn, dsz201493078@mail.dlut.edu.cn,  
czz.dut@163.com

<sup>2</sup> University of Luxembourg, Luxembourg City, Luxembourg  
xiaochen.li@uni.lu

**Abstract.** Generating test database for SQL queries is an important but challenging task in software engineering. Existing approaches have modeled the task as a single-objective optimization problem. However, due to the improper handling of the relationship between different targets, the existing approaches face strong limitations, which we summarize as the inter-objective barrier and the test database bloating barrier. In this study, we propose a two-stage approach **MoeSQL**, which features the combination of many-objective evolutionary algorithm and decomposition based test database reduction. The effectiveness of **MoeSQL** lie in the ability to handle multiple targets simultaneously, and a local search to avoid the test database from bloating. Experiments over 1888 SQL queries demonstrate that, **MoeSQL** is able to achieve high coverage comparable to the state-of-the-art algorithm **EvoSQL**, and obtain more compact solutions, only 59.47% of those obtained by **EvoSQL**, measured by the overall number of data rows.

**Keywords:** Test database generation · Search based software engineering · Many-objective optimization

## 1 Introduction

Recent years have witnessed the emergence and the rapid development of evolutionary computation based test case generation research [1, 2]. Especially, due to the importance in database-centric applications, test database generation for SQL queries has gained great research interest [3, 4]. The idea is to construct test databases, in pursuit of certain coverage criteria, such as to exercise all branches (also known as targets, see Sect. 2 for details) that can be executed in the SQL query. Due to the intrinsic complexity of SQL features, e.g., JOINS, predicates, and subqueries, test database generation for SQL queries can be difficult and time-consuming.

In the existing studies, this problem has been modeled as an optimization problem. Various approaches such as constraint solving and genetic algorithm have been employed to solve the problem [3–5]. Among these approaches, **EvoSQL** [3], a search-based algorithm, achieves the state-of-the-art results. **EvoSQL** features the support for the SQL standard, and has been evaluated over a set of real-world SQL queries.

However, despite the promising results accomplished, we could observe significant limitations in the existing studies. For example, **EvoSQL** models the test database generation problem as a single-objective problem, by designing an objective function that aggregates the coverage over all the branches. Consequently, such problem solving mechanism may face great challenges, which are summarized as follows.

- (1) Inter-objective relationship barrier: taking **EvoSQL** as an example, to achieve satisfactory coverage, the underlying genetic algorithm has to be executed for multiple times, to cover each branch in a sequential way. Hence, a solution from one pass of evolution could not take all the branches into account. Also, the solutions within one evolution process could not help improve the other independent runs of evolution [6].
- (2) Test database bloating barrier: **EvoSQL** achieves the branch coverage by merging the test databases obtained by the multiple executions of the genetic algorithm. The final test database may suffer from scalability issues [7], due to the improper handling of the relationship between different targets. Although **EvoSQL** adopts a post-process for reduction, chances are that the reduced test databases are still of large size.

To overcome these challenges, we propose a two-stage algorithm **MoeSQL** (Many-objective evolutionary algorithm for **SQL**) in search of better test data. More specifically, to tackle the inter-objective relationship barrier, in the first stage, we adopt a many-objective evolutionary algorithm to avoid redundant computation. The many-objective algorithm features a corner solution based sorting mechanism, with which we are able to cover multiple targets in a single evolution process.

To tackle the test database bloating barrier, we further leverage the solutions obtained from the first stage. We decompose the original problem into a series of sub-problems, and employ a local search operator to achieve better solutions. Due to the reduction of the search space, it is easier to obtain more compact test database.

By combining the two stages, we develop an integrated framework **MoeSQL**. To evaluate **MoeSQL**, we consider real-world datasets for experiments, with 1888 SQL queries [3]. Extensive experiments demonstrate that with the many-objective evolutionary algorithm, **MoeSQL** is able to obtain high target coverage of 99.80%, which is comparable to the state-of-the-art approach **EvoSQL**. Meanwhile, with the reduction stage, **MoeSQL** obtains much more compact test databases, only 59.47% of those provided by **EvoSQL**, measured by the overall number of data rows for all the instances.

The main contributions of this paper are as follows:

- (1) A many-objective search method is proposed for test database generation of SQL queries. To the best of our knowledge, this is the first study that solves this problem with a many-objective approach.
- (2) We propose a novel decomposition based local search algorithm to address the test database bloating issue in SQL test database generation.
- (3) We implement a prototype of **MoeSQL**. The prototype system and the experiment data are available at <https://github.com/TheSecondLoop/MoeSQL>.
- (4) We conduct extensive experiments to demonstrate the effectiveness of **MoeSQL** compared with the state-of-the-art algorithm.

The rest of the paper is organized as follows. Section 2 describes the background of test database generation for SQL queries with a motivating example. Section 3 introduces

the proposed approach. The empirical study is presented in Sect. 4. Finally, the conclusion and future work are given in Sect. 5.

## 2 Background and Motivating Example

### 2.1 Coverage Criteria

For the test database generation task, we intend to populate a set of databases based on certain coverage criteria. Considering the following SQL query  $S$  as an example:

```
SELECT * FROM
  Ta JOIN Tb ON Ta.p = Tb.q      -- step 1
WHERE (Ta.a = 1) OR (Ta.b = 2); -- step 2
```

In the query  $S$ , both columns **a** and **b** are non-nullable. To thoroughly test  $S$ , we adopt the SQL full predicate coverage criteria [8], which is inspired by the modified condition decision coverage [9] in software testing studies. The underlying idea is that given a SQL query, all the possible conditions which contribute to the query should be tested. For example, if we combine the modified conditions of the predicates in the **WHERE** clause of  $S$  with two predicates, we obtain six queries, generated by the SQL analysis tool SQLFpc [8]. More specifically, the predicates “ $Ta.a = 1$ ” and “ $Ta.b = 2$ ” correspond to targets 1–3 and 4–6, respectively:

```
(1) SELECT * FROM Ta JOIN Tb ON Ta.p = Tb.q WHERE (Ta.a = 0) AND NOT (Tb.b = 2);
(2) SELECT * FROM Ta JOIN Tb ON Ta.p = Tb.q WHERE (Ta.a = 1) AND NOT (Ta.b = 2);
(3) SELECT * FROM Ta JOIN Tb ON Ta.p = Tb.q WHERE (Ta.a = 2) AND NOT (Ta.b = 2);
(4) SELECT * FROM Ta JOIN Tb ON Ta.p = Tb.q WHERE NOT (Ta.a = 1) AND (Ta.b = 1);
(5) SELECT * FROM Ta JOIN Tb ON Ta.p = Tb.q WHERE NOT (Ta.a = 1) AND (Ta.b = 2);
(6) SELECT * FROM Ta JOIN Tb ON Ta.p = Tb.q WHERE NOT (Ta.a = 1) AND (Ta.b = 3);
```

With these targets, the next goal is to construct a set of test databases, so that each of the six queries, when applied on the test databases, retrieves non-empty result. If such goal is accomplished, it is claimed that the test databases have achieved complete coverage on the SQL query under test.

### 2.2 Test Database Generation

In this study, we focus on search-based test database generation. In these approaches, a common technique is to encode the test databases as candidate solutions, and model the objective function based on certain coverage criteria. For example, **EvoSQL** uses the concept of physical query plan [10] to divide each target into several execution steps. The objective function of the test database is determined according to its performance on each execution step. More specifically, the search problem is defined as follows:

**Problem 2.1** (single-objective model). Let  $R = \{r_1, \dots, r_k\}$  be the set of coverage targets of the SQL query under test. Find a set of test databases  $D = \{t_1, \dots, t_k\}$  to cover all the coverage targets in  $R$ , i.e., one that minimizes the following objective function:

$$\min F(D, R) = \sum_{i=1}^k \text{step\_level}(t_i, r_i) + \text{step\_distance}(t_i, L), \quad (1)$$

where  $step\_level(t_i, r_i)$  denotes the number of steps that are not executed, and  $step\_distance(t_i, L)$  is the distance of  $t_i$  in satisfying the first unsatisfied step  $L$ .

To explain the objective function, consider the distance of target 2 (**SELECT \* FROM Ta JOIN Tb ON Ta.p = Tb.q WHERE (Ta.a = 1) AND NOT (Ta.b = 2)**) and **db 1** in Fig. 1(a). In the physical query plan of  $S$ , target 2 can be divided into two steps: the first step considers the predicate in the **FROM** clause, and then the predicate in the **WHERE** clause (see the comments in  $S$ ). The predicate in the **FROM** clause could be satisfied by **db 1**. In **db 1**, an empty result is returned when the predicate in the **WHERE** clause is examined. Hence, there are no unexecuted steps, i.e.,  $step\_level(t_i, r_i) = 0$ . Meanwhile, in **db 1**, the predicate “Ta.a = 1” in the **WHERE** clause is not satisfied. According to the predicate, we choose the closest value 0 in column **a** of **db 1**. Then, the step distance is calculated as  $step\_distance(t_i, L) = |0 - 1| = 1$  [11]. In this way, we can calculate the distance between the test database and the coverage target. Further details about the objective function evaluation could be found in reference [3].

	Ta			Tb
	Ta.p	Ta.a	Ta.b	Tb.q
db 1	1	0	1	1
db 2	1	1	1	1
db 3	1	2	1	1
db 4	1	0	1	1
db 5	1	0	2	1
db 6	1	0	3	1

	Ta			Tb
	Ta.p	Ta.a	Ta.b	Tb.q
db 7	1	0	1	1
	1	1	1	
	1	2	2	
	1	2	3	

(a) Test databases obtained by **EvoSQL**

(b) A more compact test database

**Fig. 1.** Example of solutions for query  $S$

Obviously, the objective function is essentially an aggregate form of a multi-objective problem. Typically, existing approaches such as **EvoSQL** optimize each term of the summation in Eq. 1 with respect to each target, in a sequential way. The number of test databases equals to the number of coverage targets. For example, for query  $S$ , **EvoSQL** executes the underlying genetic algorithm six times, and generates six test databases, each with one row for **Ta** and **Tb**, respectively. However, the single-objective model may face obvious challenges:

- (1) Inter-objective relationship barrier: In the SQL query  $S$ , targets 1–3 share the same predicate “Ta.b = 2”. During the evolution towards target 1, the solutions obtained during the search procedure may also partially satisfy some predicates of targets 2–3. Although **EvoSQL** uses the population of the previous pass of evolution as the initial population for the next pass, the performance of this approach may depend on the invocation sequence of the underlying genetic algorithm. Consequently, single-objective approach cannot deal with the inter-objective relationship properly.
- (2) Test database bloating barrier: In Fig. 1(b), we present a more compact solution (**db 7** with five data rows) that satisfies all the targets of the query  $S$ . Compared with the results of **EvoSQL**, **db 7** has the same coverage but less data rows. Interestingly, although **EvoSQL** is equipped with a reduction operator, the results in Fig. 1(a) could not be further simplified.

### 3 Our Approach

In order to tackle the two challenges of the existing algorithms, we propose our two-stage algorithm **MoeSQL**. In the first stage, the algorithm takes the coverage target generated by SQLFpc as the input, and obtains multiple databases to cover different targets. These databases serve as an intermediate solution to the problem. In the second stage, we use these solutions to divide the problem into sub-problems, and solve the induced problems to achieve a more compact solution.

#### 3.1 Many-Objective Test Database Generation

To generate test database with many-objective algorithms, we first modify the problem definition in Sect. 2 as follows.

**Problem 3.1** (many-objective model). Let  $R = \{r_1, \dots, r_k\}$  be the set of coverage targets of the SQL query under test. Find a test database  $t$  to cover as many coverage targets in  $R$  as possible, and keep the test database compact, i.e., minimize the following  $k + 1$  objectives:

$$\min F'(t, R) = (d(t, r_1), d(t, r_2), \dots, d(t, r_k), size(t))^T, \quad (2)$$

where  $d(t, r_i) = step\_level(t, r_i) + step\_distance(t, L)$  denotes the distance between the test database  $t$  and the coverage targets  $r_i$  as in Eq. 1. The extra objective  $size(t)$  represents the number of data rows in the test database  $t$ . The superscript  $T$  represents transpose of vector.

The pseudo code of **TestDatabaseGen** is presented in Algorithm 1. The workflow is similar with most existing many-objective algorithms. In Lines 1–3, a set of solutions are initialized. More specifically, each solution is encoded as a set of tables, each of which corresponds to a schema involved in the targets. We extract the constant values in the targets, and assign the constant values to the fields in initial solutions with probability  $p_s$ . Otherwise, the value for the field is initialized by a random value with probability  $(1 - p_s)$  [12].

Then, in the main loop (Lines 4–15), the evolution process consists of the evaluation, sorting, selection, and reproduction procedures. For the evaluation procedure, we apply Eq. 2 over each solution, to calculate the objective values. In particular, we adopt a dynamic objective strategy [13], i.e., if there exists any new target that can be covered by a solution, we keep the solution and remove the target from the objective evaluation. With this strategy, we are able to deal with a relatively large number of objectives. For the sorting and the selection procedures, we consider the many-objective sorting mechanism used in MOSA [6, 14], a well-known many-objective algorithm in the search-based software engineering community. The sorting mechanism features the multi-level prioritization of the solutions. Within the sorting procedure, the population is categorized into levels. For the first level, we consider the best solutions (corner solutions) with respect to each objective. Then, the next level comprises the non-dominated solutions for the rest solutions. This process continues, until all the solutions are iterated. With this mechanism, the search could be guided towards covering more targets. During the

selection, the elitism strategy is considered, i.e., only when one level is selected, we consider the solutions in the next level. In the same level, the tournament selection [15] is applied, so that both intensification and diversification are considered.

**Algorithm 1: TestDatabaseGen**

Input: coverage set  $R$ , population size  $pop\_num$ , seeding probability  $p_s$ ,  
crossover probability  $p_c$ , mutation probability  $p_m$

Output: a set of test database  $D$

```

1   $seed \leftarrow seed\_extract(R)$ 
2   $archive \leftarrow \{\}$ 
3   $population \leftarrow initialization(pop\_num, seed, p_s)$ 
4  while stopping criterion not met
5     $evaluation(population, R)$ 
6    if there exists  $r_i$  covered by  $population_j$ 
7       $archive \leftarrow archive \cup population_j$ 
8       $R \leftarrow R \setminus \{r_i\}$ 
9    end if
10  $population \leftarrow many\_objective\_sort(population)$ 
11  $population \leftarrow selection(population, pop\_num)$ 
12  $offspring \leftarrow crossover(population, p_c)$ 
13  $offspring \leftarrow mutation(offspring, p_m, seed)$ 
14  $population \leftarrow population \cup offspring$ 
15 end while
20 return  $archive$ 

```

As for the reproduction operators, we directly adopt the crossover and the mutation operators of **EvoSQL** for simplicity, and no special modifications regarding many-objective algorithms are made in these operators. However, in our preliminary experiment, we find these operators are effective in general. When the stopping criterion is met, the evolution terminates. Finally, the archived solutions are regarded as the set of test databases.

To summarize, we compare **TestDatabaseGen** with the genetic algorithm used in **EvoSQL**. The approach proposed in this study features the following characteristics:

- (1) Many-objective model: unlike the existing approaches in which test database generation is modeled as a single-objective problem, **TestDatabaseGen** adopts a many-objective sorting mechanism, so that the solutions in the population could take all the targets into consideration during the selection. Furthermore, in contrast to **EvoSQL** in which the objective values have to be calculated for all the targets separately, **TestDatabaseGen** could handle all the targets in a single evaluation. Hence, redundant computation could be prevented to some extents.
- (2) Dynamic objective strategy: instead of applying static objective function along the evolution process, **TestDatabaseGen** dynamically removes targets that have been covered. With this strategy, the number of targets decreases along the evolution process, and the search could be focused on the uncovered targets. Consequently, the algorithm scales up well to a relatively large number of targets.

### 3.2 Sub-problem Decomposition Based Reduction

In the second stage, we focus on the test database bloating barrier. To reduce the size of the test database obtained by **TestDatabaseGen**, we develop a decomposition based local search strategy.

The idea is intuitive, i.e., when a candidate database covers one or more targets, it means that there are a series of data rows in the database that can satisfy the predicates in the SQL queries. However, it is possible that not all the data rows are contributive to the coverage. In other words, only a part of the data rows leads to the satisfaction of the predicates. Hence, we need to filter out the values with no contribution, and generate more compact test databases. To realize the reduction effect, we consider the following problem:

**Problem 3.2:** Let  $D = \{t_1, \dots, t_m\}$  be a set of test databases. For each database  $t_i$ ,  $f(t_i) = \{r_{i1}, \dots, r_{im}\} \subseteq R$  represents the targets covered by  $t_i$ . Find a subset of databases  $T' = \{t'_1, \dots, t'_c\}$  that minimizes the following function:

$$\begin{aligned} \min \sum_{i=1}^c size(t'_i), \\ s.t. \bigcup_{i=1}^c f(t'_i) = \bigcup_{i=1}^m f(t_i), \end{aligned} \quad (3)$$

where  $size(t'_i)$  indicates the number of data rows in the test database  $t'_i$ .

Unfortunately, with the increase of the targets, the number of data rows in the database  $T$  will increase accordingly, which leads to the search space explosion problem [16]. Therefore, we propose a decomposition strategy to transform the original problem into a set of sub-problems. Given two databases  $t_1$  and  $t_2$ , we can construct a sub-problem, in search of a database with more compact size in a small neighborhood. More specifically, the sub-problem is defined as follows.

**Problem 3.3:** Let  $D = \{t_1, t_2\}$  be a set of two test databases. For each database  $t_i$ ,  $f(t_i) = \{r_{i1}, \dots, r_{im}\} \subseteq R$  represents the targets covered by  $t_i$ . Find a new database  $t'$  that minimizes the following function:

$$\begin{aligned} \min size(t') \\ s.t. f(t') = f(t_1) \cup f(t_2) \end{aligned} \quad (4)$$

In this way, we can find the solution of the original problem by solving the sub-problem for each pair of test databases.

The main workflow of the second stage is presented in the pseudo code of Algorithm 2 **TestDatabaseReduction**. In the main loop, we set all the solutions in the population as unreached, to indicate whether the solution should be involved in the generation of the next sub-problem. In Lines 3–4, we select two individuals in the population to construct the sub-problem. Then, the **LocalSearch** operator is applied, to obtain a solution to the induced sub-problem. In Lines 6–9, we verify the solution obtained by the local search

operator. If a more compact solution is achieved, the two individuals under examination will be replaced with the reduced solution returned by **LocalSearch**. Otherwise, we continue investigating other pairs of individuals that have not been investigated, until all the individuals have been reached.

In particular, our method adopts a local search operator to solve the induced sub-problem. As presented in Algorithm 3, a hill climbing approach is considered.

**Algorithm 2: TestDatabaseReduction**

Input: databases  $D$ , coverage set  $R$

Output: Best solution

```

1   $population \leftarrow D$ 
2   $unreached \leftarrow \{ \langle t_i, t_j \rangle \mid t_i, t_j \in D, i < j \}$ 
3  while  $unreached$  not empty
4    select a database pair  $\langle t_1, t_2 \rangle$  from  $unreached$ 
5     $t' \leftarrow \mathbf{LocalSearch}(t_1, t_2, R)$ 
6    if  $size(t') < size(t_1) + size(t_2)$ 
7       $population \leftarrow population \setminus \{t_1, t_2\} \cup t'$ 
8       $unreached \leftarrow unreached \cup \{ \langle t_i, t' \rangle \mid t_i \in population \}$ 
9       $unreached \leftarrow unreached \cap \{ \langle t_i, t_j \rangle \mid t_i, t_j \in population, i < j \}$ 
10   else
11      $unreached \leftarrow unreached \setminus \{ \langle t_1, t_2 \rangle \}$ 
12   end if
13 end while
14 return  $population$ 

```

**Algorithm 3: LocalSearch**

Input: database  $t_1$ , database  $t_2$ , coverage set  $R$

Output: reduced database

```

1   $t^* \leftarrow merge(t_1, t_2)$ 
2  while  $size(t) \geq size(t_1) + size(t_2)$  and  $t^*$  changed in while
3     $t \leftarrow t^*$ 
4    for each data row  $r$  of  $t$ 
5      if  $evaluation(\{t \setminus \{r\}\}, R)$  not deteriorated
6         $t \leftarrow t \setminus \{r\}$ 
7      end if
8    end for
9    for each data row  $r$  of  $t$ 
10   if  $evaluation(\{t^* \setminus \{r\}\}, R)$  not deteriorated
11      $t^* \leftarrow t^* \setminus \{r\}$ 
12     break for
13   end if
14 end for
15 end while
16 return  $t$ 

```

In Algorithm 3, a first-improvement local search is realized. More specifically, we construct an incumbent database by merging the two input databases (Line 1). Then, we iteratively examine each data row of the incumbent database (Lines 2–15). If we observe that, the deletion of a data row does not deteriorate the coverage metric, we simply



delete this data row to generate a new database (Line 5–7). Otherwise, we recover the deletion, and make a perturbation accordingly (Lines 9–14). Then, we restart the investigation from the perturbed database. The traversal continues, until all the data rows have been iterated. By embedding the local search operator in Algorithm 2, we are able to accomplish the reduction of the test databases.

As a brief summary, in this section, we present the **TestDatabaseReduction** stage. The reduction algorithm features a hill climbing based local search operator to explore the possibility of minimizing the test databases obtained by the first stage. In the next section, we would conduct extensive experiments to evaluate the proposed approach.

## 4 Experimental Results

### 4.1 Research Questions

In this section, we investigate the performance of **MoeSQL**. Our experiment focuses on the following three Research Questions (RQs).

**RQ1:** How does **MoeSQL** perform in terms of coverage metrics?

**RQ2:** How does **MoeSQL** perform in terms of the runtime and the size metrics?

**RQ3:** How does **MoeSQL** performs over different instances?

In these RQs, **RQ1** is used to verify the feasibility of **MoeSQL**. **RQ2** is adopted to examine whether our algorithm tackles the existing challenges properly. **RQ3** intends to investigate the trade-off between runtime and size metrics achieved by **MoeSQL**.

To evaluate **MoeSQL**, we adopt **EvoSQL**, the state-of-the-art algorithm as the baseline of our experiments. Besides, we also propose a variant algorithm (denoted as **MoeSQLv**) as a comparative approach. In this variant, **MoeSQL** will terminate after the first stage, without further consideration of the scalability issue. In this way, we can investigate the contribution of both stages.

In the experiments, the parameter settings follow those of **EvoSQL**. More specifically, we set the population size  $pop\_num$  to 50. Seeding probability  $p_s$  is set to 0.5. Crossover probability  $p_c$  is set to 0.75. Due to the various operations in mutation operator, the mutation probability  $p_m$  is a set of numbers. The mutation probability for inserting, deleting, and duplicating operation is set to 1/3, the row change mutation probability is set to 1, and the NULL mutation probability is set to 0.1. Our experiments run under a PC with an Intel Core i5 2.3 GHz CPU, 16 GB memory, and Windows 10. All algorithms are implemented in Java 1.8. Our experiments use three datasets provided by **EvoSQL**. Over the instances, we execute each algorithm five times. There are 1888 SQL queries and 10338 coverage targets in total. The statistics of these SQL queries are shown in Table 1. Because SQLFpc may generate some targets that cannot be covered theoretically, we manually examine and delete these targets to ensure that the rest targets could be covered, given sufficient runtime.

**Table 1.** Statistics of the benchmark instances

Feature	#Targets								
	0	1–2	3–4	5–6	7–8	9–10	11–15	16–20	21+
Predicates	57	1278	424	54	27	10	14	21	3
JOINS	1831	41	3	1	11	1	–	–	–
Subqueries	1851	37	–	–	–	–	–	–	–
Functions	1735	149	2	2	–	–	–	–	–
Columns	59	1271	413	85	16	13	14	7	10
Targets	–	645	337	370	310	95	55	27	49

## 4.2 Experimental Results

**Investigation of RQ1.** We first present the coverage statistics of the comparative approaches in Table 2. In the table, the first column indicates the number of targets. Columns 2–3 represent the instance coverage (number of fully covered instances). Columns 4–5 are the target coverage (number of covered targets). The coverage of **MoeSQL<sub>v</sub>** is the same as **MoeSQL**, because the second stage of **MoeSQL** does not alter the coverage metric. From the table, the following phenomena could be observed:

- (1) **MoeSQL** achieves high coverage over all the instances. Similar as **EvoSQL**, **MoeSQL** can cover all targets over instances with less than 10 coverage targets. With the increase of the number of targets, the performance of both algorithms decreases.
- (2) In terms of the target coverage, **MoeSQL** performs slightly better than **EvoSQL**. Over all the instances, **MoeSQL** is able to cover 99.80% of targets. Meanwhile, the target coverage by **EvoSQL** is 99.52%.
- (3) In terms of instance coverage, the results of **EvoSQL** and **MoeSQL** are very close. However, the performance of the two algorithms is not the same. **EvoSQL** performs better over instances with more than 16 but less than 20 coverage targets. Meanwhile, **MoeSQL** has a higher coverage in instances with more than 20 coverage targets.

**Answer to RQ1:** **MoeSQL** can completely cover 99.63% of all instance, which is comparable to the state-of-the-art approach.

**Investigation of RQ2.** In this RQ, we are interested in the efficiency of **MoeSQL**. We calculate the runtime and the size of test database (measured by the number of data rows in the test databases). The statistics are presented in Table 3. The table is organized as follows. The first column indicates the number of targets of the queries. Columns 2–4 represent the median runtime statistics in seconds, for **EvoSQL**, **MoeSQL<sub>v</sub>**, and **MoeSQL**, respectively. Similarly, columns 5–7 are associated with the size statistics, measured by the average number of data rows in the test database, for the three approaches. From the table, we observe that:

**Table 2.** The instance coverage and the target coverage of each algorithm

#Targets	Instance Coverage		Target Coverage	
	EvoSQL	MoeSQL/MoeSQLv	EvoSQL	MoeSQL/MoeSQLv
1–2	<b>645/645</b>	<b>645/645</b>	<b>1232/1232</b>	<b>1232/1232</b>
3–4	<b>337/337</b>	<b>337/337</b>	<b>1095/1095</b>	<b>1095/1095</b>
5–6	<b>370/370</b>	<b>370/370</b>	<b>1970/1970</b>	<b>1970/1970</b>
7–8	<b>310/310</b>	<b>310/310</b>	<b>2314/2314</b>	<b>2314/2314</b>
9–10	<b>95/95</b>	<b>95/95</b>	<b>892/892</b>	<b>892/892</b>
11–15	<b>53/55</b>	<b>53/55</b>	679/699	<b>686/699</b>
16–20	<b>26/27</b>	25/27	473/485	<b>481/485</b>
20+	42/49	<b>46/49</b>	1633/1651	<b>1647/1651</b>

- (1) **MoeSQLv** achieves the minimum runtime over all instances. The time of **MoeSQLv** is almost half that of **EvoSQL** in instances with less than 15 targets. And over other instances, the runtime of **MoeSQLv** is also significantly less than **EvoSQL**.
- (2) **MoeSQL** performs the best over instances with less than 10 targets. Due to the second stage, the runtime of the whole algorithm is longer than **MoeSQLv**. With the increase of the number of targets, the gap between the two variants also increases.
- (3) When considering all the instances, without the second stage, **MoeSQLv** is able to outperform **EvoSQL** by 22.62%, in terms of the size metric of the test database. Moreover, with the reduction mechanism, **MoeSQL** is able to further reduce the test database size by 17.91%. For the instances with more than 20 coverage targets, the overall number of data rows is reduced by up to 68.59%, compared with **EvoSQL**.

**Table 3.** The runtime and the test database size statistics of each algorithm

#Targets	Runtime (s)			Size (#data rows)		
	EvoSQL	MoeSQLv	MoeSQL	EvoSQL	MoeSQLv	MoeSQL
1–2	0.03	0.02	<b>0.02</b>	2.00	2.00	<b>2.00</b>
3–4	0.04	0.02	<b>0.02</b>	3.00	3.00	<b>3.00</b>
5–6	0.07	0.02	0.04	5.00	5.00	<b>5.00</b>
7–8	0.25	0.13	0.20	8.00	7.00	<b>7.00</b>
9–10	0.62	0.32	0.61	11.00	10.00	<b>9.00</b>
11–15	2.13	1.11	3.87	16.00	13.40	<b>11.00</b>
16–20	10.15	7.76	116.30	40.40	33.40	<b>14.40</b>
20+	130.32	108.36	1483.15	54.00	40.00	<b>26.00</b>

**Answer to RQ2:** the advantage of **MoeSQL** in runtime is more reflected over instances with small number of coverage targets. At the same time, **MoeSQL** can significantly reduce the size of the test database, especially for complex instances. Although the second stage of **MoeSQL** costs extra runtime, the local search operator reduces the size of the database. For most instances, the time consumption is acceptable.

**Investigation of RQ3.** To answer RQ3, we classify all instances according to the performance of each algorithm. According to the two performance indicators, i.e., runtime and size, we categorize all the instances into the following four types:

Type A: **MoeSQL** outperforms **EvoSQL** in terms of both indicators.

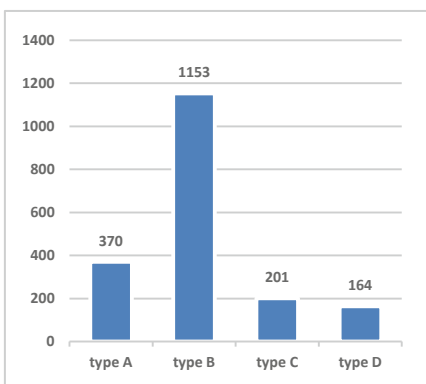
Type B: **MoeSQL** outperforms better than **EvoSQL** in terms of runtime, and the size of **MoeSQL** is the same as **EvoSQL**.

Type C: The size of **MoeSQL** is better than **EvoSQL**, but the runtime metric of **MoeSQL** is inferior to that of **EvoSQL**.

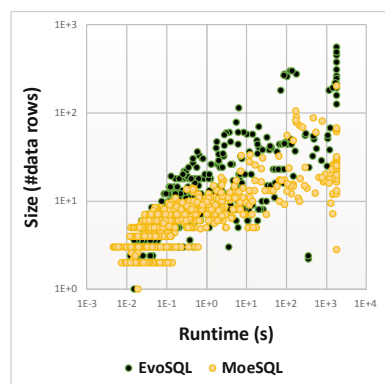
Type D: **MoeSQL** fails to outperform **EvoSQL** in terms of either indicator.

We summarize the number of instances of each type. The statistics of these type are shown in Fig. 2(a). According to the figure, we observe that:

- (1) **MoeSQL** is more time efficient than **EvoSQL** over the majority (types A and B, 1523/1888) of instances. Over these instances, **MoeSQL** can find test databases of same or more compact size than **EvoSQL**. In particular, over (370/1888) 19.60% of instances, **MoeSQL** outperforms **EvoSQL** for both performance indicators.
- (2) Over (201/1888) 10.65% of instances, **MoeSQL** consumes more time than **EvoSQL**, but is able to achieve more compact solutions. Only over (164/1888) 8.69% of instances, **MoeSQL** is dominated by **EvoSQL**.



(a) Statistics of instance types



(b) Runtime-size comparison

**Fig. 2.** Comparison between **EvoSQL** and **MoeSQL**

To gain more insights, we plot the runtime and the size metrics obtained by **EvoSQL** and **MoeSQL** over all the instances in Fig. 2(b). From the figure, we observe that the points for **MoeSQL** are concentrated in the area closer to the origin, which to some extent demonstrates the ability of **MoeSQL** to balance the runtime and the size.

**Answer to RQ3:** **MoeSQL** performs better than **EvoSQL** over most instances, and is able to achieve moderate trade-off between the runtime and the size metrics.

## 5 Conclusion and Future Work

In this paper, we present a novel two-stage algorithm **MoeSQL** to solve the test database generation for SQL queries. The proposed approach features the combination of a many-objective evolutionary algorithm and a local search based reduction mechanism, to tackle the inter-objective barrier and the test database bloating barrier. Experimental results over real-world datasets demonstrate the effectiveness of **MoeSQL**.

Despite the promising results, there is still room for improvement. For example, the local search based reduction is time consuming. To mitigate the limitation, an interesting direction is to consider very large neighborhood search [17] or surrogate based acceleration mechanisms [18]. If feasible, the efficient reduction mechanisms may enable more advanced algorithms, such as reduction during evolution.

**Acknowledgement.** This work is supported in part by the National Key Research and Development Program of China under grant no. 2018YFB1003900, and the National Natural Science Foundation of China under grant no. 61772107, 61722202.

## References

1. Fraser, G., Arcuri, A., McMinn, P.: Test suite generation with memetic algorithms. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, pp. 1437–1444. ACM, New York (2013)
2. Arcuri, A.: RESTful API automated test case generation with Evo-Master. *ACM Trans. Softw. Eng. Methodol.* **28**(1), 1–37 (2019)
3. Castelein, J., Aniche, M., Soltani, M., Panichella, A., van Deursen, A.: Search-based test data generation for SQL queries. In: Proceedings of the 40th International Conference on Software Engineering, pp. 1220–1230. ACM, Gothenburg (2018)
4. Suárez-Cabal, M.J., de la Riva, C., Tuya, J., Blanco, R.: Incremental test data generation for database queries. *Autom. Softw. Eng.* **24**(4), 719–755 (2017). <https://doi.org/10.1007/s10515-017-0212-7>
5. Shah, S., Sudarshan, S., Kajbaje, S., Patidar, S., Gupta, B., Vira, D.: Generating test data for killing SQL mutants: a constraint-based approach. In: 2011 IEEE 27th International Conference on Data Engineering, pp. 1175–1186. IEEE, Hannover (2011)
6. Panichella, A., Kifetew, F., Tonella, P.: Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. *IEEE Trans. Softw. Eng.* **44**(2), 122–158 (2018)

7. Tuya, J., de la Riva, C., Suárez-Cabal, M., Blanco, R.: Coverage-aware test database reduction. *IEEE Trans. Softw. Eng.* **42**(10), 941–959 (2016)
8. Tuya, J., Suárez-Cabal, M., de la Riva, C.: Full predicate coverage for testing SQL database queries. *Softw. Test. Verif. Reliab.* **20**(3), 237–288 (2010)
9. Chilenski, J., Miller, S.: Applicability of modified condition/decision coverage to software testing. *Softw. Eng. J.* **9**(5), 193–200 (1994)
10. Garcia-Molina, H., Ullman, J.D., Widom, J.: *Database System Implementation*. Prentice Hall, Upper Saddle River (2000)
11. Korel, B.: Automated software test data generation. *IEEE Trans. Softw. Eng.* **16**(8), 870–879 (1990)
12. Rojas, J., Fraser, G., Arcuri, A.: Seeding strategies in search-based unit test generation. *Softw. Test. Verif. Reliab.* **26**(5), 366–401 (2016)
13. Rojas, J., Vivanti, M., Arcuri, A., Fraser, G.: A detailed investigation of the effectiveness of whole test suite generation. *Empir. Softw. Eng.* **22**(2), 852–893 (2017). <https://doi.org/10.1007/s10664-015-9424-2>
14. Panichella, A., Kifetew, F., Tonella, P.: Reformulating branch coverage as a many-objective optimization problem. In: *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, pp. 1–10. IEEE, Graz (2015)
15. Goldberg, D., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In: *Proceedings of the First Workshop on Foundations of Genetic Algorithms*, pp. 69–93. Elsevier, Indiana (1991)
16. Ramírez, A., Romero, J., Ventura, S.: A survey of many-objective optimisation in search-based software engineering. *J. Syst. Softw.* **149**, 382–395 (2019)
17. Ghoniem, A., Flamand, T., Haouari, M.: Optimization-based very large-scale neighborhood search for generalized assignment problems with location/allocation considerations. *INFORMS J. Comput.* **28**(3), 575–588 (2016)
18. Pan, L., He, C., Tian, Y., Wang, H., Zhang, X., Jin, Y.: A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization. *IEEE Trans. Evol. Comput.* **23**(1), 74–88 (2018)



# A New Paradigm in Interactive Evolutionary Multiobjective Optimization

Bhupinder Singh Saini<sup>(✉)</sup> , Jussi Hakanen , and Kaisa Miettinen 

Faculty of Information Technology, University of Jyväskylä, P.O. Box 35 (Agora),  
40014 Jyväskylä, Finland

{bhupinder.s.saini,jussi.hakanen,kaisa.miettinen}@jyu.fi

**Abstract.** Over the years, scalarization functions have been used to solve multiobjective optimization problems by converting them to one or more single objective optimization problem(s). This study proposes a novel idea of solving multiobjective optimization problems in an interactive manner by using multiple scalarization functions to map vectors in the objective space to a new, so-called preference incorporated space (PIS). In this way, the original problem is converted into a new multiobjective optimization problem with typically fewer objectives in the PIS. This mapping enables a modular incorporation of decision maker's preferences to convert any evolutionary algorithm to an interactive one, where preference information is directing the solution process. Advantages of optimizing in this new space are discussed and the idea is demonstrated with two interactive evolutionary algorithms: IOPIS/RVEA and IOPIS/NSGA-III. According to the experiments conducted, the new algorithms provide solutions that are better in quality as compared to those of state-of-the-art evolutionary algorithms and their variants where preference information is incorporated in the original objective space. Furthermore, the promising results require fewer function evaluations.

**Keywords:** Interactive methods · Achievement scalarizing functions · Evolutionary algorithms · Preference information · Decision maker

## 1 Introduction

Many multiobjective optimization problems (MOPs) are encountered in real life applications. Due to the conflicting nature of objectives in these problems, often there does not exist a single optimal solution. Instead, there exists a set of *Pareto optimal solutions* which represent trade-offs among the various objectives.

One family of methods, known as *a posteriori* methods, solve MOPs by finding a set of solutions which adequately represents the entire set of Pareto optimal solutions [18]. Evolutionary algorithms (EAs) have been employed for this with a varying degree of success. An example of such methods is NSGA-II [6], which works well for solving problems with a low number of objectives, but the performance degrades as the number of objectives increases [9]. Recent *a posteriori* EAs have tackled this problem in various ways [14].

However, increasing the number of objectives brings forth new challenges. As the number of objectives increases, the number of solutions required to adequately represent the set of Pareto optimal solutions (which may have an infinite number of solutions) increases exponentially [9, 14]. Regardless of the number of objectives, only one or few of these solutions are useful to a *decision maker* (DM) who wants to find and implement the desirable solution. Hence, when using a posteriori methods, computational resources are wasted on finding solutions that are not relevant. If objective function evaluations require time-consuming simulations or physical experiments, this problem is compounded and may lead to a waste of monetary resources as well. Moreover, these algorithms leave the task of choosing the final solution to the DM. As each solution is a vector in a high-dimensional objective space, comparing potentially thousands of solutions is a difficult task. This process can set a high cognitive load on the DM.

As DMs are experts in their domain, they usually have opinions or *preferences* regarding which solutions are desirable or undesirable to them. The preference information may be elucidated in the form of desirable objective function values, ranking of importances of objectives, pair-wise comparison of solutions and many other techniques [16]. Recent advances in EAs try to incorporate this information to limit the scope of search of the EA. As the DM may learn new information about the problem during the solution process, allowing them to change their preferences during the solution process is desirable [11]. Methods which allow such change are known as *interactive* methods [17–19, 30]. Ideally, this leads to less waste of resources as only solutions that are preferable to the DM are focused upon. Moreover, as only a small subset of the Pareto optimal solutions is to be represented at a time, the number of solutions to be shown to the DM is smaller, hence reducing the cognitive load. However, many interactive EAs have problems ranging from addition of hyperparameters to lack of diversity in the population, which can impair the optimization process [1, 10].

The concept of utilizing the preferences of a DM in the solution process of an MOP is very popular in the field of multiple criteria decision making [18, 19]. One of the methods adopted is to use scalarization functions [18, 20]. These functions utilize the preferences of the DM to map the objective function values of solutions to scalar values, hence converting the MOP to one or more single objective optimization problems. Different scalarization functions interpret the same preference information differently, and may lead to different results [20]. Different solutions can hence be obtained by solving multiple scalarization functions with the same preference information, as done in synchronous NIMBUS [21], or by slightly modifying the preference information multiple times and optimizing the same scalarization function, as done in the reference point algorithm [28].

In this paper, we explore the concept of using multiple scalarization functions to create a new space: *Preference Incorporated Space* (PIS). First, we study the mathematical properties of this new space. More specifically, we study the effect of optimizing in the PIS, introducing a new paradigm in preference based optimization: Interactive Optimization using Preference Incorporated Space (IOPIS) algorithm. The IOPIS algorithm enables us to make use of preference



information with any a posteriori EA in an interactive way, as the preference information is encoded directly in the optimization problem in the PIS. It also enables us to control the dimension of the space in which dominance is judged, equal to the number of chosen scalarization functions. We then introduce the IOPIS algorithm, a modular algorithm that takes a given number of specified scalarization functions, and uses a DM’s preferences to convert a generic MOP to an MOP in the preference incorporated space. This can then be solved interactively with any appropriate non-interactive EA together with DM’s preferences. As examples, we implement two versions of the new algorithm: IOPIS/RVEA and IOPIS/NSGA-III, where the new problem in the PIS is optimized using decomposition based EAs RVEA [4] and NSGA-III [5], respectively.

The rest of the paper is organized as follows. Section 2 discusses the background of multiobjective optimization, EAs, and scalarization functions. Section 3 discusses the mathematical properties of the PIS and introduces the IOPIS algorithm with a visual explanation. In Sect. 4, we conduct an experimental study to compare the performances of the two implementations of the IOPIS algorithm with state of the art a posteriori EAs and their interactive variants and discuss the results. Finally, we draw conclusions in Sect. 5. All implementations and experimental data presented in this paper are open source and publicly available at <https://desdeo.it.jyu.fi> as a part of the DESDEO framework.

## 2 Background

### 2.1 Multiobjective Optimization

An MOP can be defined as:

$$\begin{aligned} & \text{minimize } \{f_1(\mathbf{x}), \dots, f_k(\mathbf{x})\} \\ & \text{subject to } \mathbf{x} \in S, \end{aligned} \tag{1}$$

where  $\mathbf{x} = (x_1, \dots, x_n)^T$  are vectors of decision variables belonging to the feasible set  $S \subset \mathbb{R}^n$ . The  $k (\geq 2)$  objective functions  $f_i$  map vectors of  $S$  to  $\mathbb{R}$ . The objective function values  $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))$  form objective vectors in the objective space  $\mathbb{R}^k$ . A solution  $\mathbf{x}^1 \in S$  of problem (1) is said to *dominate* another solution  $\mathbf{x}^2 \in S$  (written as  $\mathbf{f}(\mathbf{x}^1) \succ \mathbf{f}(\mathbf{x}^2)$ ) if  $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2)$  for all  $i = 1, \dots, k$  and  $f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$  for at least one  $j = 1, \dots, k$ . Pareto optimal solutions are solutions of the MOP which are not dominated by any other solution in  $S$ . For this reason, they are also referred to as *non-dominated solutions*. Sometimes, it is desirable for DMs to consider a subset of Pareto optimal solutions with bounded trade-offs [18]. Such solutions are called *properly Pareto optimal* solutions.

We can define the set of solutions of problem (1), known as a Pareto set, as:

$$PS_{OS} = \{\mathbf{x} \in S \mid \nexists_{\mathbf{x}^* \in S} \mathbf{f}(\mathbf{x}^*) \succ \mathbf{f}(\mathbf{x})\}, \tag{2}$$

where the subscript OS refers to the fact that the set was obtained by considering the objective vectors in the objective space. We can now define an *ideal point*

and a *nadir point* of problem (1). These points represent the lower and upper bounds of the ranges of the objective function values among the Pareto optimal solutions, respectively. The ideal point  $\mathbf{z}^* = (z_1^*, \dots, z_k^*)$  can be calculated as  $z_i^* = \min_{\mathbf{x} \in S} f_i(\mathbf{x})$ . The nadir point  $\mathbf{z}^{nad} = (z_1^{nad}, \dots, z_k^{nad})$  can be calculated as  $z_i^{nad} = \max_{\mathbf{x} \in PS_{OS}} f_i(\mathbf{x})$ . It should be noted that calculating the nadir point requires the calculation of the  $PS_{OS}$ . Hence, the calculation of the nadir point is tricky in problems with more than two objectives and needs to be estimated [8, 18]. Any objective vector  $\mathbf{z}$  is defined to be *achievable* if  $\mathbf{z}$  belongs to the set:

$$T = \{\mathbf{z} \in \mathbb{R}^k \mid \exists \mathbf{x} \in S \mathbf{f}(\mathbf{x}) \succ \mathbf{z} \text{ or } \mathbf{f}(\mathbf{x}) = \mathbf{z}\}. \quad (3)$$

By definition, the nadir point is an achievable point, while the ideal point is not.

## 2.2 Evolutionary Algorithms

Decomposition-based methods such as NSGA-III [5], RVEA [4], and many variants of MOEA/D [31] have become popular in the evolutionary multiobjective optimization community. These methods decompose the objective space into sections using directional vectors called reference vectors, reference points, or weights. For simplicity, in what follows, we will be using the term reference vectors (RVs). These RVs, usually spread uniformly in the objective space, represent individual single-objective optimization problems. The RVs are typically generated using a simplex lattice design, and the number of RVs is equal to  $\binom{l+k-1}{k-1}$ , where  $l$  is a parameter controlling the density of the RVs. Subsets of the population which lie in the decomposed region associated with an RV (in the objective space) evolve in the direction of that RV based on scalar fitness values calculated using the RV and their objective function values.

As mentioned in the introduction, EAs which approximate the entire Pareto front exhibit many downsides. Methods have been proposed to get around those downsides by incorporating the preferences of the DM in an interactive fashion, see, e.g. [23, 26, 30]. One of the ways to incorporate a DM's preferences in decomposition-based EAs is to manipulate the spread of the RVs to account for the preferences [4, 15]. In many such methods, the DM is required to provide their preferences in the form of a *reference point* in the objective space [12, 26, 27]. The components of a reference point are desirable values of each objective function, which may or may not be achievable. Then, uniformly spread RVs are translated towards this point. This translation introduces a new scalar hyperparameter which controls the final spread of the RVs around the reference point. This method introduces a few new problems, though. Firstly, the effect of changing the value of the newly introduced hyperparameter may be difficult for a DM to understand. But an appropriate value for this hyperparameter is important as it has been observed that a small spread of RVs may lead to a degradation in population diversity, which prohibits the convergence of the EA [1, 10].

## 2.3 Achievement Scalarizing Functions

As mentioned, scalarization functions are functions that map a vector to a real-valued scalar. The weighted sum function and the Chebyshev function used by

MOEA/D and the angle-penalized distance function used by RVEA are examples of scalarization functions [4, 31]. To be regarded as a good scalarization function, it must have some desirable properties [25]. Firstly, the solutions obtained by optimizing the scalarization function should be Pareto optimal. Secondly, these solutions should be satisfactory according to the preferences of a DM, if the preferences are feasible. Finally, any Pareto optimal solution should be discoverable by changing the preferences provided by the DM.

Unfortunately, no single scalarization function satisfies the three conditions concurrently [25]. However, if we relax the conditions to only account for properly Pareto optimal solutions, rather than all Pareto optimal solutions, then all three conditions can be satisfied by some scalarization functions. In this paper, we focus on a subclass of scalarization functions, known as achievement scalarizing functions (shortened to achievement function) [29]. An achievement function is a continuous function  $s : \mathbb{R}^k \rightarrow \mathbb{R}$ . Achievement functions are characterized by either being strictly increasing and order-representing, or strongly increasing and order-approximating [29]. We will focus on the latter kind as they satisfy all three relaxed desirable properties.

**Theorem 1** [29]. *Let us consider  $\mathbf{z}^1, \mathbf{z}^2 \in \mathbb{R}^k$  such that  $\mathbf{z}^1 \succ \mathbf{z}^2$ . Then for any order-approximating achievement function  $s : \mathbb{R}^k \rightarrow \mathbb{R}$ , we have*

$$s(\mathbf{z}^1) < s(\mathbf{z}^2). \tag{4}$$

From Theorem 1, it can be concluded that solving the following problem:

$$\begin{aligned} & \text{minimize } s(\mathbf{f}(\mathbf{x})) \\ & \text{subject to } \mathbf{x} \in S \end{aligned} \tag{5}$$

will lead to a Pareto optimal solution of problem (1) [28, 29].

A general formulation of an (order-approximating) achievement function is:

$$s(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}) = \max_{i=1, \dots, k} \left[ \frac{f_i(\mathbf{x}) - \bar{z}_i}{\mu_i} \right] + \rho \sum_{i=1}^k \left( \frac{f_i(\mathbf{x}) - \bar{z}_i}{\mu_i} \right), \tag{6}$$

where  $\rho$  is a small positive scalar and  $\mu_i$  are positive scalars and  $\bar{\mathbf{z}} \in \mathbb{R}^k$  is a reference point provided by the DM [24]. Minimizing  $s(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}})$  has the effect of optimizing problem (1) by sliding a cone along the line  $\bar{\mathbf{z}} + \lambda \boldsymbol{\mu}$ , where  $\lambda \in \mathbb{R}$ , so that a minimum ( $>0$ ) number of solutions lie in the cone [20]. Bounds of the trade-offs in the solutions obtained by (5) can be controlled by changing  $\rho$  [28].

The general formulation (6) represents achievement functions that can take preferences in other forms, not just reference points [24]. Different achievement functions differ in how  $\boldsymbol{\mu}$  is set, which means they are optimizing along different directions, albeit starting from the same reference point  $\bar{\mathbf{z}}$ . Hence, they may lead to different solutions even if the same reference point is provided to them. For the implementation of the IOPIS algorithm, we focus on the GUESS [2] and STOM [22] scalarization functions (based on e.g., [3, 20]). For the GUESS function,  $\mu_i = z_i^{nad} - \bar{z}_i$  and for the STOM function,  $\mu_i = \bar{z}_i - z_i^*$ . As  $\mu_i > 0$  for all

$i = 1, \dots, k$ , it follows that for these two achievement functions,  $z_i^* < \bar{z}_i < z_i^{nad}$  for all  $i = 1, \dots, k$ . Another achievement function of note is the achievement scalarizing function (ASF) used in the reference point method [28], which is used in the experimental study section. For ASF,  $\mu_i = z_i^{nad} - z_i^*$ .

### 3 Optimization in Preference Incorporated Space

#### 3.1 Properties of Preference Incorporated Space

Let there be a set of achievement functions  $\mathbf{s} = \{s_1, \dots, s_q\}$  with  $q \geq 2$ . Then we can define a PIS as the set  $\{\mathbf{s}(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}) \in \mathbb{R}^q\}$ , and a new MOP in the PIS as:

$$\begin{aligned} &\text{minimize } \mathbf{s}(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}) = \{s_1(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}), \dots, s_q(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}})\} \\ &\text{subject to } \mathbf{x} \in S. \end{aligned} \tag{7}$$

Two solutions  $\mathbf{x}^1$  and  $\mathbf{x}^2$  can now be compared in two spaces. As stated in Sect. 2.1, a solution  $\mathbf{x}^1$  is said to dominate another solution  $\mathbf{x}^2$  in the objective space if  $\mathbf{f}(\mathbf{x}^1) \succ \mathbf{f}(\mathbf{x}^2)$ . A solution  $\mathbf{x}^1$  is said to dominate  $\mathbf{x}^2$  in the PIS if  $\mathbf{s}(\mathbf{f}(\mathbf{x}^1), \bar{\mathbf{z}}) \succ \mathbf{s}(\mathbf{f}(\mathbf{x}^2), \bar{\mathbf{z}})$ . Similar to (2), we can define the solutions to problem (7), i.e., the Pareto set obtained by optimizing in the PIS as:

$$PS_{PIS} = \{x \in S \mid \nexists_{x^* \in S} \mathbf{s}(\mathbf{f}(\mathbf{x}^*), \bar{\mathbf{z}}) \succ \mathbf{s}(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}})\}. \tag{8}$$

We modify the desirable properties of scalarization functions as stated in [25] to reflect properties related to the PIS as:

1. Pareto optimal solutions in the PIS remain Pareto optimal in the objective space.
2. Pareto optimal solutions in the PIS follow the preference given by the DM in the objective space.
3. Any properly Pareto optimal solution of problem (1) can be discovered by changing the reference point of problem (7).

It can be shown that the first condition is true regardless of the choice or number of the achievement functions.

**Theorem 2.** *Let  $PS_{PIS}$  be the set of Pareto optimal solutions of problem (7). Let  $PS_{OS}$  be the set of Pareto optimal solutions of problem (1). Then,*

$$PS_{PIS} \subset PS_{OS}. \tag{9}$$

*Proof.* Suppose  $\mathbf{x} \in PS_{PIS}$  but  $\mathbf{x} \notin PS_{OS}$ . Therefore, there exists some  $\mathbf{x}^*$  such that  $\mathbf{f}(\mathbf{x}^*) \succ \mathbf{f}(\mathbf{x})$ . Thus, according to Theorem 1,  $s_z^i(\mathbf{f}(\mathbf{x}^*)) < s_z^i(\mathbf{f}(\mathbf{x}))$  for all  $i \in \{1, \dots, q\}$ . Hence,  $\mathbf{s}_z(\mathbf{f}(\mathbf{x}^*)) \succ \mathbf{s}_z(\mathbf{f}(\mathbf{x}))$ , which contradicts  $\mathbf{x} \in PS_{PIS}$ .  $\square$

The set  $PS_{PIS}$  represents the trade-offs between the values of the various achievement functions in problem (7). As the different achievement functions are different interpretations of the same preference information obtained from a

DM, the solutions in the set  $PS_{PIS}$  represent the trade-offs between those interpretations. Hence, it can be said that solutions obtained by solving problem (7) follow the preferences given by the DM. Moreover, as  $PS_{PIS}$  includes solutions which minimize individual achievement functions present in PIS, and as any properly Pareto optimal solution in the objective space can be found using the achievement functions by changing the reference points, it follows that the third condition also holds. Note that these results are valid for all order-approximating scalarization functions, and not just STOM and GUESS functions.

Solving the MOP in the PIS has a few benefits compared to solving the MOP in the objective space. Firstly, we can control the dimension of the PIS, which is equal to the number of achievement functions chosen. This means that, given some number ( $\geq 2$ ) of achievement functions, we can use any multiobjective EA (or biobjective EA, as PIS can be a two dimensional space) to solve problem (7), regardless of the number of objectives in the original problem. Secondly, controlling the dimension of the optimization problem also gives us an indirect control over the number of function evaluations needed by the EA during the optimization process. This is because the number of solutions required to adequately represent the set of Pareto optimal solutions increases with increasing the dimension of the objective space (for problem (1)) or PIS (for problem (7)). Hence, choosing fewer achievement functions than  $k$  is an easy way to reduce the number of function evaluations needed by an EA to solve an optimization problem. Thirdly, by incorporating the preference information in the PIS, we gain the ability to use any non-interactive EA in an interactive fashion. This modularity enables easy use of well-tested EAs without needing to change them to enable interaction with a DM.

### 3.2 The IOPIS Algorithm

The IOPIS algorithm takes a formulation of the optimization problem of the form (1) as input. The algorithm also takes as its input a set of achievement functions  $s$ . The ideal point  $\mathbf{z}^*$  and the nadir point  $\mathbf{z}^{nad}$  of the problem are also taken as inputs. As the calculation of the nadir point can be tricky in problems with more than two objectives, approximate values of the nadir point (and ideal point) can also be used. The solutions are generated between these two points. Hence, the DM can use their expertise to give the approximate values of the points within which to search. The interactive solution process begins when these points are shown to the DM. The following four steps are repeated iteratively until the DM has received a satisfactory solution:

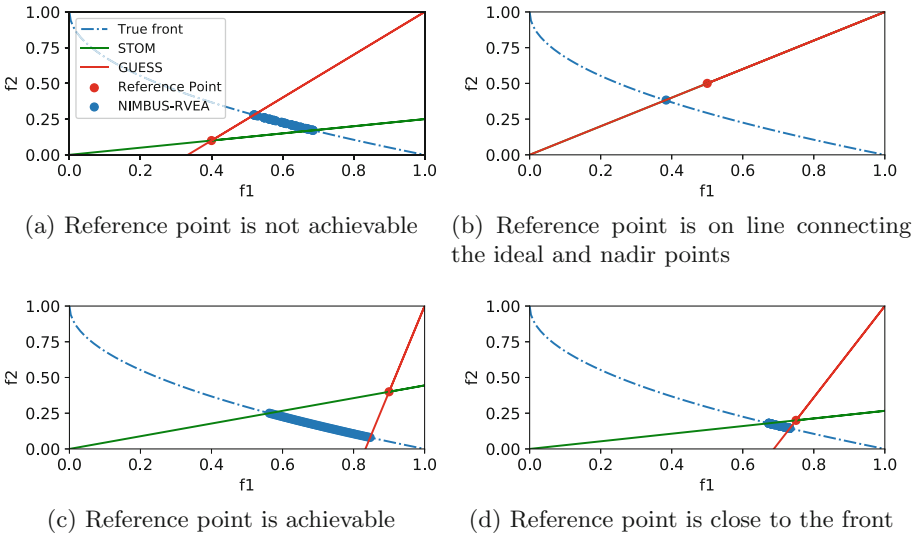
1. *Preference elicitation:* The DM is asked to give their preferences as a reference point based on the information currently available to them.
2. *Problem creation:* Using the original objectives, the known estimates of the ideal and nadir points, the reference point, and the set of achievement functions, a new optimization problem is created in the PIS, as shown in (7).
3. *Problem solution:* Solve the problem created in the previous step with an EA. If this is the first iteration of the algorithm, start the EA with a new

population, generated in a manner specific to the selected EA. In subsequent iterations, the population from the previous iteration is used as the starting population.

4. *Display solutions*: Display the solutions obtained in step 3 to the DM. The DM can indicate the maximum number of solutions to be shown at a time in step 4. If the number of solutions generated in step 3 exceeds the limit, e.g., clustering can be applied before displaying solutions.

### 3.3 Visual Interpretation

Even though the IOPIS algorithm is designed for optimization problems with more than two objectives, a biobjective problem is easily visualizable to demonstrate the algorithm. Here we use the ZDT1 problem [32] to study the effect of the choice of the reference point on the solutions returned by one implementation of the IOPIS algorithm. In this implementation, STOM and GUESS scalarization functions are used with NSGA-III to solve the resulting MOP in the PIS.



**Fig. 1.** Solutions obtained for various reference points for the ZDT1 problem. (Color figure online)

Four different reference points are given to the algorithm. Each subfigure in Fig. 1 shows the corresponding objective vectors returned by the IOPIS algorithm. In each subfigure, the blue dashed curve represents the true Pareto front of the problem and the red point is the reference point. The green line represents the direction along which the STOM function optimizes, whereas the red line represents the direction along which the GUESS function optimizes.

1. The reference point is not achievable (Fig. 1a): There is no solution that achieves values close to the reference point. Minimizing each achievement function individually returns the solution corresponding to the point of intersection of the line representing that achievement function and the Pareto front. As can be seen, the solutions returned by the algorithm include those solutions, and nondominated solutions in between.
2. Reference point is on the line joining the ideal and nadir point (Fig. 1b): Due to the nature of the chosen achievement functions, only a single solution is returned by the algorithm. This is because if the reference point is on the line connecting the ideal and nadir, both achievement functions optimize along the same line. This behaviour can be changed by choosing a different set of achievement functions to form the PIS, or by shifting the reference point slightly to increase the diversity of the solutions.
3. The reference point is achievable and dominated (Fig. 1c): The algorithm returns a set of solutions that satisfy the given reference point. As in the first case, optimal solutions of the individual achievement functions are included.
4. The reference point is close to the front (Fig. 1d): Bringing the reference point closer to the front has the effect of reducing the spread of the solutions returned by the algorithm, hence solutions are returned in a narrower region.

The spread of the solutions is controlled by the position of the reference point. A DM who does not know very well what is realistic may provide a reference point far from the front. In such cases, the algorithm will return a diverse set of solutions (with an exception and possible resolutions discussed in point 2. above). After being provided with such solutions, the DM will have more knowledge about the trade-offs involved among the solutions, and may want to fine-tune their search in a narrow region. This is easily accomplished by providing a reference point closer to the now known region of the front. This methodology of control is similar to the one proposed in the reference point method [28].

## 4 Numerical Results

### 4.1 Experimental Setup

In this study, two versions of the interactive IOPIS algorithm were implemented. IOPIS/NSGA-III uses NSGA-III to solve the problem in the PIS, while IOPIS/RVEA uses RVEA. In both implementations, the STOM and GUESS functions are used as achievement functions to form the PIS. These algorithms were compared against a posteriori RVEA [4] and NSGA-III [5]. Interactive versions of the two a posteriori algorithms (iRVEA and iNSGA-III) were also implemented and included in this study. The details of iRVEA can be found in [12] and iNSGA-III was implemented in a similar manner. RVEA and NSGA-III were chosen for this study as they have been shown to work well in problems with  $k > 2$  [4, 5]. Even though the problem in the PIS here is biobjective, the implementations of the IOPIS algorithm use RVEA and NSGA-III to ensure that only the effect of optimizing in the PIS is reflected in the results, and not the choice of the EA.

The algorithms were compared using the DTLZ{2–4} [7] and WFG{1–9}[13] problems, with 3–9 objectives each. The number of variables was kept as  $10 + k - 1$ , as recommended in [7]. For the IOPIS EAs, each component of the nadir point was randomly generated from a halfnormal distribution with the underlying normal distribution centered around 1 and having a scale of 0.15, then being scaled up by a factor equal to the true nadir point components (1 for the DTLZ problems, varying values for the WFG problems). This led to the generation of nadir points with components up to 50% worse than the true nadir point. This was done to test the performance of the IOPIS EAs in cases where only approximate values of the nadir point are available. The true ideal point was provided to the IOPIS EAs, as the calculation of it is relatively simpler.

Each EA was run for four iterations. For each EA, an iteration consisted of a constant number of generations: One of {100, 150, 200, 250} for the DTLZ problems, 100 for the WFG problems (The reason for using only 100 generations per iteration for WFG problems will be discussed in the next subsection.). All other hyperparameters, such as the number of solutions or algorithm specific hyperparameters, were set to values recommended in their respective papers. In each iteration, all interactive EAs received a common reference point randomly generated in a hyperbox with the ideal and nadir points as opposing vertices. The non-interactive EAs were ran through the iterations uninterrupted. Hence 336 tests with the DTLZ problems<sup>1</sup> and 252 tests with the WFG problems<sup>2</sup> were conducted for each of the six EAs. The algorithms were compared based on the optimality and the preferability of the solutions returned at the end of each iteration, and the number of function evaluations conducted.

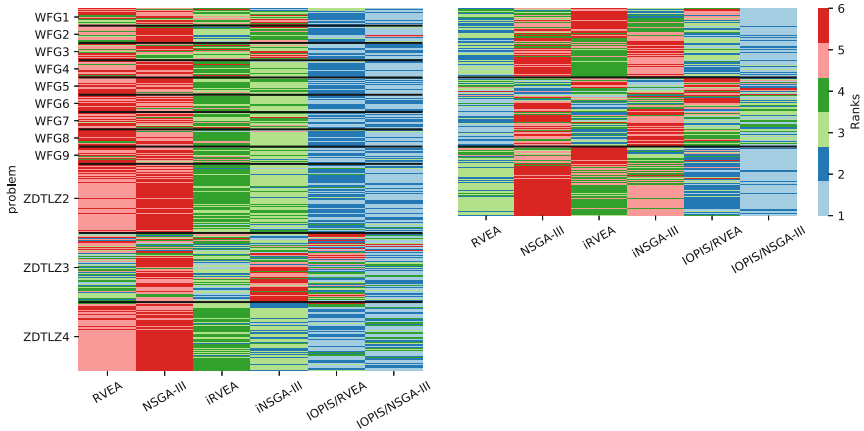
## 4.2 Experimental Results

The Pareto optimal solutions of the DTLZ2-4 problems form a hypersphere in the objective space, centered around the ideal point, which is the origin, and a radius of one. Hence, calculating the Euclidean norm of the objective vectors of the solutions returned by the EAs is a measure of optimality, with lower values of the norm being closer to the Pareto front. However, these values cannot be compared between problems, nor can they be compared for the same problem but with a different number of objectives. Instead, the median of the norm of the solutions returned at the end of each test was calculated for each of the six EAs. These values were then used to rank each EA from 1 to 6 for every test, lower ranks being given to better (lower) median norm values. A similar procedure was followed for comparing methods based on the preferability of the solutions returned by the EAs. The achievement function used in the reference point method (ASF) [28] was chosen as the metric of preferability. The median ASF values of the solutions were then used to rank the different EAs. The true nadir point was used in the calculation of the ASF values. For the tests involving the WFG 1–9 problems, ranks were only calculated based on median ASF values.

<sup>1</sup> = 3 (problems) \* 7 (objectives) \* 4 (generations per iteration) \* 4 (iterations).

<sup>2</sup> = 9 (problems) \* 7 (objectives) \* 4 (iterations).





(a) Ranks based on median ASF values    (b) Ranks based on median norm values

**Fig. 2.** Heatmaps of ranks of algorithms based on the median ASF value or median norm value of the solutions obtained. (Color figure online)

The Pareto fronts for these problems are not spherical, hence ranks based on median norm values are not relevant.

Heatmaps of the ranks based on ASF and norm are shown in Figs. 2a and 2b, respectively. A paired colormap was used in the creation of the heatmaps which gave ranks 1 and 2 a blue hue, ranks 3 and 4 a green hue, and ranks 5 and 6 a red hue. This choice brings forward a clear clustering in the rankings of the 6 EAs. As seen in Fig. 2a, iRVEA and iNSGA-III tend to return more preferable solutions than their non-interactive counterparts. This behaviour is expected as RVEA and NSGA-III focus on the entire Pareto front, whereas iRVEA and iNSGA-III focus on a limited region. However, as seen in Fig. 2b, the solutions returned by iRVEA were farther away from the Pareto front compared to RVEA. This is because as iRVEA has a much lower diversity of solutions compared to RVEA, which hampers the optimization process.

In both heatmaps, the PIS based EAs get ranks 1 or 2 in most tests, i.e., these algorithms returned solutions that were more preferable, and closer to the Pareto front than the other four algorithms. It should also be noted that IOPIS/NSGA-III performed better than IOPIS/RVEA in most cases. Further investigation of the PIS is required on this. The results obtained on the DTLZ3 problem are also interesting. RVEA returned solutions which were closer to the Pareto front, compared to the other methods. While the IOPIS EAs still outperformed RVEA based on the preferability of the solutions, RVEA outperformed an interactive method iNSGA-III. This is happening as iNSGA-III failed to converge to the Pareto front because of the lack of diversity of the solutions, and got stuck on one of the local fronts of the problem. While there was a correlation between the problem type and the performance of the methods (IOPIS EAs got ranks one or two more often in the WFG problems compared to the DTLZ

problems), there was no correlation between the performance of the method and the number of objectives. In the case of the DTLZ problems, the performance was also not dependent on the number of generations per iteration, i.e., there was no improvement in the results after a hundred generations (per iteration). This is why the number of generations per iterations was fixed to 100 for the tests involving the WFG problems.

The final metric of comparison is the number of function evaluations conducted. Given a constant number of generations, the number of function evaluations is linearly correlated with the population size, which is equal to the number of RVs in the EAs considered in this paper. For RVEA, NSGA-III, iRVEA and INSGA-III, the RVs (and hence the number of function evaluations) increase exponentially with an increasing number of objectives. As the IOPIS algorithms operate in the low-dimensional PIS, the number of reference vectors, and hence the number of function evaluations, is independent of the number of objectives, and significantly lower than that for the other algorithms considered in the study. It should also be noted that for all of the tests, only an approximate nadir point was provided to the IOPIS EAs, and yet the IOPIS EAs obtain better results than the current state of the art algorithms.

## 5 Conclusions

A new space PIS, where preferences are incorporated, was proposed as a new paradigm of solving MOPs interactively. This new space makes the creation of interactive EAs very modular, as the algorithm only needs to modify the problem to enable interactivity, rather than the EA itself. As examples, this enabled easy creation of the IOPIS/NSGA-III and IOPIS/RVEA implementations.

The results obtained in the numerical experiments were very promising. The new interactive EAs outperformed standalone NSGA-III and RVEA, as well as their interactive versions. The solutions obtained by the IOPIS EAs were closer to the Pareto optimal front, more preferable based on the reference point and spent less computational resources in the form of function evaluations. Further study of the landscape of the PIS is needed. The effect of choosing different achievement functions, their implications on the interaction mechanism by a DM and the solutions returned by the algorithm also needs to be studied.

**Acknowledgements.** This research was supported by the Academy of Finland (grant numbers 322221 and 311877). The research is related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization, [juu.fi/demo](http://juu.fi/demo)) of the University of Jyväskylä.

## References

1. Bechikh, S., Kessentini, M., Said, L.B., Ghédira, K.: Preference incorporation in evolutionary multiobjective optimization: a survey of the state-of-the-art. In: Hurson, A.R. (ed.) *Advances in Computers*, vol. 98, pp. 141–207. Elsevier (2015). (Chapter four)

2. Buchanan, J.T.: A naïve approach for solving MCDM problems: the GUESS method. *J. Oper. Res. Soc.* **48**(2), 202–206 (1997)
3. Buchanan, J., Gardiner, L.: A comparison of two reference point methods in multiple objective mathematical programming. *Eur. J. Oper. Res.* **149**(1), 17–34 (2003)
4. 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)
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 multi-objective optimization test problems. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002), pp. 825–830. IEEE (2002)
8. Deb, K., Miettinen, K.: Nadir point estimation using evolutionary approaches: better accuracy and computational speed through focused search. In: Ehrgott, M., Naujoks, B., Stewart, T.J., Wallenius, J. (eds.) *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*. LNE, vol. 634, pp. 339–354. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-04045-0\\_29](https://doi.org/10.1007/978-3-642-04045-0_29)
9. Deb, K., Saxena, D.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: Proceedings of the World Congress on Computational Intelligence (WCCI 2006), pp. 3352–3360 (2006)
10. Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. In: GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 635–642. ACM, New York (2006)
11. Eskelinen, P., Miettinen, K., Klamroth, K., Hakanen, J.: Pareto navigator for interactive nonlinear multiobjective optimization. *OR Spectrum* **32**(1), 211–227 (2010). <https://doi.org/10.1007/s00291-008-0151-6>
12. Hakanen, J., Chugh, T., Sindhya, K., Jin, Y., Miettinen, K.: Connections of reference vectors and different types of preference information in interactive multi-objective evolutionary algorithms. In: Proceeding of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8 (2016)
13. 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)
14. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: a short review. In: Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 2419–2426 (2008)
15. Li, K., Chen, R., Min, G., Yao, X.: Integration of preferences in decomposition multiobjective optimization. *IEEE Trans. Cybern.* **48**(12), 3359–3370 (2018)
16. Luque, M., Ruiz, F., Miettinen, K.: Global formulation for interactive multiobjective optimization. *OR Spectrum* **33**(1), 27–48 (2011). <https://doi.org/10.1007/s00291-008-0154-3>
17. Meignan, D., Knust, S., Frayret, J.M., Pesant, G., Gaud, N.: A review and taxonomy of interactive optimization methods in operations research. *ACM Trans. Interact. Intell. Syst.* **5**(3), 1–43 (2015)

18. Miettinen, K.: *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston (1999). <https://doi.org/10.1007/978-1-4615-5563-6>
19. Miettinen, K., Hakanen, J., Podkopaev, D.: Interactive nonlinear multiobjective optimization methods. In: Greco, S., Ehrgott, M., Figueira, J.R. (eds.) *Multiple Criteria Decision Analysis*. ISORMS, vol. 233, pp. 927–976. 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)
20. Miettinen, K., Mäkelä, M.M.: On scalarizing functions in multiobjective optimization. *OR Spectrum* **24**(2), 193–213 (2002). <https://doi.org/10.1007/s00291-001-0092-9>
21. Miettinen, K., Mäkelä, M.M.: Synchronous approach in interactive multiobjective optimization. *Eur. J. Oper. Res.* **170**(3), 909–922 (2006)
22. Nakayama, H., Sawaragi, Y.: Satisficing trade-off method for multiobjective programming. In: Grauer, M., Wierzbicki, A.P. (eds.) *Interactive Decision Analysis*. LNE, vol. 229, pp. 113–122. Springer, Heidelberg (1984). [https://doi.org/10.1007/978-3-662-00184-4\\_13](https://doi.org/10.1007/978-3-662-00184-4_13)
23. 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)
24. Ruiz, F., Luque, M., Miettinen, K.: Improving the computational efficiency in a global formulation (GLIDE) for interactive multiobjective optimization. *Ann. Oper. Res.* **197**(1), 47–70 (2012). <https://doi.org/10.1007/s10479-010-0831-x>
25. Sawaragi, Y., Nakayama, H., Tanino, T.: *Theory of Multiobjective Optimization*. Elsevier, Amsterdam (1985)
26. Thiele, L., Miettinen, K., Korhonen, P.J., Molina, J.: A preference-based evolutionary algorithm for multi-objective optimization. *Evol. Comput.* **17**(3), 411–436 (2009)
27. Vesikar, Y., Deb, K., Blank, J.: Reference point based NSGA-III for preferred solutions. In: *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1587–1594 (2018)
28. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) *Multiple Criteria Decision Making Theory and Application*. LNE, vol. 177, pp. 468–486. Springer, Heidelberg (1980). [https://doi.org/10.1007/978-3-642-48782-8\\_32](https://doi.org/10.1007/978-3-642-48782-8_32)
29. Wierzbicki, A.P.: A mathematical basis for satisficing decision making. *Math. Model.* **3**(5), 391–405 (1982)
30. Xin, B., Chen, L., Chen, J., Ishibuchi, H., Hirota, K., Liu, B.: Interactive multiobjective optimization: a review of the state-of-the-art. *IEEE Access* **6**, 41256–41279 (2018)
31. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
32. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. *Evol. Comput.* **8**(2), 173–195 (2000)



# Hypervolume Optimal $\mu$ -Distributions on Line-Based Pareto Fronts in Three Dimensions

Ke Shang, Hisao Ishibuchi<sup>(✉)</sup>, Weiyu Chen, and Lukáš Adam

Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation,  
Department of Computer Science and Engineering,  
Southern University of Science and Technology, Shenzhen 518055, China  
kshang@foxmail.com, hisao@sustech.edu.cn,  
11711904@mail.sustech.edu.cn, adam@utia.cas.cz

**Abstract.** Hypervolume optimal  $\mu$ -distribution is a fundamental research topic which investigates the distribution of  $\mu$  solutions on the Pareto front for hypervolume maximization. It has been theoretically shown that the optimal distribution of  $\mu$  solutions on a linear Pareto front in two dimensions is the one with  $\mu$  equispaced solutions. However, the equispaced property of an optimal distribution does not always hold for a single-line Pareto front in three dimensions. It only holds for the single-line Pareto front where one objective of the Pareto front is constant. In this paper, we further theoretically investigate the hypervolume optimal  $\mu$ -distribution on line-based Pareto fronts in three dimensions. In addition to a single-line Pareto front, we consider Pareto fronts constructed with two lines and three lines, where each line is a Pareto front with one constant objective. We show that even the equispaced property holds for each single-line Pareto front, it does not always hold for the Pareto fronts combined with them. Specifically, whether this property holds or not depends on how the lines are combined.

**Keywords:** Hypervolume indicator · Evolutionary multi-objective optimization · Optimal  $\mu$ -distribution

## 1 Introduction

The hypervolume indicator is a widely-used performance indicator in the evolutionary multi-objective optimization (EMO) field. It is well recognized that the hypervolume indicator can evaluate the convergence and diversity of a solution set simultaneously. The most important property is that it is the only Pareto compliant indicator [22]. Due to this unique property, it is guaranteed that the solutions which maximize the hypervolume indicator for a given problem are all Pareto optimal [10]. Therefore, the hypervolume indicator can be used in EMO algorithms (EMOAs) to push the population towards the Pareto front. These kinds of EMOAs are usually called hypervolume-based EMOAs. Their representatives are SMS-EMOA [5, 8], FV-MOEA [18], HypE [4], and R2HCA-EMOA [19].

The goal of the hypervolume-based EMOAs is to obtain a well-converged and widely-distributed solution set for hypervolume maximization. Thus, it is important to understand whether a widely-distributed solution set can be obtained by maximizing the hypervolume indicator. This research issue is called the hypervolume optimal  $\mu$ -distribution, which investigates the distribution of  $\mu$  solutions on the Pareto front for hypervolume maximization.

It has been theoretically proved that the hypervolume optimal  $\mu$ -distribution on a linear Pareto front in two dimensions is the one with  $\mu$  equispaced solutions [2,9]. This is the only known theoretical description of how  $\mu$  solutions are distributed on the Pareto front for hypervolume maximization in two dimensions. Even though Auger *et al.* [2] derived the optimal density of solutions on a continuous and differentiable nonlinear Pareto front in two dimensions for an infinitely large number of solutions, it is difficult (presumably impossible) to precisely determine the hypervolume optimal  $\mu$ -distribution on nonlinear Pareto fronts. In three dimensions, Shukla *et al.* [20] theoretically derived the hypervolume optimal  $\mu$ -distribution on a single-line Pareto front, which shows that the equispaced property of an optimal distribution for a line in two dimensions does not always hold in higher dimensions. This property only holds for the single-line Pareto front where one objective is constant. Auger *et al.* [1] also theoretically investigated the hypervolume optimal  $\mu$ -distribution in three dimensions. However, things become complicated in three dimensions and the exact hypervolume optimal  $\mu$ -distribution is not obtained in [1]. Thus, the hypervolume optimal  $\mu$ -distribution in three dimensions is mainly investigated empirically [12,14–16,21], i.e., the hypervolume optimal  $\mu$ -distribution is approximated.

In this paper, we follow the path in [20] and push forward the theoretical research on the hypervolume optimal  $\mu$ -distribution in three dimensions. In particular, we focus on the line-based Pareto fronts constructed with two and three lines where each line has a constant objective. Counter-intuitively, even the equispaced property holds for each single-line Pareto front, it does not always hold for the Pareto fronts combined with them. Specifically, the validity of this property depends on how the lines are combined.

The rest of the paper is organized as follows. The preliminaries of the paper is provided in Sect. 2. In Sect. 3, two-line Pareto fronts are investigated. In Sect. 4, three-line Pareto fronts are investigated. Finally, the paper is concluded in Sect. 5.

## 2 Preliminaries

### 2.1 Hypervolume Indicator and Its Optimal $\mu$ -Distribution

Formally, the hypervolume indicator is defined as follows. Given a solution set  $A \subset \mathbb{R}^m$  and a reference point  $\mathbf{r} \in \mathbb{R}^m$ , the hypervolume of the solution set  $A$  is defined as

$$\text{HV}(A, \mathbf{r}) = \mathcal{L} \left( \bigcup_{\mathbf{a} \in A} \{\mathbf{b} | \mathbf{a} \succeq \mathbf{b} \succeq \mathbf{r}\} \right), \quad (1)$$

where  $\mathcal{L}(\cdot)$  denotes the Lebesgue measure of a set, and  $\mathbf{a} \succeq \mathbf{b}$  means that  $\mathbf{a}$  Pareto dominates  $\mathbf{b}$  (i.e.,  $a_i \geq b_i$  for all  $i = 1, \dots, m$  and  $a_j > b_j$  for at least one  $j = 1, \dots, m$  in the maximization case).

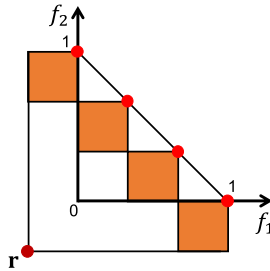
The hypervolume optimal  $\mu$ -distribution is defined as follows [3]. Given a reference point  $\mathbf{r} \in \mathbb{R}^m$  and a Pareto front  $\mathcal{F} \subset \mathbb{R}^m$ , the hypervolume optimal  $\mu$ -distribution is a set of  $\mu \in \mathbb{N}$  points on the Pareto front where the hypervolume of the  $\mu$  points is maximized. The set  $A$  containing the optimal  $\mu$  points satisfies

$$A = \arg \max_{|A'|=\mu, A' \subset \mathcal{F}} \text{HV}(A', \mathbf{r}). \tag{2}$$

### 2.2 Hypervolume Optimal $\mu$ -Distribution in Two Dimensions

As theoretically proved in [2, 9], the optimal  $\mu$ -distribution for hypervolume maximization is the one with  $\mu$  equispaced solutions on a linear Pareto front in two dimensions. Let us consider the simplest linear Pareto front  $f_1 + f_2 = 1$  and  $f_1, f_2 \geq 0$ . In order to include the two extreme points  $(0, 1)$  and  $(1, 0)$  of the Pareto front in the hypervolume optimal  $\mu$ -distribution, the reference point  $\mathbf{r} = (r, r)$  should satisfy  $r \leq -\frac{1}{\mu-1}$  as proved by Brockhoff [6]<sup>1</sup>.

Figure 1 illustrates the hypervolume optimal  $\mu$ -distribution in two dimensions, where the reference point is specified as  $r = -\frac{1}{\mu-1}$ . In this case, the two extreme solutions are included and all solutions have the same hypervolume contribution (i.e., the colored square) as shown in Fig. 1.



**Fig. 1.** An illustration of the hypervolume optimal  $\mu$ -distribution on a linear Pareto front in two dimensions. In this example,  $\mu = 4$  and  $\mathbf{r} = (-1/3, -1/3)$ . In the hypervolume optimal  $\mu$ -distribution, the four solutions (red points) are equispaced on the Pareto front. (Color figure online)

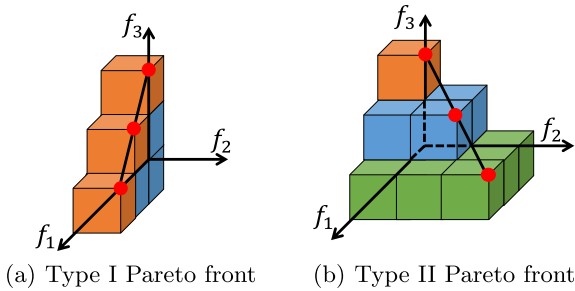
<sup>1</sup> In this paper, maximization of each objective is assumed in multi-objective optimization problems. In the case of minimization, this condition is rewritten as  $r \geq 1 + 1/(\mu - 1)$ .

### 2.3 Hypervolume Optimal $\mu$ -Distribution in Three Dimensions

Throughout of this paper, we consider Pareto fronts in the normalized objective space  $[0, 1]^3$  and the reference point  $\mathbf{r} = (r, r, r)$  (i.e., each element is the same).

In [20], the hypervolume optimal  $\mu$ -distribution on a single-line Pareto front in three dimensions is theoretically investigated. In three dimensions, single-line Pareto fronts can be categorized into the following two types:

1. **Type I:** Two objectives are conflicting with each other, and the other objective is constant. For example,  $f_1$  and  $f_3$  are conflicting with each other, and  $f_2$  has a constant value (see Fig. 2(a)).
2. **Type II:** Two objectives are consistent with each other, and the other objective is conflicting with the two objectives. For example,  $f_1$  and  $f_2$  are consistent, and  $f_3$  is conflicting with  $f_1$  and  $f_2$  (see Fig. 2(b)).



**Fig. 2.** Single-line Pareto fronts in three dimensions.

As proved in [20], the hypervolume optimal  $\mu$ -distribution is only equispaced on Type I Pareto front, whereas it is not equispaced on Type II Pareto front. This indicates that the optimal distribution theory on the linear Pareto front in two dimensions cannot be generalized to three dimensions. As illustrated in Fig. 2(a), for Type I Pareto front  $f_1 + f_3 = 1, f_1, f_3 \geq 0, f_2 = 0$  and  $\mathbf{r} = (r, r, r)$  where  $r = -\frac{1}{\mu-1}$ , the equispaced solutions have the same hypervolume contribution. We can see that the hypervolume of the three solutions equals to the two-dimensional hypervolume of the three solutions in  $f_1$ - $f_3$  space times  $|r|$ . Thus, in this situation, hypervolume maximization in three dimensions can be treated as hypervolume maximization in two dimensions. However, for Type II Pareto front  $f_1 + f_3 = 1, f_1, f_3 \geq 0, f_1 = f_2$  and  $\mathbf{r} = (r, r, r)$  where  $r = -\frac{1}{\mu-1}$  in Fig. 2(b), the equispaced solutions have different hypervolume contributions. The lower solution (with the smaller value of  $f_3$ ) has a larger hypervolume contribution than the upper solution. The whole hypervolume can be increased by slightly moving the middle solution towards the lower solution, which means that the equispaced solutions are not optimal for hypervolume maximization of  $\mu$  solutions.



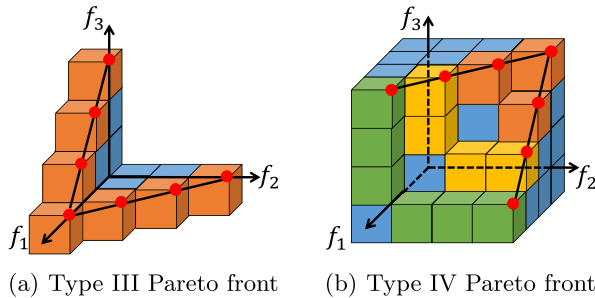
In the rest of this paper, we focus on Type I Pareto front and extend it to Pareto fronts which consist of two and three joint Type I Pareto fronts. We investigate whether the equispaced property of the hypervolume optimal  $\mu$ -distribution still holds or not on these extended Pareto fronts.

### 3 Two-Line Pareto Fronts

In this section, we extend the Type I Pareto front, and investigate the hypervolume optimal  $\mu$ -distribution on a Pareto front which consists of two joint Type I Pareto fronts.

For such a two-line Pareto front, there are two different types as follows:

1. **Type III:** The two lines lie on a triangular front. For example, the first line is  $f_1 + f_2 = 1, f_1, f_2 \geq 0$  and  $f_3 = 0$ , and the second line is  $f_1 + f_3 = 1, f_1, f_3 \geq 0$  and  $f_2 = 0$ , both lying on the triangular front  $f_1 + f_2 + f_3 = 1, f_1, f_2, f_3 \geq 0$  (see Fig. 3(a)).
2. **Type IV:** The two lines lie on an inverted triangular front. For example, the first line is  $f_1 + f_2 = 1, f_1, f_2 \geq 0$  and  $f_3 = 1$ , and the second line is  $f_1 + f_3 = 1, f_1, f_3 \geq 0$  and  $f_2 = 1$ , both lying on the inverted triangular front  $f_1 + f_2 + f_3 = 2, 0 \leq f_1, f_2, f_3 \leq 1$  (see Fig. 3(b)).



**Fig. 3.** Two-line Pareto fronts in three dimensions.

Next, we investigate the hypervolume optimal  $\mu$ -distribution on each of these two types of Pareto fronts.

#### 3.1 Type III Pareto Front

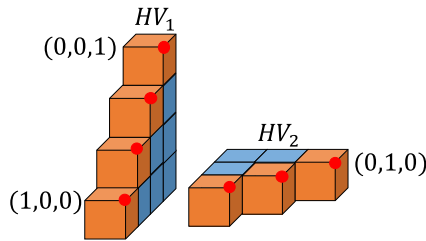
Let us consider the Pareto front with two lines where the first line is  $f_1 + f_3 = 1, f_1, f_3 \geq 0$  and  $f_2 = 0$ , and the second line is  $f_1 + f_2 = 1, f_1, f_2 \geq 0$  and  $f_3 = 0$ . We will investigate whether the uniformly distributed solutions on the Type III Pareto front are optimal or not for hypervolume maximization of  $\mu$  solutions. Here, the uniformity means that the same number of solutions are uniformly

distributed on each line. We assume that the three extreme solutions  $(1,0,0)$ ,  $(0,1,0)$  and  $(0,0,1)$  are included in the optimal  $\mu$ -distribution (i.e., all extreme points of the two lines are included in the optimal  $\mu$ -distribution). Figure 3(a) gives an illustration of the uniform solution set on the Type III Pareto front.

Given the reference point  $\mathbf{r} = (r, r, r)$ , the following theorem tells us that Fig. 3(a) is indeed optimal for hypervolume maximization of  $\mu$  solutions.

**Theorem 1.** *For  $\mu > 3$  and  $\mu$  is odd, if  $r \leq -\frac{2}{\mu-1}$ , the hypervolume optimal  $\mu$ -distribution on the Type III Pareto front is the one with  $\frac{\mu+1}{2}$  solutions uniformly distributed on each of the two lines, where the three extreme points (i.e.,  $(1,0,0)$ ,  $(0,1,0)$ ,  $(0,0,1)$ ) are included in the optimal  $\mu$ -distribution.*

*Proof.* Suppose there are  $\mu_1$  solutions on the first line and  $\mu_2$  solutions on the second line where  $\mu_1 + \mu_2 = \mu + 1$  (since  $(1,0,0)$  belongs to both lines). We can divide the whole hypervolume into two parts as illustrated in Fig. 4. The first part (denoted as  $HV_1$ ) is the hypervolume determined by the solutions on the first line with the reference point  $\mathbf{r}_1 = (r, r, r)$ . The second part (denoted as  $HV_2$ ) is the hypervolume determined by the solutions on the second line with the reference point  $\mathbf{r}_2 = (r, 0, r)$ .



**Fig. 4.** An illustration of the decomposition of the hypervolume into two parts.

After the division, the hypervolume can be calculated as  $HV = HV_1 + HV_2$ . Thus, maximizing  $HV$  is equivalent to maximizing  $HV_1 + HV_2$ . We can see that  $HV_1$  is the two-dimensional hypervolume in the  $f_1$ - $f_3$  space (denoted as  $HV_1^{f_1-f_3}$ ) times  $|r|$ , and  $HV_2$  is the two-dimensional hypervolume in the  $f_1$ - $f_2$  space (denoted as  $HV_2^{f_1-f_2}$ ) times  $|r|$ . Therefore, maximizing  $HV_1 + HV_2$  is equivalent to maximizing  $HV_1^{f_1-f_3} + HV_2^{f_1-f_2}$ .

Based on the theory for the hypervolume optimal  $\mu$ -distribution in two dimensions [2], we can conclude that the  $\mu_1$  solutions are equally spaced on the first line and the  $\mu_2$  solutions are equally spaced on the second line in order to maximize  $HV_1^{f_1-f_3}$  and  $HV_2^{f_1-f_2}$ , respectively.

Now the question is the relation between  $\mu_1$  and  $\mu_2$  when there are  $\mu$  solutions in total. Given  $\mu_1$  uniform points on the first line and  $\mu_2$  uniform points on the second line where  $\mu_1 + \mu_2 = \mu + 1$  (since  $(1,0,0)$  is on both lines),  $HV_1^{f_1-f_3}$  and

$HV_2^{f_1-f_2}$  can be calculated as follows:

$$\begin{aligned} HV_1^{f_1-f_3} &= \frac{1}{2} - \frac{1}{2(\mu_1 - 1)} - r(2 - r), \\ HV_2^{f_1-f_2} &= \frac{1}{2} - \frac{1}{2(\mu_2 - 1)} - r. \end{aligned} \tag{3}$$

Then we need to solve the following optimization problem:

$$\begin{aligned} \text{maximize: } & HV_1^{f_1-f_3} + HV_2^{f_1-f_2} \\ &= 1 - \frac{1}{2(\mu_1 - 1)} - \frac{1}{2(\mu_2 - 1)} - r(3 - r), \\ \text{subject to: } & \mu_1 + \mu_2 = \mu + 1 \text{ and } \mu_1, \mu_2 \in \mathbb{Z}_+. \end{aligned} \tag{4}$$

It is not difficult to solve the above optimization problem by relaxing  $\mu_1$  and  $\mu_2$  to real values. The optimal values of  $\mu_1$  and  $\mu_2$  are as follows:

$$\mu_1 = \mu_2 = \frac{\mu + 1}{2}. \tag{5}$$

Since  $\mu$  is assumed to be odd, Eq. (5) is the optimal solution of (4). Finally, from the analysis about the reference point specification in two dimensions, it is easy to obtain that the reference point should satisfy  $r \leq -\frac{2}{\mu-1}$  in order to include the three extreme solutions in the optimal  $\mu$ -distribution.  $\square$

If the reference point is specified as  $r = -\frac{2}{\mu-1}$ , then all the uniformly distributed solutions have the same hypervolume contribution as shown in Fig. 3(a). Here we need to note that Theorem 1 only holds when  $\mathbf{r} = (r, r, r)$ , i.e., each element of  $\mathbf{r}$  is the same. For example, when  $\mathbf{r} = (r, r', r)$  where  $r' \ll r$ , Theorem 1 will not hold since the solutions on the first line will have much larger hypervolume contribution than the solutions on the second line. We can expect that more solutions will distribute on the first line in this situation. Since we only consider  $\mathbf{r} = (r, r, r)$  as stated in Sect. 2.3, we do not discuss this issue much in this paper.

### 3.2 Type IV Pareto Front

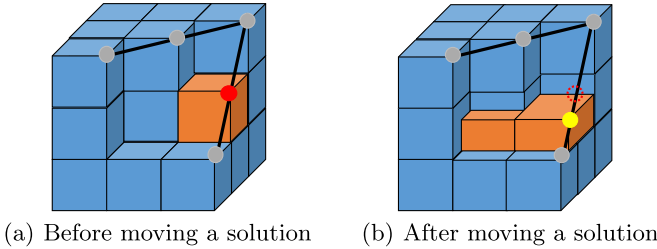
Let us consider the Pareto front with two lines where the first line is  $f_1 + f_2 = 1, f_1, f_2 \geq 0$  and  $f_3 = 1$ , and the second line is  $f_1 + f_3 = 1, f_1, f_3 \geq 0$  and  $f_2 = 1$ . Similar to the previous subsection, we investigate whether the uniformly distributed solutions on the Type IV Pareto front are optimal or not for hypervolume maximization of  $\mu$  solutions. We assume that the same number of solutions are uniformly distributed on each line, and the three extreme solutions (1,0,1), (0,1,1) and (1,1,0) are included in the optimal  $\mu$ -distributions. Figure 3(b) gives an illustration of the uniform solution set on the Type IV Pareto front.

Given the reference point  $\mathbf{r} = (r, r, r)$ , the following theorem tells us that Fig. 3(b) is not optimal for hypervolume maximization of  $\mu$  solutions.

**Theorem 2.** For  $\mu > 3$  and  $\mu$  is odd, the uniformly distributed solutions (including the three extreme solutions) on the Type IV Pareto front are not optimal for hypervolume maximization of  $\mu$  solutions.

*Proof.* Suppose there are  $\mu'$  solutions uniformly distributed on each of the two lines, then  $\mu' = \frac{\mu+1}{2}$ . Let  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_{\mu'}$  denote the solutions on the line from  $(0, 1, 1)$  to  $(1, 1, 0)$  where  $\mathbf{a}_1 = (0, 1, 1)$  and  $\mathbf{a}_{\mu'} = (1, 1, 0)$  are the two extreme solutions of this line. We assume that  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_{\mu'}$  are on the line in this order. Figure 3(b) illustrates the hypervolume of the uniformly distributed solutions on the Type IV Pareto front. We can observe that different solutions have different hypervolume contributions. The hypervolume contribution of  $\mathbf{a}_i$  can be calculated as follows based on the observation in Fig. 3(b):

$$HVC(\mathbf{a}_i) = (i - 1) \left( \frac{1}{\mu' - 1} \right)^3, \quad i = 2, \dots, \mu' - 1. \tag{6}$$



**Fig. 5.** An illustration of moving a solution.

Now if we move  $\mathbf{a}_i$  towards  $\mathbf{a}_{i+1}$  for any  $i \in \{2, \dots, \mu' - 1\}$  (i.e., only one solution is moved and all the others are fixed as illustrated in Fig. 5) to obtain a new solution  $\mathbf{a}'_i = \mathbf{a}_i + \alpha(\mathbf{a}_{i+1} - \mathbf{a}_i)$  where  $\alpha \in [0, 1]$ , the hypervolume contribution of  $\mathbf{a}'_i$  is

$$HVC(\mathbf{a}'_i) = [(i - 1)(1 - \alpha)(1 + \alpha) + \alpha(1 - \alpha)] \left( \frac{1}{\mu' - 1} \right)^3. \tag{7}$$

The difference between  $HVC(\mathbf{a}'_i)$  and  $HVC(\mathbf{a}_i)$  is

$$HVC(\mathbf{a}'_i) - HVC(\mathbf{a}_i) = (-i\alpha^2 + \alpha) \left( \frac{1}{\mu' - 1} \right)^3. \tag{8}$$

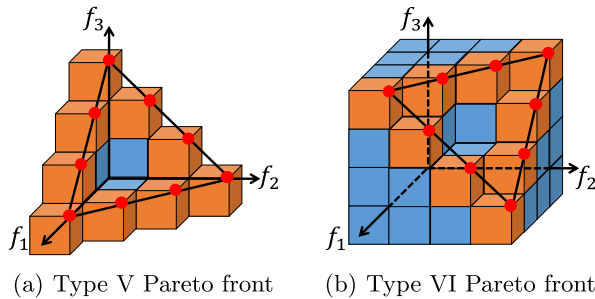
It is easy to obtain that if  $0 < \alpha < \frac{1}{i}$ ,  $HVC(\mathbf{a}'_i) - HVC(\mathbf{a}_i) > 0$ , which means that the hypervolume contribution of  $\mathbf{a}'_i$  is larger than that of  $\mathbf{a}_i$ . Thus, the overall hypervolume can be improved by moving  $\mathbf{a}_i$  to  $\mathbf{a}'_i$ . We can conclude that the original uniformly distributed solution set is not optimal for hypervolume maximization of  $\mu$  solutions. □

In Theorem 2, we only proved that the uniform solution set on the Type IV Pareto front is not optimal for hypervolume maximization of  $\mu$  solutions. We leave it as an open question to exactly describe the hypervolume optimal  $\mu$ -distribution on the Type IV Pareto front. Also, we explicitly assume that the three extreme solutions are included whereas this depends on the reference point specification. It seems from our empirical investigation that all the extreme points are included in the optimal  $\mu$ -distribution if the reference point is sufficiently far away from the Pareto front.

### 4 Three-Line Pareto Fronts

In the previous section, we investigated the Pareto fronts with two joint lines and showed that the hypervolume optimal  $\mu$ -distribution is uniform on the Type III Pareto front and nonuniform on the Type IV Pareto front. In this section, we examine Pareto fronts with three Type I lines. Two types of Pareto fronts with three Type I lines are considered as follows:

1. **Type V:** The three lines lie on the triangular front. For example, the three lines are the three edges of the triangular front  $f_1 + f_2 + f_3 = 1, f_1, f_2, f_3 \geq 0$  (see Fig. 6(a)).
2. **Type VI:** The three lines lie on the inverted triangular front. For example, the three lines are the three edges of the inverted triangular front  $f_1 + f_2 + f_3 = 2, 0 \leq f_1, f_2, f_3 \leq 1$  (see Fig. 6(b)).



**Fig. 6.** Three-line Pareto fronts in three dimensions.

Next, the hypervolume optimal  $\mu$ -distributions on these two types of Pareto fronts are investigated.

#### 4.1 Type V Pareto Front

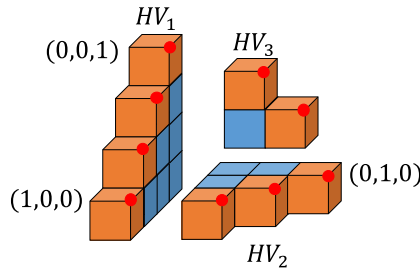
Let us consider the three-line Pareto front with a triangular shape specified by  $f_1 + f_2 + f_3 = 1, f_1, f_2, f_3 \geq 0$ . We investigate whether the uniformly distributed

solutions on the Type V Pareto front are optimal or not for hypervolume maximization of  $\mu$  solutions. Here, the uniformity means that the same number of solutions are uniformly distributed on each line, and the three extreme solutions (i.e.,  $(1,0,0)$ ,  $(0,1,0)$ ,  $(0,0,1)$ ) are included in the optimal  $\mu$ -distribution. Figure 6(a) gives an illustration of the uniformly distributed solution set on the Type V Pareto front.

Given the reference point  $\mathbf{r} = (r, r, r)$ , the following theorem tells us that Fig. 6(a) is optimal for hypervolume maximization of  $\mu$  solutions.

**Theorem 3.** *For  $\mu > 3$  and  $\mu$  is a multiple of three, if  $r \leq -\frac{3}{\mu}$ , the hypervolume optimal  $\mu$ -distribution on the Type V Pareto front is that  $\frac{\mu+3}{3}$  solutions are uniformly distributed on each of the three lines, where the three extreme points (i.e.,  $(1,0,0)$ ,  $(0,1,0)$ ,  $(0,0,1)$ ) are included.*

*Proof.* The proof is similar to that of Theorem 1. We divide the hypervolume into three parts ( $HV_1$ ,  $HV_2$ , and  $HV_3$ ) as illustrated in Fig. 7. Then the hypervolume can be calculated as  $HV = HV_1 + HV_2 + HV_3$ . Similarly, maximizing  $HV_1 + HV_2 + HV_3$  is equivalent to maximizing  $HV_1^{f_1-f_3} + HV_2^{f_1-f_2} + HV_3^{f_2-f_3}$ .



**Fig. 7.** An illustration of the decomposition of the hypervolume into three parts.

Given  $\mu_1$ ,  $\mu_2$  and  $\mu_3$  solutions on each of the three lines where  $\mu_1 + \mu_2 + \mu_3 = \mu + 3$  (since each extreme solution belongs to two lines),  $HV_1^{f_1-f_3}$ ,  $HV_2^{f_1-f_2}$  and  $HV_3^{f_2-f_3}$  can be calculated as follows:

$$\begin{aligned}
 HV_1^{f_1-f_3} &= \frac{1}{2} - \frac{1}{2(\mu_1 - 1)} - r(2 - r), \\
 HV_2^{f_1-f_2} &= \frac{1}{2} - \frac{1}{2(\mu_2 - 1)} - r, \\
 HV_3^{f_2-f_3} &= \frac{1}{2} - \frac{1}{2(\mu_3 - 1)}.
 \end{aligned}
 \tag{9}$$

Then we need to solve the following optimization problem:

$$\begin{aligned} \text{maximize: } & HV_1^{f_1-f_3} + HV_2^{f_1-f_2} + HV_3^{f_2-f_3} \\ &= \frac{3}{2} - \frac{1}{2(\mu_1 - 1)} - \frac{1}{2(\mu_2 - 1)} - \frac{1}{2(\mu_3 - 1)} - r(3 - r), \quad (10) \\ \text{subject to: } & \mu_1 + \mu_2 + \mu_3 = \mu + 3 \text{ and } \mu_1, \mu_2, \mu_3 \in \mathbb{Z}_+. \end{aligned}$$

By relaxing  $\mu_1, \mu_2$  and  $\mu_3$  to real values, we can obtain the optimal values of  $\mu_1, \mu_2$  and  $\mu_3$  as follows:

$$\mu_1 = \mu_2 = \mu_3 = \frac{\mu + 3}{3}. \quad (11)$$

Since  $\mu$  is a multiple of three, Eq. (11) is the optimal solution of (10). Finally, from the analysis about the reference point specification in two dimensions, it is easy to show that the reference point should satisfy  $r \leq -\frac{3}{\mu}$  in order to include the three extreme solutions in the optimal distribution.  $\square$

If the reference point is specified as  $r = -\frac{3}{\mu}$ , each of the uniformly distributed solutions has the same hypervolume contribution as illustrated in Fig. 6(a). Theorem 3 is also restricted to the case where each element of  $\mathbf{r}$  is the same, so that each line has equal importance to the overall hypervolume.

### 4.2 Type VI Pareto Front

Let us consider the three-line Pareto front with an inverted triangular shape specified by  $f_1 + f_2 + f_3 = 2, 0 \leq f_1, f_2, f_3 \leq 1$ . We investigate whether the uniformly distributed solutions on the Type VI Pareto front are optimal or not for hypervolume maximization of  $\mu$  solutions. We assume that the same number of solutions are distributed on each line, and the three extreme solutions (i.e., (1,0,1), (0,1,1), (1,1,0)) are included in the optimal distribution. Figure 6(b) illustrates the uniformly distributed solution set on the Type VI Pareto front.

Given the reference point  $\mathbf{r} = (r, r, r)$ , the following theorem tells that Fig. 6(b) is not optimal for hypervolume maximization of  $\mu$  solutions.

**Theorem 4.** *Given  $\mu > 6$  and  $\mu$  is a multiple of three, the uniformly distributed solutions (including the three extreme solutions) on the Type VI Pareto front are not optimal for hypervolume maximization of  $\mu$  solutions.*

*Proof.* The proof is similar to that of Theorem 2. Suppose there are  $\mu'$  solutions uniformly distributed on each of the three lines, then  $\mu' = \frac{\mu+3}{3}$ . Let  $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_{\mu'}$  denote the uniformly distributed solutions in this order on the line from (0, 1, 1) to (1, 1, 0) where  $\mathbf{a}_1 = (0, 1, 1)$  and  $\mathbf{a}_{\mu'} = (1, 1, 0)$  are the two extreme solutions of this line. Figure 6(b) illustrates the hypervolume of the uniformly distributed solutions on the Type VI Pareto front. Then the hypervolume contribution of  $\mathbf{a}_i$

can be calculated as follows<sup>2</sup>:

$$HVC(\mathbf{a}_i) = HVC(\mathbf{a}_{\mu'+1-i}) = (i - 1) \left( \frac{1}{\mu' - 1} \right)^3, \quad i = 2, \dots, \left\lfloor \frac{\mu'}{2} \right\rfloor + 1. \quad (12)$$

If we move  $\mathbf{a}_i$  towards  $\mathbf{a}_{i+1}$  for any  $i \in \{2, \dots, \lfloor \frac{\mu'}{2} \rfloor\}$  (i.e., only one solution is moved and all the others are fixed, which is similar to the case in Fig. 5) to obtain a new solution  $\mathbf{a}'_i = \mathbf{a}_i + \alpha(\mathbf{a}_{i+1} - \mathbf{a}_i)$  where  $\alpha \in [0, 1]$ , we can improve the overall hypervolume if  $0 < \alpha < \frac{1}{i}$ . We can conclude that the original uniformly distributed solution set is not optimal for hypervolume maximization of  $\mu$  solutions.  $\square$

In Theorem 4, we only show that the uniformly distributed solution set on the Type VI Pareto front is not optimal for hypervolume maximization of  $\mu$  solutions. We leave it as an open question to exactly describe the hypervolume optimal  $\mu$ -distribution on the Type VI Pareto front. We also explicitly assume that the three extreme solutions are included in the optimal distribution. A sufficiently far away reference point guarantees this based on our empirical investigation.

We also need to emphasize that Theorem 4 only holds for  $\mu > 6$ . If  $\mu \leq 6$ , which means that at most three solutions lie on each line, we cannot use the method in Theorem 4 to perturb one solution to get a better overall hypervolume.

## 5 Conclusions

In this paper, we investigated the hypervolume optimal  $\mu$ -distributions theoretically on the line-based Pareto fronts in three dimensions. In addition to the single-line Pareto front, we considered two-line and three-line Pareto fronts. We demonstrated that the optimal distribution theory on the linear Pareto front in two dimensions cannot always be generalized to three dimensions. The hypervolume optimal  $\mu$ -distributions are only uniform on the Types I, III and V Pareto fronts, whereas they are nonuniform on the Types II, IV, and VI Pareto fronts. Our results can help EMO researchers to understand the hypervolume indicator more deeply. Uniformity cannot always be guaranteed by maximizing the hypervolume indicator, which suggests us to carefully use the hypervolume indicator in more than two dimensions.

In the future, the following research topics can be addressed based on our current work. 1) We did not prove the exact hypervolume optimal  $\mu$ -distributions on the Types IV and VI Pareto fronts. It will be interesting and challenging to derive the exact hypervolume optimal  $\mu$ -distribution for each of these Pareto fronts. The hypervolume Newton method [13] is promising for addressing this issue. 2) The Pareto fronts considered in this paper are very simple and not realistic. We will consider more realistic Pareto fronts (e.g., the Pareto fronts of

---

<sup>2</sup> Due to the page limitation, the derivation of Eq. (12) is omitted here. Equation (12) can be obtained by visually examining figures like Fig. 6(b) with a different number of solutions (e.g., five or six solutions on each line).



DTLZ1 [7] and minus-DTLZ1 [17] problems) in our future study. In fact, the Pareto fronts considered in this paper can be seen as the Pareto fronts of the subproblems of DTLZ1 and minus-DTLZ1 [11].

**Acknowledgments.** This work was supported by National Natural Science Foundation of China (Grant No. 61876075), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531), the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).


## References

1. Auger, A., Bader, J., Brockhoff, D.: Theoretically investigating optimal  $\mu$ -distributions for the hypervolume indicator: first results for three objectives. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 586–596. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_59](https://doi.org/10.1007/978-3-642-15844-5_59)
2. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: optimal  $\mu$ -distributions and the choice of the reference point. In: Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, pp. 87–102. ACM (2009)
3. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Hypervolume-based multiobjective optimization: theoretical foundations and practical implications. *Theor. Comput. Sci.* **425**, 75–103 (2012)
4. Bader, J., Zitzler, E.: HypE: an algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.* **19**(1), 45–76 (2011)
5. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur. J. Oper. Res.* **181**(3), 1653–1669 (2007)
6. Brockhoff, D.: Optimal  $\mu$ -distributions for the hypervolume indicator for problems with linear bi-objective fronts: exact and exhaustive results. In: Deb, K., et al. (eds.) 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)
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. Advanced Information and Knowledge Processing*, 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)
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. Emmerich, M., Deutz, A., Beume, N.: Gradient-based/evolutionary relay hybrid for computing Pareto front approximations maximizing the S-metric. In: Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.) HM 2007. LNCS, vol. 4771, pp. 140–156. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-75514-2\\_11](https://doi.org/10.1007/978-3-540-75514-2_11)

10. Fleischer, M.: The measure of Pareto optima applications to multi-objective meta-heuristics. 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. Gebken, B., Peitz, S., Dellnitz, M.: On the hierarchical structure of pareto critical sets. *J. Glob. Optim.* **73**(4), 891–913 (2019). <https://doi.org/10.1007/s10898-019-00737-6>
12. Glasmachers, T.: Optimized approximation sets for low-dimensional benchmark Pareto fronts. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) PPSN 2014. LNCS, vol. 8672, pp. 569–578. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10762-2\\_56](https://doi.org/10.1007/978-3-319-10762-2_56)
13. Hernández, V.A.S., Schütze, O., Wang, H., Deutz, A., Emmerich, M.: The set-based hypervolume Newton method for bi-objective optimization. *IEEE Trans. Cybern.* **50**(5), 2186–2196 (2018)
14. Ishibuchi, H., Imada, R., Masuyama, N., Nojima, Y.: Comparison of hypervolume, IGD and IGD<sup>+</sup> from the viewpoint of optimal distributions of solutions. In: Deb, K., et al. (eds.) EMO 2019. LNCS, vol. 11411, pp. 332–345. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-12598-1\\_27](https://doi.org/10.1007/978-3-030-12598-1_27)
15. 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 the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, pp. 95–110. ACM (2017)
16. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: Reference point specification in hypervolume calculation for fair comparison and efficient search. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 585–592. ACM (2017)
17. 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)
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**(10), 2202–2213 (2015)
19. Shang, K., Ishibuchi, H.: A new hypervolume-based evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* (Early Access) (2020)
20. Shukla, P.K., Doll, N., Schmeck, H.: A theoretical analysis of volume based Pareto front approximations. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 1415–1422 (2014)
21. Tanabe, R., Ishibuchi, H.: An analysis of quality indicators using approximated optimal distributions in a three-dimensional objective space. *IEEE Trans. Evol. Comput.* (Early Access) (2020)
22. 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)



# Adaptive Operator Selection Based on Dynamic Thompson Sampling for MOEA/D

Lei Sun<sup>1</sup> and Ke Li<sup>2</sup>(✉) 

<sup>1</sup> College of Computer Science and Engineering,

University of Electronic Science and Technology of China, Chengdu, China

<sup>2</sup> Department of Computer Science, University of Exeter, Exeter EX4 5DS, UK

k.li@exeter.ac.uk

**Abstract.** In evolutionary computation, different reproduction operators have various search dynamics. To strike a well balance between exploration and exploitation, it is attractive to have an adaptive operator selection (AOS) mechanism that automatically chooses the most appropriate operator on the fly according to the current status. This paper proposes a new AOS mechanism for multi-objective evolutionary algorithm based on decomposition (MOEA/D). More specifically, the AOS is formulated as a multi-armed bandit problem where the dynamic Thompson sampling (DYTS) is applied to adapt the bandit learning model, originally proposed with an assumption of a fixed award distribution, to a non-stationary setup. In particular, each arm of our bandit learning model represents a reproduction operator and is assigned with a prior reward distribution. The parameters of these reward distributions will be progressively updated according to the performance of its performance collected from the evolutionary process. When generating an offspring, an operator is chosen by sampling from those reward distribution according to the DYTS. Experimental results fully demonstrate the effectiveness and competitiveness of our proposed AOS mechanism compared with other four state-of-the-art MOEA/D variants.

**Keywords:** Dynamic Thompson sampling · Adaptive operator selection · Evolutionary computation · MOEA/D

## 1 Introduction

Multi-objective optimisation problems (MOPs) are ubiquitous in various scientific [3] and engineering domains [11]. Due to the conflicting nature of different objectives, there does not exist a global optimum that optimises all objectives

---

K. Li was supported by UKRI Future Leaders Fellowship (Grant No. MR/S017062/1) and Royal Society (Grant No. IEC/NSFC/170243).

© Springer Nature Switzerland AG 2020

T. Bäck et al. (Eds.): PPSN 2020, LNCS 12270, pp. 271–284, 2020.

[https://doi.org/10.1007/978-3-030-58115-2\\_19](https://doi.org/10.1007/978-3-030-58115-2_19)

simultaneously. Instead, multi-objective optimisation aims to find a set of trade-off alternatives, an improvement at one objective of which can lead to a degradation of at least one other objective, that approximate the Pareto-optimal front (PF).

Due to the population-based characteristics, evolutionary algorithm (EA) has been recognised as the major approach for multi-objective optimisation. Over the past three decades and beyond, a significant amount of efforts have been devoted to the development of evolutionary multi-objective optimisation (EMO) algorithms, e.g., fast non-dominated sorting genetic algorithm II (NSGA-II) [5], indicator-based EA (IBEA) [17] and multi-objective EA based on decomposition (MOEA/D) [13]. It is widely appreciated that the search behaviour of an EA largely depends on its reproduction operator(s). In particular, some operators are exploration-oriented and are good at exploring unknown regions in the search space; whilst the others are exploitation-oriented and mainly focus on exploiting the current superior regions. How to strike a balance between exploration and exploitation is a long standing topic in order to achieve an efficient and effective evolutionary search process.

Adaptive operator selection (AOS) is an emerging paradigm that aims to autonomously select the most appropriate reproduction operator according to the latest search dynamics. Generally speaking, an AOS paradigm consists of two major steps. One is credit assignment that gives an operator a reward according to its up to date performance; the other is decision-making that selects the ‘appropriate’ operator for the next stage according to the accumulated awards. A fundamental issue behind the AOS is an exploration versus exploitation (EvE) dilemma. One hopes to give more chances to operators with decent track records (exploitation), but also wants to explore poor operators in the future search (exploration), since an operator might perform significantly differently at different search stages. It is worth noting that credit assignment under a multi-objective setting is even more challenging due to the conflicting characteristics of different objectives. To address the EvE dilemma, Li et al. [9] initially transformed the AOS problem into a multi-armed bandits (MAB) problem and applied the classic upper confidence bound (UCB) algorithm [1] to implement an AOS paradigm in EMO. As for the difficulty of credit assignment in multi-objective optimisation, MOEA/D is used as the baseline given that it decomposes the original MOP into several single objective subproblems which facilitate the fitness evaluation.

However, one of the major concerns of using bandit learning for AOS is its stationary environment assumption used in the traditional MAB model. In other words, the probability distribution of the reward for pulling an arm is fixed *a priori* whereas the evolutionary search process is highly non-stationary. Bearing this consideration in mind, this paper proposes to use the dynamic Thompson sampling strategy [6] to address the EvE dilemma under a non-stationary environment. More specifically, in our AOS paradigm, each reproduction operator is regarded as an arm in a bandit game and is assigned with a prior reward distribution. During the evolutionary search process, the credit of each operator is

updated according to the fitness improvement achieved by using the corresponding operator along with the parameters associated with its reward distribution. During the decision-making step, a reproduction operator is selected by sampling from those reward distributions. To facilitate the fitness evaluation under a multi-objective setting, we carry on with the MOEA/D as the baseline and the end algorithm is denoted as MOEA/D-DYTS. From our experiments, we have witnessed the superior performance of MOEA/D-DYTS over other four state-of-the-art MOEA/D variants on 19 benchmark test problems.

The remainder of this paper is organised as follows. Section 2 provides some preliminary knowledge. Section 3 delineates the technical details of our proposed MOEA/D-DYTS step by step. The performance of MOEA/D-DYTS is validated in Sect. 4. At the end, Sect. 5 concludes this paper and shed some lights on future directions.

## 2 Preliminaries

In this section, we provide the preliminary knowledge, including some definitions related to multi-objective optimisation and the baseline algorithm, required in this paper.

### 2.1 Multiobjective Optimization Problems

Without loss of generality, the MOP considered in this paper is defined as:

$$\begin{aligned} & \text{minimize} \quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x})) \\ & \text{subject to} \quad \mathbf{x} \in \Omega \end{aligned}, \quad (1)$$

where  $\mathbf{x} = (x_1, \dots, x_n) \in \Omega$  is a decision variable vector,  $\Omega = \prod_{i=1}^n [l_i, u_i] \in \mathbb{R}^n$  is the decision space where  $l_i$  and  $u_i$  are the lower and upper bounds of the  $i$ -th dimension.  $\mathbf{F} : \Omega \rightarrow \mathbb{R}^m$  consists of  $m$  conflicting objective functions. Given two decision vectors  $\mathbf{x}_1$  and  $\mathbf{x}_2 \in \Omega$ ,  $\mathbf{x}_1$  is said to dominate  $\mathbf{x}_2$ , denoted as  $\mathbf{x}_1 \preceq \mathbf{x}_2$ , if and only if  $f_i(x_1) \leq f_i(x_2)$  for all  $i \in \{1, \dots, m\}$  and  $\mathbf{F}(\mathbf{x}_1) \neq \mathbf{F}(\mathbf{x}_2)$ . A solution  $\mathbf{x}^* \in \Omega$  is said to be Pareto-optimal when no other solution  $\mathbf{x} \in \Omega$  can dominate  $\mathbf{x}^*$ . The set of all Pareto-optimal solutions is called Pareto-optimal set (PS) whilst its image in the objective space is called the PF.

### 2.2 Baseline Algorithm

In this paper, we use the MOEA/D as the baseline EMO framework. The basic idea of MOEA/D is to decompose the original MOP into several subproblems, each of which is either as an aggregated scalarising function or simplified MOP. Thereafter, MOEA/D uses a population-based meta-heuristic to solve these subproblems in a collaborative manner. In particular, the widely used Tchebychff function is used to form the subproblem in this paper and it is defined as:

$$\min_{\mathbf{x} \in \Omega} g^{\text{tch}}(\mathbf{x} | \mathbf{w}, \mathbf{z}^*) = \max_{1 \leq i \leq m} \{|f_i(\mathbf{x}) - \mathbf{z}_i^*| / w_i\}, \quad (2)$$

where  $\mathbf{w} = (w_1, \dots, w_m)$  is a weight vector, evenly sampled from a canonical simplex, with  $w_i \geq 0$ ,  $i \in \{1, \dots, m\}$ , and  $\sum_{i=1}^m w_i = 1$ .  $\mathbf{z}^* = (z_1, \dots, z_m)$  is the ideal objective where  $z_i = \min_{\mathbf{x} \in PS} f_i(x)$ ,  $i \in \{1, \dots, m\}$ . From the above description, we can see that the quality of a solution  $\mathbf{x} \in \Omega$  can be evaluated by a subproblem  $g^{\text{tch}}(\mathbf{x}|\mathbf{w}, \mathbf{z}^*)$  which facilitates the credit assignment under a multi-objective setting.

Note that instead of using the vanilla MOEA/D, here we use its variant with a dynamic resource allocation scheme, dubbed as MOEA/D-DRA [14], as the baseline algorithm given its outstanding performance in CEC 2009 MOEA competition. Different from the vanilla MOEA/D where all subproblems are allocated with the same amount of the computational resources, MOEA/D-DRA dynamically selects some most promising subproblems to evolve according to their utilities defined as:

$$\pi^i = \begin{cases} 1 & \text{if } \Delta^i > 0.001 \\ \left(0.95 + 0.05 \times \frac{\Delta^i}{0.001}\right) \times \pi^i & \text{otherwise} \end{cases}, \quad (3)$$

where  $\Delta^i$  is the fitness improvement rates (FIR) of the objective function value in subproblem  $i$ , which is defined as:

$$\Delta^i = \frac{g^{\text{tch}}(\mathbf{x}_{t-\Delta t}^i|\mathbf{w}^i, \mathbf{z}^*) - g^{\text{tch}}(\mathbf{x}_t^i|\mathbf{w}^i, \mathbf{z}^*)}{g^{\text{tch}}(\mathbf{x}_{t-\Delta t}^i|\mathbf{w}^i, \mathbf{z}^*)}, \quad (4)$$

where  $t$  is the current generation, and  $\Delta t$  is the updating period. Interested readers are referred to [14] for more details of MOEA/D-DRA.

### 3 Proposed Algorithm

In this section, we delineate the implementation our proposed MOEA/D-DYTS. Specifically, we will start with the classical Bernoulli MAB problem and the vanilla Thompson sampling. Thereafter, we develop the definition of the dynamic Thompson sampling strategy which is the main crux of our AOS paradigm. At the end, we give the algorithmic implementation of MOEA/D-DYTS.

#### 3.1 Thompson Sampling

In probability theory and machine learning, the MAB problem considers optimally allocating a fixed set of limited resources among competing actions  $\mathcal{A} = \{a_1, \dots, a_k\}$  that finally maximises the expected return (or gain). In this paper, we consider a traditional Bernoulli MAB problem where the reward of conducting an action  $a_i$  is  $r_i \in \{0, 1\}$ ,  $i \in \{1, \dots, k\}$ . At the time step  $t$ , the probability of obtaining a reward of one by conducting the action  $a_i$  is  $\mathbb{P}(r_i = 1|a_i, \theta) = \theta_i$ . On the other hand,  $\mathbb{P}(r_i = 0|a_i, \theta) = 1 - \theta_i$  when obtaining a reward of none. In particular,  $\theta_i$  is defined as the mean reward of action  $a_i$ ,  $i \in \{1, \dots, k\}$ . Under the MAB setting, the mean rewards for all actions

$\theta = (\theta_1, \dots, \theta_k)$  are fixed over time but are unknown beforehand. The ultimate goal of this Bernoulli MAB problem is to maximise the cumulative rewards over a period of  $T$  time steps.

Thompson sampling [12] is an efficient algorithm for solving the Bernoulli MAB problem. It takes advantage of Bayesian estimation for online decision problems where actions are applied in a sequential manner. In Thompson sampling, each arm is assigned with an independent prior reward distribution of  $\theta_i, i \in \{1, \dots, k\}$ . In particular, we use the Beta distribution with parameters  $\alpha = (\alpha_1, \dots, \alpha_k)$  and  $\beta = (\beta_1, \dots, \beta_k)$  to represent this prior distribution:

$$\mathcal{P}^{Beta}(\theta_i) = \frac{\Gamma(\alpha_i + \beta_i)}{\Gamma(\alpha_i)\Gamma(\beta_i)} \theta_i^{\alpha_i-1} (1 - \theta_i)^{\beta_i-1}, \tag{5}$$

where  $\alpha_i$  and  $\beta_i$  are the parameters associated with the  $i$ -th arm,  $i \in \{1, \dots, k\}$  and  $\Gamma(\cdot)$  is the gamma function. The parameters of this distribution are updated according to the Bayes' rule after receiving the up to date rewards. Due to the conjugate properties between Beta distribution and Bernoulli distribution, the posterior distribution of each action is still a Beta distribution. For a given action  $a_i, i \in \{1, \dots, k\}$ , if the reward of its application is  $r_i = 1$ , then  $\alpha_i$  and  $\beta_i$  are updated as:

$$\alpha_i = \alpha_i + 1, \beta_i = \beta_i. \tag{6}$$

Otherwise, if  $r_i = 0$  then we have:

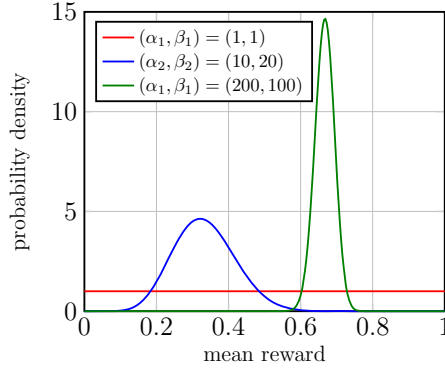
$$\alpha_i = \alpha_i, \beta_i = \beta_i + 1, \tag{7}$$

whilst the other parameters are kept unchanged. In the decision-making stage, the arm having the largest sampling value from the distribution is chosen for the next step. This is the underlying mechanism of Thompson sampling for balancing exploration and exploitation. In practice, each component of  $\alpha$  and  $\beta$  is initialised to be 1 whilst  $\mathcal{P}^{Beta}(\theta_i)$  is set to be a uniform distribution over  $[0, 1]$ . The Beta distribution has a mean  $\alpha_i/(\alpha_i + \beta_i)$  and its distribution become more concentrated at its mean as  $\alpha_i + \beta_i$  grows. For example, as shown in Fig. 1, the probability density function of  $(\alpha_2, \beta_2) = (200, 100)$  is the most concentrated one whilst that of  $(\alpha_1, \beta_1) = (1, 1)$  is a flat line.

In order to apply Thompson sampling for AOS in an EA, we consider each reproduction operator as a bandit arm with a Bernoulli distribution. In particular, an operator receives a reward of one in case its application improves the evolutionary population towards optimum, otherwise its reward is none.

### 3.2 Dynamic Thompson Sampling

Since the reward distribution used in the aforementioned vanilla Thompson sampling is fixed *a priori*, it does not fit the non-stationary nature of EAs. In other words, the reward distribution of a reproduction operator is highly likely to change with the progress of the evolution. To address this issue, this paper considers using a dynamic Thompson sampling [6] strategy (DYTS) to serve as the



**Fig. 1.** Illustration of Beta distribution with different  $(\alpha, \beta)$  settings.

---

**Algorithm 1:** ParameterUpdate( $\alpha_i, \beta_i, r, C$ ): parameter update rule of the DYTS strategy

---

**Input:** Parameters of the  $i$ -th operator  $\alpha_i, \beta_i$ , its up to date reward  $reward$  and threshold  $C$   
**Output:** Updated  $(\alpha_i, \beta_i)$

```

1 if  $\alpha_i + \beta_i < C$  then
2    $\alpha_i = \alpha_i + r$ ;
3    $\beta_i = \beta_i + 1 - r$ ;
4 else
5    $\alpha_i = (\alpha_i + r) \frac{C}{C+1}$ ;
6    $\beta_i = (\beta_i + 1 - r) \frac{C}{C+1}$ ;
7 return  $(\alpha_i, \beta_i)$ ;
```

---

foundation of our AOS paradigm. In particular, DYTS mainly uses a threshold  $C$  to control the two different parameter update rules. If  $\alpha_i + \beta_i < C$ , the parameters of the Beta distribution are updated as the vanilla Thompson sampling introduced in Sect. 3.1. On the other hand, if  $\alpha_i + \beta_i \geq C$ , the corresponding update rule aims to tweak the parameters so that we have  $\alpha_i + \beta_i = C$  after this update. The pseudocode of the parameter update strategy of DYTS is given in Algorithm 1.

There are two characteristics of this parameter update rule.

- It ensures that  $\alpha_i + \beta_i \leq C$ . More specifically, if  $\alpha_i + \beta_i = C$ , we have

$$\begin{aligned}
 \alpha'_i + \beta'_i &= (\alpha_i + \beta_i + 1) \frac{C}{C+1} \\
 &= (C+1) \frac{C}{C+1} \quad , \\
 &= C
 \end{aligned}
 \tag{8}$$

where  $\alpha'_i$  and  $\beta'_i$  represent the updated  $\alpha_i$  and  $\beta_i$ . In this case, we are able to avoid concentration of samples coming out of the DYTS strategy. Henceforth, it enables the DYTS a certain level of exploration ability.



- It gives larger weights to those recent awards thus help track the up to date reward distribution. More specifically, let us denote the reward received by the  $i$ -th reproduction operator at the  $t$ -th update as  $r_i^t$ , the corresponding parameter  $\alpha_i^t$  can be derived as:

$$\begin{aligned}
 \alpha_i^t &= (\alpha_i^{t-1} + r_i^t) \frac{C}{C+1} \\
 &= ((\alpha_i^{t-2} + r_i^{t-1}) \frac{C}{C+1} + r_i^t) \frac{C}{C+1} \\
 &= \alpha_i^{t-2} (\frac{C}{C+1})^2 + r_i^{t-1} (\frac{C}{C+1})^2 + r_i^t (\frac{C}{C+1}).
 \end{aligned} \tag{9}$$

Analogously, we can have the same derivation of  $\beta_i^t$ . According to this derivation, we can see that the latest reward is assigned with the largest decay ratio. By this means, the DYTS strategy enables the Thompson sampling to a non-stationary environment.

### 3.3 Operator Pool

From our preliminary experiments, this paper considers using the following five reproduction operator to constitute our operator pool. In particular, four of them are differential evolution (DE) variants directly derived from our previous paper [9] whilst the other one is the uniform mutation (UM) [4].

- DE/rand/1:  $\mathbf{x}^c = \mathbf{x}^i + F * (\mathbf{x}^{r1} - \mathbf{x}^{r2})$ ,
- DE/rand/2:  $\mathbf{x}^c = \mathbf{x}^i + F * (\mathbf{x}^{r1} - \mathbf{x}^{r2}) + F * (\mathbf{x}^{r3} - \mathbf{x}^{r4})$ ,
- DE/current-to-rand/1:  $\mathbf{x}^c = \mathbf{x}^i + K * (\mathbf{x}^i - \mathbf{x}^{r1}) + F * (\mathbf{x}^{r2} - \mathbf{x}^{r3})$ ,
- DE/current-to-rand/2:  $\mathbf{x}^c = \mathbf{x}^i + K * (\mathbf{x}^i - \mathbf{x}^{r1}) + F * (\mathbf{x}^{r2} - \mathbf{x}^{r3}) + F * (\mathbf{x}^{r4} - \mathbf{x}^{r5})$ ,
- UM:  $\mathbf{x}^c = \mathbf{x}^i + \text{uniform}(0, 1) * (\mathbf{u} - \mathbf{l})$ ,

where  $\text{uniform}(0, 1)$  represents a uniform distribution within the range  $[0, 1]$ ,  $\mathbf{u} = (u_1, \dots, u_n)$  and  $\mathbf{l} = (l_1, \dots, l_n)$  indicate the upper and lower bounds of decision space.

### 3.4 Credit Assignment

It is worth noting that the reward considered in the Bernoulli distribution is a binary value thus is not directly applicable in an EA. To address this issue, this paper uses the fitness improvement (FI) as the measure to evaluate the credit of an application of the reproduction operator chosen by the AOS. Formally, FI is defined as:

$$FI = \max_{\mathbf{x}^i \in P} \{g^{\text{tch}}(\mathbf{x}^i | \mathbf{w}^i, \mathbf{z}^*) - g^{\text{tch}}(\mathbf{x}^c | \mathbf{w}^i, \mathbf{z}^*)\}, \tag{10}$$

where  $\mathbf{w}^i$  is the weight vector associated with the subproblem of a solution  $\mathbf{x}^i \in P$ ,  $i \in \{1, \dots, N\}$  and  $\mathbf{x}^c$  is the offspring solution generated by using the selected operator. If  $FI > 0$ , it indicates that the application of the selected operator is successful thus the corresponding reward is one. Otherwise, the reward is set to be none instead.

---

**Algorithm 2:** AOS( $\alpha, \beta$ ): adaptive operator selection based on DYTS

---

**Input:** Parameters of the Beta distributions  $\alpha = (\alpha_1, \dots, \alpha_k)$  and  $\beta = (\beta_1, \dots, \beta_k)$   
**Output:** Index of the selected reproduction operator

```

1 for  $i \leftarrow 1$  to  $k$  do
2    $\lfloor$  Sample the estimated mean reward  $\hat{\theta}_i$  from Beta distribution  $\mathcal{P}^{Beta}(\alpha_i, \beta_i)$ .
3    $a = \operatorname{argmax}_{i \in \{1, \dots, k\}} \hat{\theta}_i$ ;
4 return  $a$ ;
```

---

### 3.5 AOS Based on DYTS Strategy

The idea of our proposed AOS paradigm based on the DYTS strategy is simple and intuitive. As shown in Algorithm 2, the operator having the largest sampling value from the up to date Beta distribution is chosen as the target operator for the next iteration.

### 3.6 Framework of MOEA/D-DYTS

Our proposed AOS based on DYTS can be applied to MOEA/D-DRA (dubbed MOEA/D-DYTS) in a plug-in manner without any significant modification. In particular, we only need to maintain an operator pool and keep a record of the FI achieved by the application of an operator. As the pseudocode of MOEA/D-DYTS given in Algorithm 3, we can see that most parts are the same as the original MOEA/D-DRA. The only difference lies in the offspring reproduction where the AOS based on DYTS strategy is applied to select the most appropriate operator in practice (line 8 of Algorithm 3). In addition, after the generation of an offspring, the corresponding *FI* is calculated followed by an update of the reward (lines 17 to 22 of Algorithm 3). Thereafter, the collected reward is used to update the parameters of the Beta distribution (line 23 of Algorithm 3).

## 4 Experimental Studies

In this section, we will use a set of experiments to validate the effectiveness of our proposed MOEA/D-DYTS. The experimental settings used in this paper are briefly overviewed in Sect. 4.1 including the benchmark test problems, parameter settings and the peer algorithms used in our experiments.

### 4.1 Experimental Settings

In our experimental studies, 19 unconstrained test problems are used to constitute the benchmark suite including UF1 to UF10 from the CEC 2009 MOEA competition [15] and WFG1 to WFG9 chosen from the Walking Fish Group test problem set [7]. In particular, the number of decision variables of UF problem is 30 whilst it is set to 38 (18 are position related and 20 are distance related) for the WFG problems. Four state-of-the-art MOEA/D variants i.e., MOEA/D-FRRMAB [9], MOEA/D-GRA [16], MOEA/D-IRA [10] and MOEA/D-DE [8]

**Algorithm 3:** MOEA/D-DYTS

---

```

Input: Algorithm parameters
Output: Approximated solution set  $\mathcal{P}$ 
1 Initialise the population  $\mathcal{P} = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ , the weight vectors  $\{\mathbf{w}^1, \dots, \mathbf{w}^N\}$ , parameters
  of the Beta distribution  $\alpha, \beta$  and the ideal point  $\mathbf{z}^*$ ;
2  $gen \leftarrow 0$ ,  $neval \leftarrow 0$ ;
3 for  $i \leftarrow 1$  to  $N$  do
4    $\mathcal{B}(i) \leftarrow \{i_1, \dots, i_T\}$  where  $\mathbf{w}^{i_1}, \dots, \mathbf{w}^{i_T}$  are the  $T$  closest weight vectors to  $\mathbf{w}^i$  and set
      $\pi_i \leftarrow 1$ ;
5 while  $neval < maxEvaluations$  do
6   Let all the indices of the subproblems whose objectives are MOP individual objectives
      $f_i$ ; form the initial  $I$ . By using 10-tournament selection based on  $\pi_i$ , select other
      $\lfloor N/5 \rfloor - m$  indices and add them to  $I$ ;
7   for each  $i \in I$  do
8      $op \leftarrow AOS(\alpha, \beta)$ ;
9     if  $uniform(0, 1) < \delta$  then
10       $P \leftarrow \mathcal{B}(i)$ ;
11    else
12       $P \leftarrow$  the entire population ;
13    Randomly select a required number of parent solutions from  $P$ ;
14    Generate an offspring  $\mathbf{x}^c$  by the  $op$ -th operator over the selected solutions;
15    Use polynomial mutation to further mutate  $\mathbf{x}^c$ ;
16    Update the ideal point  $\mathbf{z}^*$  according to  $\mathbf{x}^c$ ;
17     $FI \leftarrow \max_{\mathbf{x}^i \in P} \{g^{tch}(\mathbf{x}^i | \mathbf{w}^i, \mathbf{z}^*) - g^{tch}(\mathbf{x}^c | \mathbf{w}^i, \mathbf{z}^*)\}$ ;
18    if  $FI > 0$  then
19       $reward \leftarrow 1$ ;
20      Replace the  $\mathbf{x}^i$  associated with  $FI$  by  $\mathbf{x}^c$ ;
21    else
22       $reward \leftarrow 0$ ;
23     $(\alpha_{op}, \beta_{op}) \leftarrow ParameterUpdate(\alpha_{op}, \beta_{op}, reward, C)$ ;
24     $neval \leftarrow neval + 1$ ;
25   $gen \leftarrow gen + 1$ ;
26  if  $modulo(gen, 50) == 0$  then
27    Update the utility  $\pi^i$  of each subproblem  $i, i \in \{1, \dots, N\}$ ;
28 return  $P$ ;
```

---

are used as the peer algorithms in comparison. The parameters associated with these peer algorithms are set the same as recommended in their original paper. Those of our proposed MOEA/D-DYTS are set as follows:

- The population size  $N$  is set to 300 for the two-objective UF instances and 600 for the three-objective UF instances. As for WFG instances, we set  $N = 100$ .
- Each algorithm is run 31 times on each test problem instance. The maximum number of function evaluations is set to 300,000 for the UF instances and 25,000 for the WFG instances.
- The neighbourhood size is fixed to 20. Probability  $\delta$  with regard to selecting  $P$  is set to 0.8 as suggested in [10].
- The update threshold of our DYTS strategy is set as  $C = 100$ .

To evaluate the performance of different algorithms, two widely used performance metrics, i.e., inverted generational distance (IGD) [2] and Hypervolume (HV) [18], are used in our experiments. Both of them are able to evaluate the convergence and diversity simultaneously. In order to calculate the IGD, 10,000

**Table 1.** Comparative results of all the algorithms on the UF and WFG test problems regarding IGD.

	MOEA/D-DE	MOEA/D-FRRMAB	MOEA/D-GRA	MOEA/D-IRA	MOEA/D-DYTS
UF1	1.97E-3 <sub>1.56E-4</sub> <sup>-</sup>	2.50E-3 <sub>2.19E-4</sub> <sup>-</sup>	1.90E-3 <sub>8.3E-5</sub> <sup>~</sup>	1.92E-3 <sub>8.9E-5</sub> <sup>~</sup>	<b>1.89E-3</b> <sub>7.1E-5</sub>
UF2	7.12E-3 <sub>1.58E-3</sub> <sup>-</sup>	5.64E-3 <sub>5.21E-4</sub> <sup>-</sup>	<b>3.92E-3</b> <sub>5.01E-4</sub> <sup>~</sup>	4.16E-3 <sub>5.92E-4</sub> <sup>~</sup>	4.09E-3 <sub>9.86E-4</sub>
UF3	1.20E-2 <sub>1.33E-2</sub> <sup>-</sup>	6.97E-3 <sub>4.81E-3</sub> <sup>-</sup>	4.24E-3 <sub>2.31E-3</sub> <sup>~</sup>	5.27E-3 <sub>2.69E-3</sub> <sup>-</sup>	<b>4.04E-3</b> <sub>2.07E-3</sub>
UF4	6.27E-2 <sub>4.09E-3</sub> <sup>-</sup>	5.26E-2 <sub>3.88E-3</sub> <sup>-</sup>	5.48E-2 <sub>3.41E-3</sub> <sup>-</sup>	5.36E-2 <sub>2.85E-3</sub> <sup>-</sup>	<b>3.04E-2</b> <sub>1.28E-3</sub>
UF5	3.12E-1 <sub>1.15E-1</sub> <sup>-</sup>	2.90E-1 <sub>6.89E-2</sub> <sup>-</sup>	2.41E-1 <sub>2.74E-2</sub> <sup>-</sup>	2.35E-1 <sub>2.83E-2</sub> <sup>-</sup>	<b>1.26E-1</b> <sub>2.37E-2</sub>
UF6	1.85E-1 <sub>1.72E-1</sub> <sup>-</sup>	2.07E-1 <sub>1.84E-2</sub> <sup>-</sup>	<b>7.43E-2</b> <sub>2.97E-2</sub> <sup>+</sup>	8.17E-2 <sub>5.06E-2</sub> <sup>+</sup>	1.23E-1 <sub>6.38E-2</sub>
UF7	4.07E-3 <sub>4.24E-3</sub> <sup>-</sup>	2.64E-3 <sub>3.37E-4</sub> <sup>-</sup>	2.05E-3 <sub>1.02E-4</sub> <sup>-</sup>	2.04E-3 <sub>1.03E-4</sub> <sup>-</sup>	<b>1.91E-3</b> <sub>8.90E-5</sub>
UF8	7.82E-2 <sub>1.27E-2</sub> <sup>-</sup>	6.96E-2 <sub>1.29E-2</sub> <sup>~</sup>	8.11E-2 <sub>1.43E-2</sub> <sup>-</sup>	8.12E-2 <sub>1.20E-2</sub> <sup>-</sup>	<b>6.86E-2</b> <sub>1.85E-2</sub>
UF9	8.82E-2 <sub>4.96E-2</sub> <sup>-</sup>	7.97E-2 <sub>4.55E-2</sub> <sup>-</sup>	3.82E-2 <sub>3.40E-2</sub> <sup>+</sup>	<b>3.49E-2</b> <sub>2.79E-2</sub> <sup>+</sup>	4.24E-2 <sub>2.55E-2</sub>
UF10	5.14E-1 <sub>6.85E-2</sub> <sup>-</sup>	7.56E-1 <sub>1.19E-1</sub> <sup>-</sup>	1.32E+00 <sub>2.39E-1</sub> <sup>-</sup>	1.53E+00 <sub>3.24E-1</sub> <sup>-</sup>	<b>4.06E-1</b> 5.87E-2
WFG1	1.28E+00 <sub>8.03E-3</sub> <sup>-</sup>	1.28E+00 <sub>4.06E-3</sub> <sup>-</sup>	1.26E+00 <sub>8.36E-3</sub> <sup>-</sup>	1.26E+00 <sub>6.45E-3</sub> <sup>-</sup>	<b>1.20E+00</b> 6.08E-3
WFG2	4.06E-1 <sub>2.20E-1</sub> <sup>-</sup>	2.54E-1 <sub>1.37E-1</sub> <sup>~</sup>	2.91E-1 <sub>1.63E-1</sub> <sup>-</sup>	2.59E-1 <sub>1.37E-1</sub> <sup>-</sup>	<b>2.33E-1</b> 1.22E-1
WFG3	1.55E-2 <sub>4.42E-3</sub> <sup>-</sup>	1.40E-2 <sub>2.74E-3</sub> <sup>-</sup>	1.39E-2 <sub>2.43E-3</sub> <sup>-</sup>	1.39E-2 <sub>2.57E-3</sub> <sup>-</sup>	<b>1.32E-2</b> 1.00E-6
WFG4	2.89E-2 <sub>5.91E-3</sub> <sup>-</sup>	2.81E-2 <sub>4.34E-3</sub> <sup>-</sup>	1.87E-2 <sub>3.96E-3</sub> <sup>-</sup>	2.07E-2 <sub>6.60E-3</sub> <sup>-</sup>	<b>1.72E-2</b> 1.89E-3
WFG5	4.23E-2 <sub>1.19E-2</sub> <sup>-</sup>	3.88E-2 <sub>1.14E-2</sub> <sup>-</sup>	3.02E-2 <sub>6.50E-3</sub> <sup>-</sup>	3.06E-2 <sub>6.72E-3</sub> <sup>-</sup>	<b>2.12E-2</b> 1.78E-3
WFG6	5.97E-2 <sub>2.89E-2</sub> <sup>-</sup>	3.80E-2 <sub>2.30E-2</sub> <sup>-</sup>	3.30E-2 <sub>9.99E-3</sub> <sup>-</sup>	3.78E-2 <sub>1.00E-2</sub> <sup>-</sup>	<b>2.44E-2</b> 9.90E-3
WFG7	1.45E-2 <sub>4.89E-3</sub> <sup>-</sup>	1.41E-2 <sub>3.76E-3</sub> <sup>-</sup>	1.43E-2 <sub>4.41E-3</sub> <sup>-</sup>	1.44E-2 <sub>4.36E-3</sub> <sup>-</sup>	<b>1.31E-2</b> 2.00E-6
WFG8	1.60E-2 <sub>6.14E-3</sub> <sup>-</sup>	1.42E-2 <sub>4.10E-3</sub> <sup>-</sup>	1.36E-2 <sub>2.63E-3</sub> <sup>-</sup>	1.37E-2 <sub>3.30E-3</sub> <sup>~</sup>	<b>1.31E-2</b> 2.00E-6
WFG9	3.83E-2 <sub>5.54E-3</sub> <sup>-</sup>	3.64E-2 <sub>5.11E-3</sub> <sup>-</sup>	3.81E-2 <sub>7.61E-3</sub> <sup>-</sup>	3.70E-2 <sub>4.96E-3</sub> <sup>-</sup>	<b>2.63E-2</b> 2.85E-3
-/~/+	19/0/0	17/2/0	13/4/2	14/3/2	

points were uniformly sampled from the true PF to constitute the reference set. The lower the IGD is, the better the solution set for approximating the PF. As for the HV calculation, we set the reference point as (2.0, 2.0) for two-objective UF instances and (2.0, 2.0, 2.0) for three-objective UF instances. For the WFG instances, it is set as (3.0, 5.0). In contrast to the IGD, the larger the HV is, the better quality of the solution set for approximating the PF.

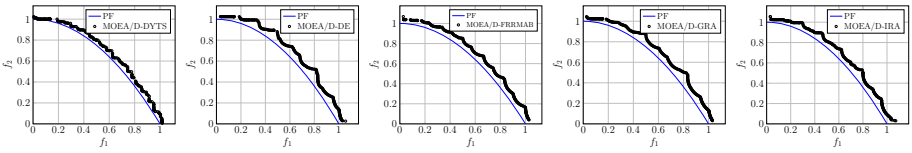
### 4.2 Experimental Results

In this section, we have compared MOEA/D-DYTS with four state-of-the-art MOEA/D variants, namely, MOEA/D-DE, MOEA/D-FRRMAB, MOEA/D-GRA and MOEA/D-IRA on UF instances and WFG instances. Tables 1 and 2 present the result of the IGD and HV metric values obtained from 31 independent runs. The best mean result for each problem is highlighted in boldface with grey background. Wilcoxon’s rank sum test with a 5% significance level was also conducted to provide a statistically conclusion. Where “-”, “+” and “~” denote that the results obtained by corresponding algorithm are worse than, better than or similar to those of MOEA/D-DYTS.

From Tables 1 and 2 we clearly see that MOEA/D-DYTS is the best algorithm. With respect to the IGD values, MOEA/D-DYTS performs best on UF1, UF3 to UF5, UF7, UF8, UF10 and all WFG test instances. Considering the HV values, Table 2 gives the similar result to Table 1. In total, it has obtained better results in 16 out of 19 performance comparisons for IGD and 18 out of 19 for HV. In the following paragraphs, we will give a gentle discussion over these results.

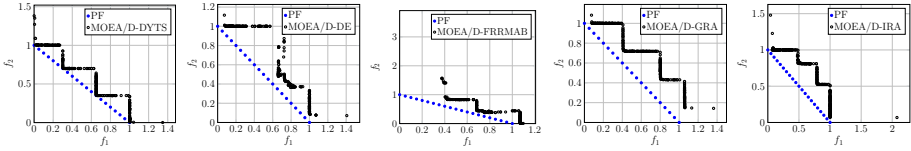
**Table 2.** Comparative results of all the algorithms on the UF and WFG test problems regarding HV.

	MOEA/D-DE	MOEA/D-FRRMAB	MOEA/D-GRA	MOEA/D-IRA	MOEA/D-DYTS
UF1	3.65874 <sub>1.54E-3</sub>	3.65673 <sub>1.85E-3</sub>	3.65992 <sub>1.13E-3</sub>	3.65973 <sub>1.08E-3</sub>	<b>3.66314</b> 3.08E-4
UF2	3.63909 <sub>1.59E-2</sub>	3.64864 <sub>7.83E-3</sub>	3.65107 <sub>8.39E-3</sub>	3.65291 <sub>8.85E-3</sub>	<b>3.65313</b> 1.10E-2
UF3	3.62601 <sub>6.92E-2</sub>	3.64753 <sub>1.74E-2</sub>	3.65765 <sub>5.71E-3</sub>	3.65162 <sub>1.51E-2</sub>	<b>3.65836</b> 5.60E-3
UF4	3.14765 <sub>1.59E-2</sub>	3.18129 <sub>1.19E-2</sub>	3.17597 <sub>1.03E-2</sub>	3.17725 <sub>1.27E-2</sub>	<b>3.26432</b> 4.40E-3
UF5	2.56553 <sub>2.35E-1</sub>	2.71102 <sub>2.07E-1</sub>	2.90646 <sub>9.60E-2</sub>	2.92076 <sub>1.01E-1</sub>	<b>3.07248</b> 1.62E-1
UF6	2.90857 <sub>3.27E-1</sub>	2.84999 <sub>4.43E-1</sub>	<b>3.17650</b> 6.75E-2 <sup>+</sup>	3.14163 <sub>1.53E-1</sub> <sup>+</sup>	3.02291 <sub>2.11E-1</sub>
UF7	3.46933 <sub>4.88E-2</sub>	3.49045 <sub>2.29E-3</sub>	3.49274 <sub>2.55E-3</sub>	3.49232 <sub>2.01E-3</sub>	<b>3.49626</b> 2.46E-4
UF8	7.18889 <sub>2.24E-2</sub>	7.17681 <sub>3.06E-2</sub>	7.19519 <sub>2.28E-2</sub>	7.19303 <sub>2.26E-2</sub>	<b>7.27682</b> 3.68E-2
UF9	7.06634 <sub>4.78E-1</sub>	7.28091 <sub>3.52E-1</sub>	7.55889 <sub>1.51E-1</sub>	7.53802 <sub>2.51E-1</sub>	<b>7.59815</b> 1.15E-1
UF10	3.49981 <sub>3.94E-1</sub>	2.42976 <sub>4.40E-1</sub>	0.89670 <sub>5.48E-1</sub>	0.56221 <sub>5.53E-1</sub>	<b>4.49922</b> 5.35E-1
WFG1	5.09320 <sub>1.42E-1</sub>	5.16428 <sub>6.33E-2</sub>	5.19093 <sub>9.00E-2</sub>	5.17729 <sub>9.63E-2</sub>	<b>5.58013</b> 2.70E-2
WFG2	10.02039 <sub>6.70E-1</sub>	10.50875 <sub>4.29E-1</sub>	10.37559 <sub>4.91E-1</sub>	10.46272 <sub>4.22E-1</sub>	<b>10.58567</b> 3.96E-1
WFG3	10.89220 <sub>1.19E-1</sub>	10.92397 <sub>8.95E-2</sub>	10.93098 <sub>8.32E-2</sub>	10.93077 <sub>8.34E-2</sub>	<b>10.95286</b> 2.00E-6
WFG4	8.40620 <sub>5.17E-2</sub>	8.41161 <sub>3.88E-2</sub>	8.51377 <sub>4.25E-2</sub>	8.49385 <sub>6.52E-2</sub>	<b>8.53702</b> 2.97E-2
WFG5	8.06233 <sub>1.18E-1</sub>	8.09696 <sub>1.20E-1</sub>	8.19163 <sub>8.07E-2</sub>	8.18601 <sub>8.67E-2</sub>	<b>8.35099</b> 4.20E-2
WFG6	8.22110 <sub>1.52E-1</sub>	8.39915 <sub>2.09E-1</sub>	8.37701 <sub>1.10E-1</sub>	8.34767 <sub>1.01E-1</sub>	<b>8.49870</b> 1.45E-1
WFG7	8.63700 <sub>1.25E-1</sub>	8.65283 <sub>9.48E-2</sub>	8.64190 <sub>1.24E-1</sub>	8.63866 <sub>1.25E-1</sub>	<b>8.67611</b> 3.00E-6
WFG8	8.59288 <sub>1.77E-1</sub>	8.64452 <sub>1.20E-1</sub>	8.66164 <sub>7.84E-2</sub>	8.65935 <sub>9.16E-2</sub>	<b>8.67610</b> 9.00E-6
WFG9	8.17868 <sub>5.21E-2</sub>	8.19778 <sub>4.88E-2</sub>	8.18257 <sub>7.02E-2</sub>	8.19077 <sub>4.86E-2</sub>	<b>8.30853</b> 3.39E-2
-/~/+	19/0/0	19/0/0	14/4/1	14/4/1	

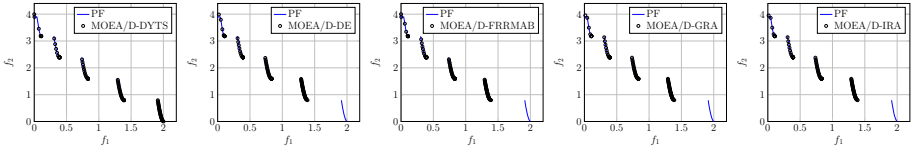
**Fig. 2.** Non-dominated solutions obtained by five algorithms on UF4 with the best IGD value.

UF1 to UF3 and UF7 are relatively simple test problems among UF benchmark problems, on which all the algorithms do not have too much difficulty to converge to the global PFs as shown in Figs. 1, 2, 3 and Fig. 7 in the supplementary document<sup>1</sup>. However, it is interesting to observe that MOEA/D-GRA achieves better convergency on the tail of the PF of UF2 on which our proposed algorithm takes a fall. On UF4 only MOEA/D-DYTS can find some solutions on the PF whereas the solutions found by other four algorithms are away from the PF. The PF of UF5 consists of 21 points which is challenging for EAs to converge. As shown in Fig. 3, the solutions obtained by MOEA/D-DYTS are much closer to the PF than the other algorithms on UF5. The similar results can also be observed on UF10. For other two three-objective UF test problems, all the algorithms can find most of the solutions of PF on UF8. Note that MOEA/D-DYTS is beaten by MOEA/D-GRA and MOEA/D-IRA on UF9 test problem

<sup>1</sup> <https://cola-laboratory.github.io/supplementary/dyts-sup.pdf>.



**Fig. 3.** Non-dominated solutions obtained by five algorithms on UF5 with the best IGD value.



**Fig. 4.** Non-dominated solutions obtained by five algorithms on WFG2 with the best IGD value.

which has two disconnected parts of PS. This may imply that MOEA/D-DYTS do not have enough search ability in decision space.

For WFG test problems, MOEA/D-DYTS has completely won all the 9 test instances. As observed from Tables 1 and 2, MOEA/D-DYTS perform significantly better than other four algorithms on WFG1. Regarding WFG2, it is a discontinuous problem whose PF is five disconnected segments. As shown in Fig. 4, no algorithm can find solutions on the last segment on WFG2 except for MOEA/D-DYTS. For WFG3 to WFG9, all the compared algorithms can converge to the true PFs. The Wilcoxon rank sum test shows that MOEA/D-DYTS is similar to MOEA/D-GRA and MOEA/D-IRA on WFG3, WFG7 and WFG8 according to the results of HV values. It is worth noting that our proposed algorithm MOEA/D-DYTS has best mean values on WFG3, WFG7 and WFG8 from 31 independent runs. This implies that MOEA/D-DYTS can achieve better stability than other algorithms in optimisation process. As shown in Figs. 14 to 16 and 19 in the supplementary document, the better diversity of solutions found by MOEA/D-DYTS on the head or the tail of the PF can be observed on WFG4 to WFG6 and WFG9. These results indicate that our proposed algorithm can be favourable among other four state-of-the-art MOEA/D variants.

## 5 Conclusion

This paper proposes a new AOS paradigm for MOEA/D that is able to autonomously select the appropriate reproduction operator for the next step. Specifically, the dynamic Thompson sampling is served as the foundation for AOS. Different from the vanilla Thompson sampling and bandit learning model, the dynamic Thompson sampling is able to track the search dynamics under a non-stationary environment. From our empirical results, we have witnessed the

superiority of our proposed MOEA/D-DYTS over four state-of-the-art MOEA/D variants on 19 test problems.

AOS is an attractive paradigm to equip EAs with intelligence to autonomously adapt their search behaviour according to the current search landscapes. In addition to the bandit model considered in this paper, it is also interesting to look into methods from reinforcement learning or automatic control domain for new decision-making paradigm. Furthermore, the credit assignment of an application of an operator is important to gear the AOS. More sophisticated methods are worthwhile being considered in future.

## References

1. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.* **3**, 397–422 (2002)
2. Bosman, P.A.N., Thierens, D.: The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Trans. Evol. Comput.* **7**(2), 174–188 (2003)
3. Clark, D.E., Westhead, D.R.: Evolutionary algorithms in computer-aided molecular design. *J. Comput. Aided Mol. Des.* **10**(4), 337–358 (1996)
4. Deb, K.: *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. Wiley, Hoboken (2001)
5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002)
6. Gupta, N., Granmo, O., Agrawala, A.K.: Thompson sampling for dynamic multi-armed bandits. In: Chen, X., Dillon, T.S., Ishibuchi, H., Pei, J., Wang, H., Wani, M.A. (eds.) *ICMLA 2011: Proceedings of the 2011 10th International Conference on Machine Learning and Applications*, pp. 484–489. IEEE Computer Society (2011)
7. Huband, S., Hingston, P., Barone, L., While, R.L.: A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **10**(5), 477–506 (2006)
8. 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)
9. Li, K., Fialho, Á., Kwong, S., Zhang, Q.: Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **18**(1), 114–130 (2014)
10. Lin, Q., et al.: A diversity-enhanced resource allocation strategy for decomposition-based multiobjective evolutionary algorithm. *IEEE Trans. Cybern.* **48**(8), 2388–2401 (2018)
11. di Pierro, F., Khu, S., Savic, D.A., Berardi, L.: Efficient multi-objective optimal design of water distribution networks on a budget of simulations using hybrid algorithms. *Environ. Model Softw.* **24**(2), 202–213 (2009)
12. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* **25**(3/4), 285–294 (1933)
13. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
14. Zhang, Q., Liu, W., Li, H.: The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In: *CEC 2009: Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pp. 203–208. IEEE (2009)

15. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the CEC 2009 special session and competition. *Mech. Eng.* (2008)
16. 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)
17. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) *PPSN 2004*. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30217-9\\_84](https://doi.org/10.1007/978-3-540-30217-9_84)
18. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans. Evol. Comput.* **3**(4), 257–271 (1999)





# A Study of Swarm Topologies and Their Influence on the Performance of Multi-Objective Particle Swarm Optimizers

Diana Cristina Valencia-Rodríguez<sup>(✉)</sup> and Carlos A. Coello Coello<sup>ID</sup>

CINVESTAV-IPN (Evolutionary Computation Group),  
Av. IPN 2508, San Pedro Zacatenco, 07360 Ciudad de México, Mexico  
dvalencia@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

**Abstract.** It has been shown that swarm topologies influence the behavior of Particle Swarm Optimization (PSO). A large number of connections stimulates exploitation, while a low number of connections stimulates exploration. Furthermore, a topology with four links per particle is known to improve PSO's performance. In spite of this, there are few studies about the influence of swarm topologies in Multi-Objective Particle Swarm Optimizers (MOPSOs). We analyze the influence of star, tree, lattice, ring and wheel topologies in the performance of the Speed-constrained Multi-objective Particle Swarm Optimizer (SMPSO) when adopting a variety of multi-objective problems, including the well-known ZDT, DTLZ and WFG test suites. Our results indicate that the selection of the proper topology does indeed improve the performance in SMPSO.

**Keywords:** Swarm topology · Particle Swarm Optimization · Multi-Objective Particle Swarm Optimization · Multi-objective optimization

## 1 Introduction

Particle Swarm Optimization (PSO) is a metaheuristic proposed in the mid-1990s by Kennedy and Eberhart [7] that mimics the social behavior of bird flocks and schools of fish. PSO searches a solution to an optimization problem using particles that move through the search space employing their best previous position and the best position of the particles to which that particle is connected. The graph that represents these connections is called *swarm topology*. It has been empirically shown that the topology influences the behavior of a single-objective PSO [6, 8]. A topology with many connections improves the exploitative

---

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 grant no. 2016-01-1920 (*Investigación en Fronteras de la Ciencia 2016*) and from a SEP-Cinvestav grant (application no. 4).

behavior of PSO, while a topology with few connections improves its explorative behavior [6].

A wide variety of Multi-Objective Particle Swarm Optimizers (MOPSOs) have been developed [10] over the years. However, unlike the case for single-objective PSO, studies on the influence of a swarm topology in the performance of a MOPSO are very scarce. Yamamoto et al. [12] studied the influence of a swarm topology for the bi-objective problems ZDT1, ZDT3, and ZDT4. They found that increasing the topology connections improves the convergence towards the true Pareto Front, and that decreasing such connections promotes diversity. On the other hand, Taormina and Chau [11] examined the effect of a swarm topology for a bi-objective problem of neural networks training. They noticed that a topology with four connections (a lattice topology) improves the performance of a MOPSO. Both studies offer relevant information about the influence of a swarm topology. However, the results of these two studies are limited to bi-objective problems having similar features. In contrast, the study presented in this paper covers a wide variety of problems with two and three objectives, taken from the Zitzler-Deb-Thiele (ZDT), the Deb-Thiele-Laumanns-Zitzler (DTLZ) and Walking-Fish-Group (WFG) test suites.

The remainder of this paper is organized as follows. In Sect. 2, we provide some basic concepts related to multi-objective optimization and PSO, including swarm topologies. Then, in Sect. 3, we describe the operation of the Speed-constrained Multi-objective Particle Swarm Optimizer (SMPSO) which is our baseline algorithm. Section 4 presents a discussion on the use of topologies in MOPSOs. Section 5 presents two schemes for handling swarm topologies in MOPSOs, as a framework for conducting the study presented herein. Our experimental results are provided in Sect. 6. Finally, our conclusions and some potential paths for future research are provided in Sect. 7.

## 2 Background

### 2.1 Multi-objective Optimization

We are interested in solving a continuous unconstrained multi-objective optimization problem that is defined as follows:

$$\underset{\mathbf{x} \in \Omega}{\text{minimize}} \quad F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}))^T \quad (1)$$

where  $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$  belongs to the decision variable space defined by  $\Omega$ . And  $F(\mathbf{x}) : \Omega \rightarrow \mathbb{R}^m$  consist of  $m$  objective functions  $f_i(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$  that are usually in conflict. In a multi-objective problem, we aim to find the best trade-off solutions that can be defined in terms of the notion of Pareto Optimality. We provide the following definitions to describe this concept.

**Definition 1.** Let  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$ ,  $\mathbf{u}$  is said to **dominates**  $\mathbf{v}$  (denoted by  $\mathbf{u} \preceq \mathbf{v}$ ), if and only if  $u_i \leq v_i$  for all  $i = 1, \dots, m$  and  $u_i < v_i$  for at least one index  $j \in \{1, \dots, m\}$ .

**Definition 2.** A solution  $\mathbf{x} \in \Omega$  is **Pareto Optimal** if it does not exist another solution  $\mathbf{y} \in \Omega$  such that  $F(\mathbf{y}) \preceq F(\mathbf{x})$ .

**Definition 3.** Given a multi-objective optimization problem  $(F(\mathbf{x}), \Omega)$ , the **Pareto Optimal Set (PS)** is defined by:

$$PS = \{\mathbf{x} \in \Omega \mid \mathbf{x} \text{ is a Pareto Optimal solution}\},$$

and its image  $PF = \{F(\mathbf{y}) \mid \mathbf{y} \in PS\}$  is called **Pareto Front**.

### 2.2 Particle Swarm Optimization

PSO is a bio-inspired metaheuristic that works with a set of particles (called *swarm*) that represents potential solutions to the optimization problem. Each particle  $\mathbf{x}_i \in \mathbb{R}^n$  at generation  $t$  updates its position using the following expression:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t - 1) + \mathbf{v}_i(t). \tag{2}$$

The factor  $\mathbf{v}_i(t)$  is called *velocity* and is defined by

$$\mathbf{v}_i(t) = w\mathbf{v}_i(t - 1) + C_1r_1(\mathbf{x}_{p_i} - \mathbf{x}_i(t - 1)) + C_2r_2(\mathbf{x}_{l_i} - \mathbf{x}_i(t - 1)) \tag{3}$$

where  $w$  is a positive constant known as inertia weight;  $C_1$  and  $C_2$  are positive constants known as *cognitive* and *social* factors, respectively;  $r_1$  and  $r_2$  are two random numbers with a uniform distribution in the range  $[0, 1]$ ;  $\mathbf{x}_{p_i}$  is the best personal position found by the  $i^{th}$  particle, and  $\mathbf{x}_{l_i}$  is the best particle to which it is connected (called *leader*). In order to define the connections that allow us to select the leader, we need to determine the topology of the swarm.

### 2.3 Swarm Topology

A swarm topology (or, simply, a topology) is a graph where each vertex represents a particle, and there is an edge between two particles if they influence each other [8]. The set of particles that affect a given particle is called *neighborhood*. In the experiments reported below, we use five topologies that have been studied before in PSO:

- **Fully connected** (star or *gbest*). All the particles in this topology influence each other [6]. See Fig. 1a. Therefore, the information between particles expands quickly.
- **Ring** (*lbest*). In this topology, each particle is influenced by its two immediate neighbors [6]. See Fig. 1b. For this reason, the information transmission between particles is slow.
- **Wheel**. It consists of one central particle that influences and is influenced by the remainder particles in the swarm [6]. See Fig. 1c. The central particle acts as a filter that delays the information.
- **Lattice**. In this topology, each particle is influenced by one particle above, one below and two on each side [6]. See Fig. 1d.
- **Tree**. The swarm in this topology is organized as a binary tree where each node represents a particle. See Fig. 1e.

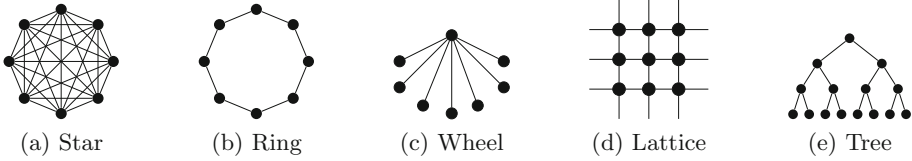


Fig. 1. Swarm topologies

### 3 SMPSO

In contrast to single-objective PSO, a MOPSO’s particle could have more than one leader due to the nature of multi-objective problems. Therefore, a large number of MOPSOs usually store their leaders in an *external archive*, which retains the non-dominated solutions found so far [10]. For this reason, we assume in this paper that a MOPSO works with an external archive and selects the leaders from it. Accordingly, we selected for the experimental analysis a standard Pareto-based MOPSO that works in this manner: the Speed-constrained Multi-objective Particle Swarm Optimizer (SMPSO) [9]. The core idea behind SMPSO is to control the particles’ velocity employing a constriction coefficient  $\chi$  defined by:

$$\chi = 2 / (2 - \varphi - \sqrt{\varphi^2 - 4\varphi}) \tag{4}$$

where  $\varphi = 1$  if  $C_1 + C_2$  is less or equal than four. Otherwise,  $\varphi = C_1 + C_2$ . Besides the constriction coefficient, SMPSO bounds the  $j^{th}$  velocity component of each  $i^{th}$  particle, denoted by  $v_{i,j}(t)$ , using the equation:

$$v_{i,j}(t) = \begin{cases} \delta_j & \text{if } v_{i,j}(t) > \delta_j \\ -\delta_j & \text{if } v_{i,j}(t) \leq -\delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \tag{5}$$

where  $\delta_j = (upper\_limit_j - lower\_limit_j) / 2$ , and the upper and lower limits of the  $j^{th}$  decision variable are *upper\_limit<sub>j</sub>* and *lower\_limit<sub>j</sub>* respectively.

In summary, for computing the velocity, SMPSO selects the leader by randomly taking two solutions from the external archive and chooses the one with the largest crowding distance, which measures how isolated a particle is from the others. After that, the velocity is estimated with the selected leader using Eq. (3). Then, the result is multiplied by the constriction factor defined in Eq. (4) and bounded using the rule defined in Eq. (5).

SMPSO works in the following way. First, the swarm is randomly initialized, and the external archive is constructed with the non-dominated solutions currently available. During a certain (pre-defined) number of iterations, the velocity and position of each particle is computed. Then, polynomial-based mutation [1] is applied to the resulting individual, using a mutation rate  $p_m$ , and the new particle is evaluated. Finally, the particles’ personal best and the external archive

**Algorithm 1.** Pseudocode of SMPSO

---

```

1: Initialize the swarm with random values
2: Initialize the external archive with the non-dominated solutions of the swarm
3: while the maximum number of iterations is not reached do
4:   for each particle  $p_i$  in the swarm do
5:     Randomly take two solutions from the external archive and select the one
       with the largest crowding distance as the leader  $x_{l_i}$ 
6:     Compute the velocity using equation (3) and multiply it by equation (4)
7:     Constrain the velocity using equation (5)
8:     Compute the particle's position with equation (2)
9:     Apply polynomial-based mutation
10:    Evaluate the new particle
11:  end for
12:  Update the particle's memory and the external archive
13:  if the size of the external archive exceeds its limit then
14:    Remove from the external archive the particle with the lowest crowding
       distance
15:  end if
16: end while

```

---

are updated. If the archive exceeds a pre-defined limit, the solution with the lowest crowding distance is removed. The pseudocode of SMPSO is shown in Algorithm 1.

## 4 Handling Topologies in Multi-Objective Particle Swarm Optimizers

In PSO, each particle updates its best personal position by comparing both the current and the previous positions and selecting the best one. Furthermore, the leader of each particle is selected by examining the best personal position of the particles to which it is connected. In multi-objective problems, however, we cannot select just one solution as the best. Therefore many MOPSOs store the best position found by particles in an external archive and select the leaders from it. This leader selection scheme does not allow MOPSOs to use distinct topologies because the neighborhood of a particle is not examined to select its leader. Moreover, many MOPSOs use a fully connected topology because each particle takes into consideration the positions found by the whole swarm. For this reason, it is necessary to design a leader selection scheme to handle swarm topologies in MOPSOs.

Yamamoto *et al.* [12] introduced a topology handling scheme where each particle had a sub-archive that was updated by the particle and its neighbors. Accordingly, each particle selected its leader from its sub-archive and the sub-archives of its neighbors. One advantage of this scheme is that it promotes diversity because a solution from a sub-archive could dominate a solution from another one. Furthermore, this scheme allows us to manipulate directly the best position

found by the particles. On the other hand, one disadvantage of this scheme is that the space and time complexity of the MOPSO increase due to the use of sub-archives, and they get worse when the population size is increased.

Taormina and Chau [11] proposed another topology handling scheme where the leaders are added to the swarm. Each leader will influence four particles, but the particles will not influence the leaders, so they will not move. Taormina and Chau mentioned that these leaders are instances of non-dominated solutions found by the particles, but they do not provide any further information.

Due to the disadvantages of these two previously described schemes, we propose here two topology handling schemes which are described next.

## 5 The Proposed Topology Handling Schemes

In order to analyze the influence of the topology in MOPSOs, we propose two topology handling schemes and implement them in SMPSO. Both schemes differ only in the place from which the leader is taken: either the particle's memory or the external archive:

### 5.1 Scheme 1

The idea of scheme 1 is to emulate the leader selection scheme from a single-objective PSO. Therefore, it selects the leader of each particle by examining the personal best positions of the particles in the neighborhood and selecting the best from them. In order to implement scheme one in SMPSO (we called this algorithm SMPSO-E1), we modified line 5 of Algorithm 1. Thus, SMPSO-E1 obtains the particle's neighborhood and saves it in  $N_i$ . Next, it selects as a leader, the particle whose personal best position dominates most of the others in  $N_i$ . After that, the particle's position and its velocity are computed as in the original SMPSO.

### 5.2 Scheme 2

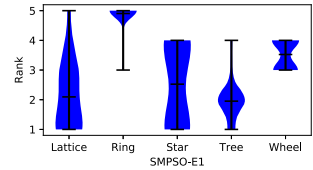
Under scheme 2, we associate each element of the external archive to each particle in the swarm, i.e., the  $i^{th}$  element of the external archive is associated with the  $i^{th}$  particle in the swarm. If the archive size is smaller than the swarm size, the archive elements are assigned again. Furthermore, the swarm size is restricted to be larger or equal to the archive size. Afterwards, a particle will select its leader by exploring the external archive components that are assigned to the particle's neighbors. The idea of scheme 2 is to use each external archive element as an alternative memory, in order to operate with the global best positions as leaders. In order to implement scheme 2 in SMPSO (we named this algorithm SMPSO-E2), we modified Algorithm 1. First, before computing the new positions of the particles, we assign the external archive elements to each particle. Then, for each particle, we randomly take two elements in the neighborhood and select as leader the one with the largest crowding distance. After that, we compute the particle's distance as in the original SMPSO.

## 6 Experiments and Analysis

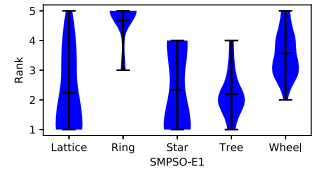
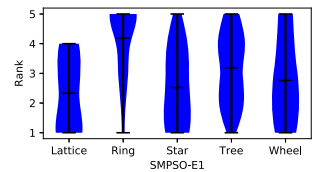
In this work, we compare five state-of-the-art topologies: star, ring, lattice, wheel, and tree. The influence of each topology is evaluated both using SMPSO-E1 and SMPSO-E2. We also contrast the performance of SMPSO-E1, SMPSO-E2, and the original version of SMPSO. In order to analyze the impact of a particular topology in the performance of a MOPSO, we adopted several test problems: the Zitzler-Deb-Thiele (ZDT) [14], the Deb-Thiele-Laumanns-Zitzler (DTLZ) [2], and the Walking Fish Group (WFG) [4] test suites. From the ZDT test suite, we excluded ZDT5 due to its discrete nature. We use 3-objective instances of DTLZ and WFG problems. The number of variables is 30 for ZDT1 to ZDT3, and 10 for ZDT4 and ZDT6. In the case of the DTLZ problems, the number of variables is  $n = 3 + k - 1$ , where  $k = 5$  for DTLZ1,  $k = 10$  for DTLZ2 to DTLZ6, and  $k = 20$  for DTLZ7. Finally, we use 24 variables for the WFG problems.

For assessing performance, we selected three performance indicators: the hypervolume (HV) [13], the Modified Inverted Generational Distance (IGD<sup>+</sup>) [5], and the  $s$ -energy [3]. The two first indicators assess both the convergence and the spread of the approximation set, while the third indicator measures only the diversity of the approximation set. The reference points used for the hypervolume, per problem, are the worst values found of the objective functions multiplied by 1.1.

To ensure a fair comparison, we defined the same set of parameters for each MOPSO. We set the swarm and archive size to 100 for the ZDT problems and to 91 for the WFG and DTLZ test problems. The mutation probability was set to  $p_m = 1/n$ , and the inertia weight was set to  $w = 0.1$ . Moreover, the MOPSOs stop after performing 2500 iterations.



(a) Hypervolume

(b) IGD<sup>+</sup>

(c) S-energy

**Fig. 2.** Distribution of ranks of SMPSO-E1 for each topology where rank 1 is the best and rank 5 is the worst.

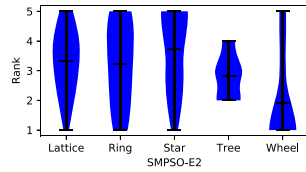
## 6.1 Methodology

We performed 30 independent runs of each MOPSO and normalized the resulting Pareto Front approximations. Then, we computed the indicators, normalized their values, and computed the corresponding means and standard deviations. Since we are dealing with stochastic algorithms, we also applied the Wilcoxon signed-rank test with a significance level of 5% to validate the statistical confidence of our results. We used the *mannwhitneyu* function from the SciPy Python library for this purpose.

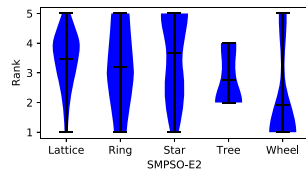
## 6.2 Experimental Results

Here, we present the comparison of SMPSO, SMPSO-E1, and SMPSO-E2 for each of the 5 topologies considered. Tables 1, 2, and 3 summarize the results for each indicator where the best values have a gray background, and the “\*” symbol means that this result is statistically significant. Figures 2 and 3 show the rank distribution among the topologies of SMPSO-E1 and SMPSO-E2, respectively. In this case, rank 1 is better than rank 5. In Fig. 2, the SMPSO-E1 with lattice, star, and tree topologies rank more frequently in the first places regarding the hypervolume and IGD<sup>+</sup> indicators. In contrast, the ring and wheel topologies rank more regularly in the last positions. This indicates that topologies with more connections promote the convergence of SMPSO-E1. Regarding the *s*-energy indicator, the lattice topology ranks more frequently in the best places, while the ring topology commonly ranks in the worst. Therefore, the lattice topology offers the best trade-off between convergence and diversity for SMPSO-E1.

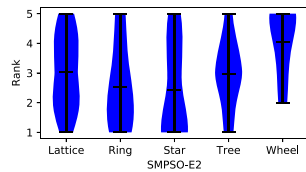
In the case of SMPSO-E2, we can see in Fig. 3 that the wheel topology ranks more frequently in the best places with respect to the hypervolume and IGD<sup>+</sup>, followed by the tree topology, followed by the ring and lattice topologies, and finally, by the star topology. It is worth noting that topologies with fewer connections have better values in the convergence indicators. Regarding the *s*-energy indicator, the star topology ranks more frequently in the first places, followed by the ring topology, and then the tree and the lattice topologies. Ultimately, the wheel topology ranks more often in the worst places. In this case, we cannot define a topology that provides the best possible trade-off between convergence and diversity.



(a) Hypervolume



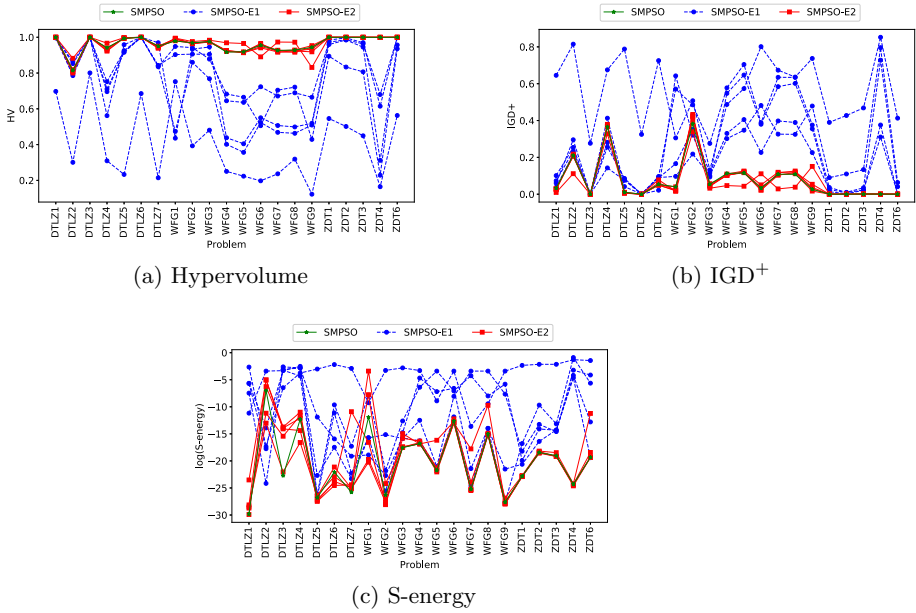
(b) IGD<sup>+</sup>



(c) S-energy

**Fig. 3.** Distribution of ranks of SMPSO-E2 for each topology where rank 1 is the best and rank 5 is the worst.





**Fig. 4.** Indicator values of SMPSO, SMPSO-E1, and SMPSO-E2 for each problem. Lower values are preferred for  $s$ -energy and  $IGD^+$ , while higher values are preferred for the hypervolume. (Color figure online)

Figure 4 compares the performance of the MOPSOs in each problem. The blue and red connected lines denote the behavior of SMPSO-E1 and SMPSO-E2, respectively, for each topology. Furthermore, the green line represents the original SMPSO. We can see that in Figs. 4b and 4c, most of the blue lines are above the green and red lines. Conversely, in Fig. 4a, all the lines are below the green and red lines. Therefore, it is clear that SMPSO-E1 performs worse than SMPSO-E2 and SMPSO.

Finally, in Tables 1 and 2, we can see that SMPSO-E2 with a wheel topology has the best performance with respect to  $IGD^+$  and the hypervolume. Besides, in Table 3, SMPSO-E2 with a star topology performs better with respect to  $s$ -energy, but the difference is not statistically significant.

**Table 1.** Mean and standard deviation of the HV indicator for SMPSO, SMPSO-E1, and SMPSO-E2. The best values are highlighted in gray, and “\*” indicates that the results are statistically significant

	SMPSO	SMPSO-E1					SMPSO-E2				
		Lattice	Ring	Star	Tree	Wheel	Lattice	Ring	Star	Tree	Wheel
<b>DTLZ1</b>	0.99999 (0.000)	0.99697 (0.003)	0.69748 (0.303)	0.99966 (0.001)	0.99967 (0.001)	0.99633 (0.014)	0.99999 (0.000)	0.99999 (0.000)	0.99999 (0.000)	0.99999 (0.000)	1.00000 (0.000)*
<b>DTLZ2</b>	0.81405 (0.038)	0.79588 (0.053)	0.30091 (0.135)	0.86375 (0.093)	0.85269 (0.069)	0.78544 (0.086)	0.81054 (0.041)	0.80032 (0.051)	0.80880 (0.037)	0.81901 (0.037)	0.88217 (0.064)
<b>DTLZ3</b>	1.00000 (0.000)	1.00000 (0.000)	0.80084 (0.254)	0.99999 (0.000)	1.00000 (0.000)	0.99999 (0.000)	1.00000 (0.000)	1.00000 (0.000)	1.00000 (0.000)	1.00000 (0.000)	1.00000 (0.000)
<b>DTLZ4</b>	0.94139 (0.019)	0.56177 (0.095)	0.30938 (0.150)	0.75153 (0.117)	0.71058 (0.119)	0.69577 (0.100)	0.93544 (0.016)	0.92248 (0.019)	0.93991 (0.017)	0.92746 (0.015)	0.96688 (0.015)*
<b>DTLZ5</b>	0.99326 (0.002)	0.95838 (0.006)	0.23308 (0.104)	0.92515 (0.024)	0.91583 (0.014)	0.91969 (0.020)	0.99601 (0.002)	0.99702 (0.001)	0.99270 (0.003)	0.99633 (0.001)	0.99760 (0.014)*
<b>DTLZ6</b>	0.99997 (0.000)	0.99993 (0.000)	0.68536 (0.277)	0.99955 (0.001)	0.99971 (0.000)	0.99920 (0.001)	0.99997 (0.000)	0.99997 (0.000)	0.99997 (0.000)	0.99997 (0.000)	0.99997 (0.000)
<b>DTLZ7</b>	0.95036 (0.027)	0.96958 (0.021)*	0.21446 (0.132)	0.83502 (0.055)	0.84259 (0.052)	0.83981 (0.065)	0.93912 (0.019)	0.95026 (0.027)	0.94900 (0.027)	0.94681 (0.018)	0.94085 (0.026)
<b>WFG1</b>	0.97934 (0.005)	0.43578 (0.217)	0.75196 (0.070)	0.94859 (0.015)	0.90301 (0.038)	0.47389 (0.217)	0.99078 (0.003)	0.99114 (0.002)	0.98043 (0.003)	0.99183 (0.002)	0.99495 (0.004)*
<b>WFG2</b>	0.96900 (0.010)	0.93333 (0.018)	0.39314 (0.170)	0.94195 (0.026)	0.90560 (0.027)	0.85999 (0.052)	0.96614 (0.010)	0.96408 (0.012)	0.96622 (0.012)	0.96427 (0.010)	0.97596 (0.014)*
<b>WFG3</b>	0.97586 (0.009)	0.94573 (0.036)	0.48072 (0.155)	0.87778 (0.048)	0.90471 (0.031)	0.76816 (0.144)	0.97749 (0.011)	0.97382 (0.012)	0.97226 (0.012)	0.97692 (0.009)	0.98341 (0.011)*
<b>WFG4</b>	0.91697 (0.020)	0.43842 (0.111)	0.25039 (0.110)	0.68273 (0.097)	0.64494 (0.065)	0.40172 (0.168)	0.92541 (0.016)	0.92125 (0.018)	0.92499 (0.020)	0.92351 (0.016)	0.96908 (0.020)*
<b>WFG5</b>	0.91891 (0.018)	0.40458 (0.127)	0.22354 (0.100)	0.66482 (0.070)	0.63603 (0.072)	0.35667 (0.153)	0.91564 (0.017)	0.91365 (0.019)	0.91434 (0.015)	0.91843 (0.016)	0.96516 (0.020)*
<b>WFG6</b>	0.95530 (0.059)	0.54999 (0.083)	0.19753 (0.085)	0.50558 (0.107)	0.72334 (0.067)	0.52365 (0.089)	0.96534 (0.017)	0.94079 (0.073)	0.94778 (0.060)	0.96042 (0.018)	0.89054 (0.100)
<b>WFG7</b>	0.92619 (0.017)	0.50668 (0.103)	0.23671 (0.101)	0.70426 (0.092)	0.67143 (0.059)	0.46856 (0.130)	0.91768 (0.015)	0.91782 (0.021)	0.92472 (0.018)	0.92754 (0.012)	0.97322 (0.015)*
<b>WFG8</b>	0.92835 (0.017)	0.49882 (0.101)	0.31892 (0.109)	0.72133 (0.082)	0.68940 (0.059)	0.46319 (0.111)	0.91636 (0.020)	0.92307 (0.015)	0.92857 (0.013)	0.92391 (0.017)	0.97240 (0.016)*
<b>WFG9</b>	0.94363 (0.057)	0.51928 (0.074)	0.12279 (0.063)	0.42907 (0.118)	0.66481 (0.086)	0.50926 (0.113)	0.94353 (0.054)	0.91859 (0.089)	0.95324 (0.016)	0.93372 (0.067)	0.83133 (0.114)
<b>ZDT1</b>	0.99974 (0.000)	0.98460 (0.006)	0.54580 (0.190)	0.89412 (0.037)	0.97277 (0.011)	0.95933 (0.035)	0.99980 (0.000)	0.99983 (0.000)	0.99972 (0.000)	0.99981 (0.000)	0.99989 (0.000)*
<b>ZDT2</b>	0.99977 (0.000)	0.99640 (0.001)	0.50140 (0.189)	0.83360 (0.072)	0.98613 (0.006)	0.98471 (0.010)	0.99982 (0.000)	0.99983 (0.000)	0.99976 (0.000)	0.99981 (0.000)	0.99982 (0.000)
<b>ZDT3</b>	0.99973 (0.000)	0.96979 (0.012)	0.44926 (0.190)	0.80671 (0.083)	0.96309 (0.016)	0.94488 (0.039)	0.99980 (0.000)	0.99982 (0.000)	0.99973 (0.000)	0.99982 (0.000)	0.99988 (0.000)*
<b>ZDT4</b>	0.99946 (0.000)	0.61466 (0.213)	0.16460 (0.109)	0.22890 (0.148)	0.67996 (0.198)	0.31222 (0.140)	0.99964 (0.000)	0.99970 (0.000)	0.99946 (0.000)	0.99970 (0.000)	0.99982 (0.000)*
<b>ZDT6</b>	0.99984 (0.000)	0.99970 (0.000)	0.56279 (0.222)	0.93559 (0.192)	0.99724 (0.003)	0.95678 (0.048)	0.99988 (0.000)	0.99992 (0.000)	0.99983 (0.000)	0.99990 (0.000)	0.99983 (0.000)

**Table 2.** Mean and standard deviation of the IGD<sup>+</sup> indicator for SMPSO, SMPSO-E1, and SMPSO-E2. The best values are highlighted in gray, and “\*” represents that the results are statistically significant

	SMPSO	SMPSO-E1					SMPSO-E2				
		Lattice	Ring	Star	Tree	Wheel	Lattice	Ring	Star	Tree	Wheel
<b>DTLZ1</b>	0.03301 (0.008)	0.10113 (0.034)	0.64555 (0.235)	0.03768 (0.019)	0.07282 (0.028)	0.06023 (0.034)	0.02695 (0.006)	0.02469 (0.007)	0.03161 (0.008)	0.02813 (0.008)	0.00964 (0.007)*
<b>DTLZ2</b>	0.21076 (0.034)	0.25408 (0.063)	0.81479 (0.101)	0.21706 (0.094)	0.23453 (0.069)	0.29559 (0.088)	0.21078 (0.041)	0.21764 (0.049)	0.20805 (0.038)	0.20568 (0.043)	0.11160 (0.053)*
<b>DTLZ3</b>	0.00010 (0.000)	0.00025 (0.000)	0.27655 (0.000)	0.00039 (0.000)	0.00063 (0.000)	0.00050 (0.000)	0.00009 (0.000)	0.00009 (0.000)	0.00011 (0.000)	0.00008 (0.000)	0.00006 (0.000)*
<b>DTLZ4</b>	0.36628 (0.058)	0.41288 (0.122)	0.67536 (0.157)	0.14286 (0.101)*	0.25582 (0.117)	0.28146 (0.098)	0.36892 (0.036)	0.37913 (0.045)	0.36749 (0.050)	0.37046 (0.032)	0.32834 (0.045)
<b>DTLZ5</b>	0.01056 (0.004)	0.04157 (0.116)	0.78828 (0.027)	0.07523 (0.048)	0.08581 (0.017)	0.08063 (0.019)	0.00810 (0.003)	0.00711 (0.002)	0.01042 (0.003)	0.00645 (0.003)	0.00549 (0.003)
<b>DTLZ6</b>	0.00011 (0.000)	0.00015 (0.000)	0.32484 (0.307)	0.00056 (0.001)	0.00041 (0.000)	0.00092 (0.001)	0.00011 (0.000)	0.00011 (0.000)	0.00011 (0.000)	0.00011 (0.000)	0.00011 (0.000)
<b>DTLZ7</b>	0.05341 (0.028)	0.02154 (0.011)*	0.72537 (0.168)	0.09052 (0.048)	0.08785 (0.041)	0.09626 (0.056)	0.06084 (0.017)	0.04887 (0.027)	0.05003 (0.020)	0.05035 (0.017)	0.07792 (0.025)
<b>WFG1</b>	0.04042 (0.009)	0.64259 (0.198)	0.30569 (0.067)	0.09692 (0.035)	0.16581 (0.056)	0.57108 (0.219)	0.02178 (0.008)	0.02101 (0.008)	0.04016 (0.007)	0.01833 (0.006)	0.01592 (0.011)
<b>WFG2</b>	0.38268 (0.110)	0.32075 (0.065)	0.50751 (0.148)	0.21807 (0.115)*	0.35269 (0.098)	0.48516 (0.206)	0.43306 (0.115)	0.42087 (0.129)	0.40543 (0.135)	0.42631 (0.117)	0.33968 (0.138)
<b>WFG3</b>	0.05400 (0.023)	0.09340 (0.072)	0.13147 (0.059)	0.10034 (0.095)	0.11614 (0.070)	0.27577 (0.196)	0.04774 (0.020)	0.04590 (0.022)	0.05720 (0.025)	0.04256 (0.023)	0.03254 (0.018)*
<b>WFG4</b>	0.11260 (0.030)	0.54752 (0.124)	0.48730 (0.102)	0.30173 (0.091)	0.33005 (0.071)	0.57747 (0.220)	0.10234 (0.030)	0.10968 (0.029)	0.10192 (0.029)	0.10363 (0.030)	0.04687 (0.034)*
<b>WFG5</b>	0.11549 (0.020)	0.64706 (0.138)	0.57343 (0.070)	0.34744 (0.072)	0.40533 (0.069)	0.70434 (0.165)	0.12136 (0.023)	0.12583 (0.023)	0.12115 (0.019)	0.11680 (0.023)	0.04322 (0.025)*
<b>WFG6</b>	0.03305 (0.063)	0.37975 (0.076)	0.80127 (0.107)	0.48182 (0.105)	0.22680 (0.067)	0.38804 (0.081)	0.02238 (0.008)	0.05078 (0.087)	0.03640 (0.062)	0.02242 (0.010)	0.11099 (0.120)
<b>WFG7</b>	0.10877 (0.023)	0.58401 (0.128)	0.67342 (0.125)	0.32610 (0.106)	0.39782 (0.067)	0.63484 (0.167)	0.11558 (0.021)	0.11913 (0.029)	0.10563 (0.020)	0.10341 (0.019)	0.02883 (0.019)*
<b>WFG8</b>	0.10817 (0.025)	0.60173 (0.119)	0.63615 (0.108)	0.32510 (0.092)	0.38945 (0.065)	0.63409 (0.133)	0.12590 (0.027)	0.12015 (0.022)	0.11112 (0.018)	0.11776 (0.025)	0.03815 (0.019)*
<b>WFG9</b>	0.02673 (0.049)	0.35521 (0.050)	0.73704 (0.082)	0.47878 (0.084)	0.22603 (0.086)	0.37548 (0.080)	0.02877 (0.054)	0.05392 (0.095)	0.01700 (0.007)	0.03652 (0.069)	0.15095 (0.121)
<b>ZDT1</b>	0.00039 (0.000)	0.01323 (0.005)	0.39003 (0.203)	0.09011 (0.030)	0.02464 (0.010)	0.03646 (0.032)	0.00035 (0.000)	0.00033 (0.000)	0.00041 (0.000)	0.00031 (0.000)	0.00025 (0.000)
<b>ZDT2</b>	0.00021 (0.000)	0.00251 (0.001)	0.42733 (0.213)	0.10997 (0.046)	0.00930 (0.004)	0.01021 (0.007)	0.00018 (0.000)	0.00015 (0.000)	0.00021 (0.000)	0.00018 (0.000)	0.00009 (0.000)*
<b>ZDT3</b>	0.00103 (0.000)	0.02096 (0.007)	0.46837 (0.214)	0.13350 (0.057)	0.02279 (0.008)	0.03485 (0.026)	0.00093 (0.000)	0.00085 (0.000)	0.00106 (0.000)	0.00098 (0.000)	0.00078 (0.000)
<b>ZDT4</b>	0.00063 (0.000)	0.37585 (0.234)	0.85215 (0.130)	0.79590 (0.174)	0.31050 (0.215)	0.72769 (0.158)	0.00052 (0.000)	0.00048 (0.000)	0.00063 (0.000)	0.00047 (0.000)	0.00043 (0.000)
<b>ZDT6</b>	0.00019 (0.000)	0.00035 (0.000)	0.41284 (0.224)	0.06345 (0.192)	0.00269 (0.003)	0.04083 (0.044)	0.00014 (0.000)	0.00010 (0.000)	0.00017 (0.000)	0.00012 (0.000)	0.00020 (0.000)

**Table 3.** Mean and standard deviation of the  $s$ -energy indicator for SMPSO, SMPSO-E1, and SMPSO-E2. The best values are highlighted in gray, and “\*” represents that the results are statistically significant

	SMPSO	SMPSO-E1					SMPSO-E2				
		Lattice	Ring	Star	Tree	Wheel	Lattice	Ring	Star	Tree	Wheel
<b>DTLZ1</b>	1.141e-13 (0.000)	3.506e-03 (0.018)	7.015e-02 (0.239)	3.376e-03 (0.018)	5.690e-04 (0.002)	1.401e-05 (0.000)	3.346e-13 (0.000)	6.078e-13 (0.000)	4.183e-13 (0.000)	1.031e-13 (0.000)	6.109e-11 (0.000)
<b>DTLZ2</b>	9.007e-04 (0.005)	9.139e-07 (0.000)	2.869e-08 (0.000)	1.957e-08 (0.000)	3.223e-11 (0.179)	3.337e-02 (0.000)	1.391e-05 (0.000)	6.640e-03 (0.011)	2.229e-06 (0.000)	1.896e-03 (0.007)	6.427e-03 (0.011)
<b>DTLZ3</b>	1.355e-10 (0.000)	3.907e-02 (0.180)	1.544e-03 (0.008)	4.636e-02 (0.162)	7.300e-02 (0.110)	3.635e-02 (0.122)	1.965e-07 (0.000)	1.093e-06 (0.000)	2.665e-10 (0.000)	9.056e-07 (0.000)	7.648e-07 (0.000)
<b>DTLZ4</b>	4.563e-06 (0.000)	1.350e-02 (0.037)	2.343e-02 (0.053)	5.861e-02 (0.162)	5.994e-02 (0.192)	8.029e-02 (0.203)	9.265e-06 (0.000)	1.679e-05 (0.000)	6.177e-08 (0.000)	5.557e-06 (0.000)	5.653e-07 (0.000)
<b>DTLZ5</b>	2.277e-12 (0.000)	2.626e-12 (0.000)	4.879e-02 (0.000)	6.785e-06 (0.000)	4.160e-12 (0.000)	1.408e-10 (0.000)	2.918e-12 (0.000)	1.302e-12 (0.000)	3.706e-12 (0.000)	1.162e-12 (0.180)	1.657e-12 (0.000)
<b>DTLZ6</b>	2.565e-10 (0.000)	1.483e-05 (0.000)	1.115e-01 (0.290)	1.209e-07 (0.000)	6.510e-05 (0.000)	2.566e-08 (0.000)	6.661e-10 (0.000)	4.564e-11 (0.000)	1.127e-10 (0.000)	2.188e-11 (0.000)	3.971e-11 (0.000)
<b>DTLZ7</b>	6.239e-12 (0.000)	2.101e-10 (0.000)	5.481e-02 (0.205)	4.958e-09 (0.000)	3.105e-08 (0.000)	7.612e-11 (0.000)	1.291e-11 (0.000)	1.447e-11 (0.000)	1.245e-11 (0.000)	2.506e-11 (0.000)	1.847e-05 (0.000)
<b>WFG1</b>	6.707e-06 (0.000)	1.523e-07 (0.000)	9.602e-05 (0.000)	6.198e-09 (0.000)	3.914e-04 (0.000)	1.454e-07 (0.000)	2.714e-09 (0.000)	4.275e-04 (0.002)	1.621e-09 (0.000)	3.333e-02 (0.180)	6.297e-08 (0.000)*
<b>WFG2</b>	3.857e-12 (0.000)	7.653e-12 (0.000)	3.792e-02 (0.179)	1.370e-10 (0.000)	3.513e-10 (0.000)	2.660e-07 (0.000)	1.913e-12 (0.000)	6.343e-13 (0.000)	9.577e-13 (0.000)	3.085e-11 (0.000)	7.659e-13 (0.000)
<b>WFG3</b>	2.375e-08 (0.000)	3.806e-07 (0.000)	6.042e-02 (0.212)	1.434e-07 (0.000)	3.310e-06 (0.000)	1.295e-07 (0.000)	3.176e-07 (0.000)	2.546e-08 (0.000)	2.581e-08 (0.000)	2.816e-08 (0.000)	3.365e-07 (0.000)
<b>WFG4</b>	5.093e-08 (0.000)	9.281e-03 (0.050)	3.680e-02 (0.180)	8.530e-08 (0.000)	1.686e-03 (0.009)	3.754e-06 (0.000)	4.755e-08 (0.000)	4.844e-08 (0.000)	6.139e-08 (0.000)	4.854e-08 (0.000)	8.009e-08 (0.000)
<b>WFG5</b>	3.552e-10 (0.000)	7.542e-04 (0.004)	1.361e-04 (0.001)	5.059e-10 (0.000)	3.333e-02 (0.180)	7.000e-10 (0.000)	9.346e-08 (0.000)	3.491e-10 (0.000)	2.681e-10 (0.000)	5.375e-10 (0.000)	5.671e-10 (0.000)
<b>WFG6</b>	3.291e-06 (0.000)	1.376e-03 (0.007)	3.336e-02 (0.180)	7.085e-06 (0.000)	8.563e-04 (0.004)	3.146e-04 (0.002)	3.014e-06 (0.000)	2.469e-06 (0.000)	2.242e-06 (0.000)	2.407e-06 (0.000)	4.877e-06 (0.000)
<b>WFG7</b>	1.089e-11 (0.000)	4.965e-10 (0.000)	1.204e-06 (0.000)	2.080e-11 (0.000)	1.406e-02 (0.076)	3.333e-02 (0.180)	1.054e-11 (0.000)	8.564e-12 (0.000)	1.898e-08 (0.000)	1.033e-11 (0.000)	4.042e-11 (0.000)
<b>WFG8</b>	2.987e-07 (0.000)	8.504e-07 (0.000)	7.951e-05 (0.000)	8.527e-07 (0.000)	3.385e-04 (0.002)	3.336e-02 (0.180)	2.860e-07 (0.000)	2.381e-07 (0.000)	5.952e-05 (0.000)	2.796e-07 (0.000)	3.430e-07 (0.000)
<b>WFG9</b>	1.001e-12 (0.000)	4.561e-10 (0.000)	3.333e-02 (0.180)	1.009e-12 (0.000)	2.981e-03 (0.016)	4.756e-04 (0.003)	7.260e-13 (0.000)	7.276e-13 (0.000)	9.823e-13 (0.000)	7.737e-13 (0.000)	2.156e-12 (0.000)
<b>ZDT1</b>	1.222e-10 (0.000)	1.104e-09 (0.208)	9.597e-02 (0.208)	5.123e-08 (0.000)	2.253e-09 (0.000)	1.240e-08 (0.000)	1.165e-10 (0.000)	1.175e-10 (0.000)	1.130e-10 (0.000)	1.215e-10 (0.000)	1.374e-10 (0.000)
<b>ZDT2</b>	9.719e-09 (0.000)	7.584e-08 (0.000)	1.170e-01 (0.293)	6.223e-05 (0.000)	7.512e-07 (0.000)	1.764e-06 (0.000)	1.144e-08 (0.000)	9.838e-09 (0.000)	8.591e-09 (0.000)	9.851e-09 (0.000)	1.269e-08 (0.000)
<b>ZDT3</b>	5.414e-09 (0.000)	6.296e-07 (0.000)	1.171e-01 (0.221)	1.945e-06 (0.000)	5.979e-07 (0.000)	4.632e-07 (0.000)	4.847e-09 (0.000)	4.776e-09 (0.000)	4.701e-09 (0.000)	4.816e-09 (0.000)	9.529e-09 (0.000)
<b>ZDT4</b>	2.786e-11 (0.000)	9.603e-03 (0.027)	2.743e-01 (0.179)	4.082e-01 (0.249)	1.501e-02 (0.079)	3.949e-02 (0.108)	2.426e-11 (0.000)	2.049e-11 (0.000)	2.770e-11 (0.000)	2.624e-11 (0.000)	2.999e-11 (0.000)
<b>ZDT6</b>	4.016e-09 (0.000)	3.719e-09 (0.000)	2.354e-01 (0.291)	3.585e-03 (0.015)	2.762e-06 (0.000)	1.615e-02 (0.087)	4.053e-09 (0.000)	3.969e-09 (0.000)	3.975e-09 (0.000)	9.804e-09 (0.000)	1.334e-05 (0.000)

## 7 Conclusions and Future Work

In this work, we proposed two topology handling schemes that differ in the place from which the leader is taken, and we implemented them in SMPSO. Moreover, using the resulting MOPSOs (SMPSO-E1 and SMPSO-E2), we performed an experimental analysis of the influence of the topology in the performance of a MOPSO.

The experiments show that a scheme that uses information from the external archive perform better than a scheme that uses information from the swarm. Furthermore, the same topology will influence the performance of a MOPSO

in a different manner if the topology handling scheme is changed. On the other hand, our experiments also indicate that the fewer topology connections SMPSE2 has, the better its convergence is. This effect could be because the particles in a topology with many connections could try to go in multiple directions due to the existence of multiple optimal solutions, causing the MOPSO to converge. Conversely, if the topology has few connections, the information flows slowly and the particles move to specific optimal solutions. Furthermore, the wheel topology in SMPSE2 performs better than SMPSE1 and SMPSE0.

Therefore, the right selection of a topology can indeed improve the performance of a MOPSO. In SMPSE0 and SMPSE1, the swarm topology had little influence in the distribution of solutions. Thus, a topology handling scheme that focuses on this topic could be worth developing.

## References

1. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. *Complex Syst.* **9**, 115–148 (1995)
2. 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*. Advanced Information and Knowledge Processing, 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)
3. Hardin, D., Saff, E.: Discretizing manifolds via minimum energy points. *Not. Am. Math. Soc.* **51**(10), 1186–1194 (2004)
4. 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)
5. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Difficulties in specifying reference points to calculate the inverted generational distance for many-objective optimization problems. In: *2014 IEEE Symposium on Computational Intelligence in Multi-criteria Decision-Making (MCDM 2014)*, pp. 170–177, December 2014
6. Kennedy, J.: Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999)*, vol. 3, pp. 1931–1938, July 1999
7. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: *Proceedings of the 1995 IEEE International Conference on Neural Networks (ICNN 1995)*, vol. 4, pp. 1942–1948 (1995)
8. Mendes, R.: Population topologies and their influence in particle swarm performance. Ph.D. thesis, Departamento de Informática, Escola de Engenharia, Universidade do Minho, April 2004
9. Nebro, A.J., Durillo, J.J., García-Nieto, J., Coello, C.A.C., Luna, F., Alba, E.: SMPSE: a new PSO-based metaheuristic for multi-objective optimization. In: *2009 IEEE Symposium on Computational Intelligence in Multi-criteria Decision-Making (MCDM 2009)*, pp. 66–73, March 2009
10. Parsopoulos, K.E., Vrahatis, M.N.: Multi-objective particles swarm optimization approaches. In: *Multi-objective Optimization in Computational Intelligence: Theory and Practice*, pp. 20–42. IGI Global (2008)

11. Taormina, R., Chau, K.: Neural network river forecasting with multi-objective fully informed particle swarm optimization. *J. Hydroinform.* **17**(1), 99–113 (2014)
12. Yamamoto, M., Uchitane, T., Hatanaka, T.: An experimental study for multi-objective optimization by particle swarm with graph based archive. In: *Proceedings of SICE Annual Conference (SICE 2012)*, pp. 89–94, August 2012
13. Zitzler, E.: *Evolutionary algorithms for multiobjective optimization: methods and applications*. Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Suiza, November 1999
14. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. *Evol. Comput.* **8**(2), 173–195 (2000)



# Visualising Evolution History in Multi- and Many-objective Optimisation

Mathew J. Walter, David J. Walker, and Matthew J. Craven<sup>(✉)</sup>

School of Engineering, Computing and Mathematics, University of Plymouth,  
Plymouth PL4 8AA, UK

{mathew.walter,david.walker,matthew.craven}@plymouth.ac.uk

**Abstract.** Evolutionary algorithms are widely used to solve optimisation problems. However, challenges of transparency arise in both visualising the processes of an optimiser operating through a problem and understanding the problem features produced from many-objective problems, where comprehending four or more spatial dimensions is difficult. This work considers the visualisation of a population as an optimisation process executes. We have adapted an existing visualisation technique to multi- and many-objective problem data, enabling a user to visualise the EA processes and identify specific problem characteristics and thus providing a greater understanding of the problem landscape. This is particularly valuable if the problem landscape is unknown, contains unknown features or is a many-objective problem. We have shown how using this framework is effective on a suite of multi- and many-objective benchmark test problems, optimising them with NSGA-II and NSGA-III.

**Keywords:** Visualisation · Evolutionary computation · Multi-objective optimisation

## 1 Introduction

Optimisation problems abound in science and industry, and in recent decades a plethora of approaches have arisen to solve them. A prominent example are *evolutionary algorithms* (EAs). An EA takes an initial population and uses nature-inspired operators to perturb the solutions towards optimal solution (or solutions). As well as solving an optimisation problem, it is important that the processes with which they are generated are understandable by non-expert problem owners, and often this is not the case: therein lies a challenge of transparency. Visualisation is a natural approach to addressing this issue, exposing the solutions, and the mechanisms used to generate them, to the end user.

This paper expands upon [4], which compared the extent to which dimension reduction techniques preserved population movements and the exploration-exploitation trade-off. It also proposes two compact visualisations, one of which we extend to visualise the search history of an EA optimising a multi- and many-objective problem. We use this to identify specific characteristics of problems as

well as identifying the population dynamic through the evolution process. This paper offers the following novel contributions:

1. The method is applied to a suite of multi- and many-objective test problems containing a wider set of problem features that can be visualised than the single-objective problems used in [4].
2. Specific problem features that can be identified through the proposed visualisation are identified and examples are shown.
3. The method is applied in both the search space and objective space. Specific problem characteristics and population movements can be more prominent in a low-dimensional embedding of a particular space (i.e. discontinuities in the search space may only be able to be identified in the objective space embedding visualisation for a particular problem).

The remainder of this paper is structured as follows: In Sect. 2, relevant formal definitions are introduced. We review existing work on visualising many/multi-objective optimisation from the literature, and we acknowledge the paper this work extends. Section 3 contains details of the dimension reduction techniques, the test problem suite, problem features and a summary of the methodology implemented for visualisation. The experimental setup, containing details of the parameters used in this experiment are highlighted in Sect. 4. Section 5 hosts the results and analysis and provides a discussion of the many-objective problems. We conclude and discuss future work in Sect. 6.

## 2 Background

A multi-objective optimisation problem comprises  $M$  competing objectives, such that a solution  $\mathbf{x}$  is quantified by an objective vector  $\mathbf{y}$  with  $M$  elements:

$$\mathbf{y} = (f_1(\mathbf{x}), \dots, f_M(\mathbf{x})) \quad (1)$$

such that  $\mathbf{x} \in \Omega$ ,  $\mathbf{y} \in \Lambda$

where  $\Omega$  is the *search space* and  $\Lambda$  is the *objective space*. Many-objective optimisation problem comprises four or more competing objectives (and thus  $M \geq 4$  for such problems). The task of a multi-objective evolutionary algorithm (MOEA) and many-objective evolutionary algorithm (MaOEA) is to optimise a problem comprising a set of  $M$  conflicting objectives to which there can be no solution that simultaneously optimises all  $M$  objectives. Solutions are compared using the dominance relation, whereby solution  $\mathbf{y}_i$  dominates solution  $\mathbf{y}_j$  if it is no worse than  $\mathbf{y}_j$  on any objective and better on at least one. More formally, assuming a minimisation problem without loss of generality:

$$\mathbf{y}_i \prec \mathbf{y}_j \iff \forall m (y_{im} \leq y_{jm}) \wedge \exists m (y_{im} < y_{jm}). \quad (2)$$

If neither  $\mathbf{y}_i$  dominates  $\mathbf{y}_j$ , nor  $\mathbf{y}_j$  dominates  $\mathbf{y}_i$ , then the solutions are *mutually non-dominating*. A solution with no dominating solutions is *non-dominated*. The goal of a MaOEA/MOEA is to identify the Pareto set, the set of feasible solutions that cannot be dominated. The objective space image is called the Pareto front.



## 2.1 Previous Visualising Search History Literature

Rather than visualising the Pareto set and/or Pareto front, visualising the search population can generate more useful information to the decision-maker (DM). The DM will often be interested not only in the non-dominated solutions but also the mechanism from which they are generated. The existing literature concerning visualising population movements in evolutionary multi-objective optimisation (EMO) is minimal. One study that does consider population movement presents a visual method for benchmarking the performance of EAs. The method is used to illustrate a range of good and bad performance characteristics [14, 15]. Other examples of visual methods for examining algorithm parameters are [2, 11]. Existing work from [1] visualises search history in EMO.

Other papers are concerned with visualising the non-dominated solutions [13, 16]. However, these do not address a significant issue: the DM's comprehension of the algorithm's population movements. Further work that visualises the whole population is that of [9], which maps individuals from a high-dimensional objective space into a 2-D polar coordinate plot while preserving the Pareto dominance relationship. Like the studies referenced above, that work is concerned with configuring MaOEAs rather than characterising the problem landscape.

There are many choices of content to be visualised. For example, one could visualise the search space, the objective space or search process. This work considers visualising all three with multi/many-objective problems. As shown in Sect. 5, often more information can be captured when visualising both spaces simultaneously than by visualising a single space. Further, this work chooses to visualise the whole population rather than a subset (such as the Pareto front) to capture the maximum amount of population movement information from the visualisation. This paper pays particular attention to visualising the population as the population evolves through the algorithm.

## 3 Visualising Search History

One of the difficulties that arise from visualising many-objective optimisation data is being able to comprehend four or more spatial dimensions visually. In order to make a visualisation tool applicable to the class of multi/many-objective EAs, a dimension reduction technique needs to be applied to the population. The mapping  $\Pi$  is a dimension reduction mapping if  $\Pi : \mathbb{R}^m \rightarrow \mathbb{R}^n$ , where  $n < m$ . In this work, Multi-Dimensional Scaling (MDS) [12] is the chosen technique for dimension reduction; this is due to its effectiveness at maintaining population structure [4]. MDS is used to translate pairwise distances of the population individuals into a lower-dimension Cartesian space.

As discussed in Sect. 2, this work chooses to visualise the search space and the objective space. Through visualising the population evolving through the space, one is able to identify specific problem features. There are many choices of problem features that can be considered. In this work, we consider four:

- Local optimum - A local optimum is a solution that is optimal within a neighbouring set of candidate solutions.

- Modality - For the problems considered, the objective functions can be unimodal or multimodal. An objective function is unimodal if it has a single optimum, or multimodal if it has multiple local optima.
- Bias - A problem is biased if there's significant density variation of solutions in the objective space, given an even spread of solutions in parameter space.
- Disconnected Pareto optimum set/front - In this case, a problem has a Pareto set/front in disconnected regions.

In order to detect these problem features, the experiment requires a diverse choice of test problems to allow various properties of the individual problems to be identified in the visualisations; for this work the DTLZ test suite [7] is used. The problems contain many features which can be used to support the exposure of population movements. The application of the DTLZ test suite allows one to identify specific problem features from the population MDS - however, the DTLZ test suite is far from comprehensive. It is noted in [10] that the DTLZ test suite has several limitations such as: none of its problems feature fitness landscapes with flat regions, none of its problems are deceptive, none of its problems are (practically) nonseparable and the number of position parameters is always fixed relative to the number of objectives.

### 3.1 Visualising Search History Methodology

The visualisation method used herein is based on that defined by [4], visualising the search history of an optimiser once the optimisation process has completed. The optimiser results in a sequence of populations in which  $P_i$  is the population from the  $i$ -th generation, ranked according to its members' fitness values, where  $i \in \{1, \dots, n_{gen}\}$ , where  $n_{gen}$  is the total number of generations. This sequence of populations is concatenated into a single multiset, the dimensionality of which is reduced using multidimensional scaling (MDS) from  $M$  to 2. In all cases herein,  $M > 2$ . The resulting embedding is then used for visualisation, with the two embedded coordinates forming the  $x$  and  $y$  coordinates, and the generation number providing the value for the  $z$ -axis.

Within the visualisation, colour is used to illustrate the trade-off between search and exploitation, showing which mode of optimisation the algorithm is currently operating in. The work of [3, 4] is employed to determine to what level the set of all solutions at a particular generation is being explored or exploited. Exploration is inversely proportional to exploitation. This metric is applied to the visualisation in [4].

The exploration and exploitation metric is calculated for each generation. At each generation, the Euclidean distance between each pairwise individual in the population is calculated, and the minimum distance is saved from which the median minimum pairwise distance is calculated. At each generation, the minimum distance for each pairwise individual is compared against the overall median. Thus the individuals with lower distance are considered to be exploiting.

## 4 Experimental Setup

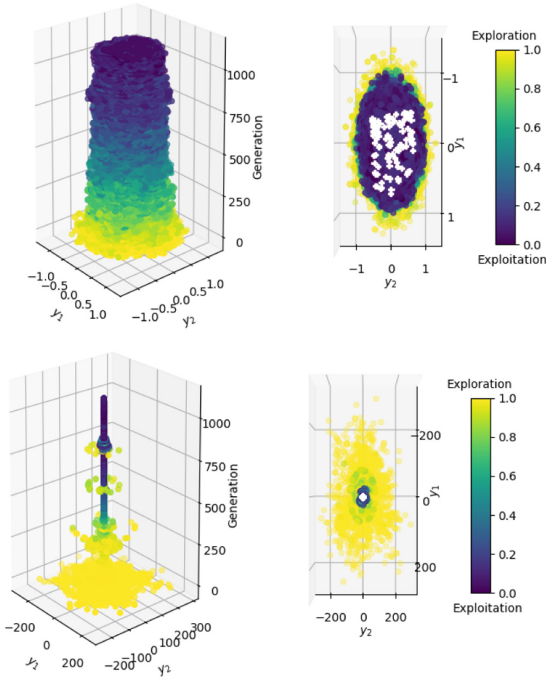
The experiment comprises of running EAs on five continuous problems from the DTLZ test suite [7], namely, DTLZ1-4 and DTLZ7. These five problems have real-valued decision variables lying in the region  $[0, 1]$ . The suggested number of decision variables is  $D = k + M - 1$ , where  $k = 5$  for DTLZ1,  $k = 20$  for DTLZ7, and  $k = 10$  for the remaining problems. The problems are scalable in the number of objectives; in this experiment, three and five objective problems are utilised. The 3-objective problems are optimised with NSGA-II [6] and the 5-objective problems with NSGA-III [5]. The crossover probability is 0.8, and the mutation probability is set to 0.1. The distribution index, controlling the size of the perturbation, in both cases is fixed (15 for SBX, 7 for polynomial mutation). The algorithm's runtime is 100,000 function evaluations for  $M = 3$  and 200,000 for  $M = 5$ . The visualisations are then produced as in Sect. 3.1.

Having generated data by optimising one of the problems with either NSGA-II or NSGA-III, the search and objective spaces are visualised. The MDS plots of both whole populations in the objective and search spaces are generated. The points in the visualisations are coloured according to the exploration-exploitation metric, except, however, the final generation in each visualisation which is plotted as a white cross. An analysis is then performed of the plots, with the intention of both identifying characteristics of the test problems and identifying the dynamics of the population as it evolves through the problem.

## 5 Results

### 5.1 Multi-objective Problems

Figure 1 illustrates results for optimising DTLZ1 in three objectives. The top panel visualises the search space, and the lower panel shows the corresponding objective space results. DTLZ1 is multimodal, and the effect of this on the search process can be seen in the result. As the population approaches optima, the population movements should decrease; by definition, this decreases the exploration measure. The MDS mapping would translate a decrease in population movements to an MDS population confined together and converged around  $(0, 0)$  in the  $(y_1, y_2)$  plane. From this perspective, the population distances can be observed to converge to approximately  $(0, 0)$  when finally reaching the global optima (this occurs in the final 250 generations in the objective space). However, before the objective space population converges to the global optimum, a subset of the population (identified as the 'rings' in Fig. 1) are exploring after diverging from a local optimum, whilst the remaining population are converging to the global optimum. This can be observed at approximately generations 250, 500 and 750. Furthermore, the population is coloured according to the exploration-exploitation metric. At the identified local optima, the population possesses lighter colours such as yellow and green, suggesting the points are exploring in an attempt to escape from the local optima.

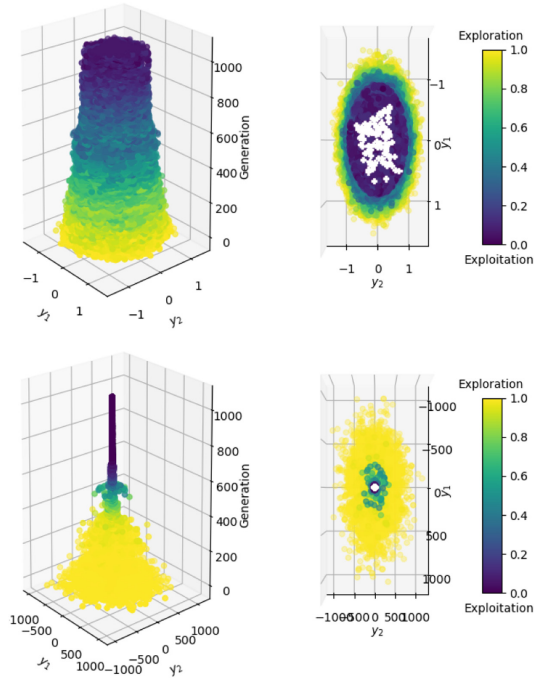


**Fig. 1.** DTLZ1, with the top illustrations showing the MDS reduced search space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric. (Colour figure online)

The final generation presents some structure from the final generation in the objective space (a triangular plane). This corresponds to earlier work using MDS to visualise many-objective populations, wherein it was shown to preserve the structure and, to an extent, geometry of a mutually non-dominating set [16]. As in both spaces, the exploration/exploitation seems to indicate that as the generations increase, exploration increases and hence exploitation decreases; this matches intuition.

DTLZ3 is also a multimodal problem, and similar results can be observed in Fig. 2 to that of DTLZ1. In the MDS reduced search space the population appears to converge, with the population distances reducing as the optimiser evolves. At around generation 400, it can be seen to diverge and converge again as the population encounters a local minimum. On the MDS reduced search space the final generation preserves some of the structure from the final generation in the objective space (the positive orthant of the unit sphere).

DTLZ2 and DTLZ4 are illustrated in Figs. 3 and 4 respectively. Both problems are unimodal, and there is usually much less to see as the problems are less challenging. For DTLZ2, both spaces appear to be cylindrical shapes with very little character; this is because the problems contain a simple search space and there is little to prevent an optimiser converging to the global optimum very

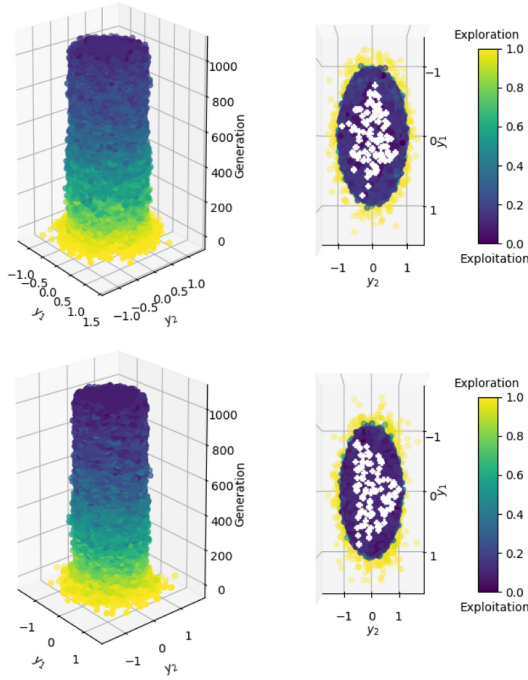


**Fig. 2.** DTLZ3, the top illustrations showing the MDS reduced search space. The bottom illustrations show the MDS reduced objective space.

quickly. This is reflected in the spaces. The final generation in the MDS space has preserved much of the final generation structure in the objective and search space. The objective space of DTLZ2 is similar to that of DTLZ3 and DTLZ4, and hence the final generations are all similar in structure. The transition from population exploration to exploitation is more gradual than the exploration-exploitation transition in the multimodal problems; this appears intuitive.

In the visualisations illustrated so far, it seems most of the information is contained within the MDS reduced objective space. In the case of DTLZ4 there appear to be more interesting characteristics of the plot in the MDS reduced search space, this can be seen in Fig. 4. In the decision space MDS the population appears to form circular ‘shockwaves’. This is because the search space contains a dense area of solutions next to the  $f_M/f_1$  plane. DTLZ4 is a biased problem, which increases the difficulty of a problem by making it harder to converge to and fully cover the Pareto front.

DTLZ7 (Fig. 5) is a mixed modality problem. Objectives  $f_{1:M-1}$  are unimodal and objective  $f_M$  is multimodal. This problem has disconnected Pareto-optimal regions in the search space. There appears to be more similarity with the unimodal objective problems. The disconnected regions of the Pareto front are shown in Fig. 6, and can be seen as they evolve through the search space. Note,



**Fig. 3.** DTLZ2, the top illustrations showing the MDS reduced search space. The bottom illustrations show the MDS reduced objective space.

the regions are clearly visible in the MDS reduced search space embedding of the final generation. This structure is not visible in the objective space visualisation, and this problem is an example of a case in which considering both spaces can yield useful information (in this case the objective space visualisation yields no new information, and is omitted due to lack of space).

Figure 7 illustrates the use of the hypervolume indicator [8] in parallel with the MDS visualisation in order to understand population movements. The hypervolume is a measure of performance widely used to assess the progress of a MOEA in terms of both convergence and diversity – in order to achieve the maximum possible hypervolume score the Pareto front approximation must converge to the Pareto front and cover it completely. In this case, the population appears to converge, in approximately the first 30 generations. Then up to around 400 generations, the population distances are small; subsequently, the population seems to converge to the optima. These changes in optimiser progress correspond to the changes that are visible in the objective space visualisation, indicating that they are highlighting the same artefact in the optimisation history.

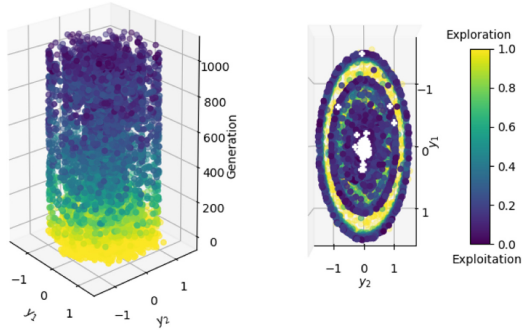


Fig. 4. DTLZ4, showing the MDS reduced search space.

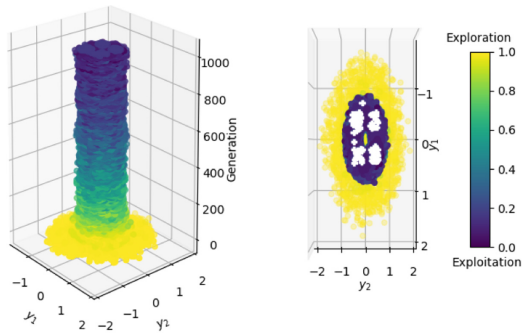


Fig. 5. DTLZ7, showing the MDS reduced search space.

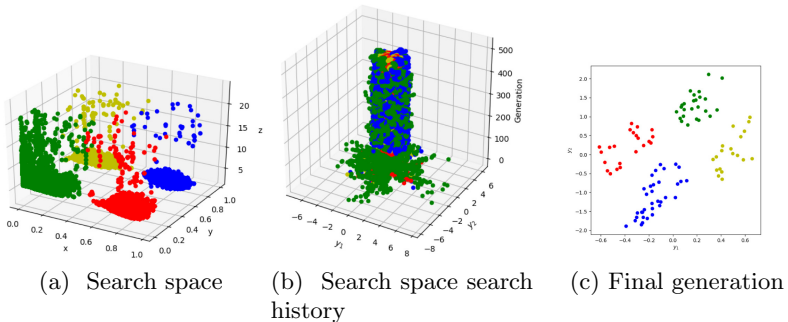
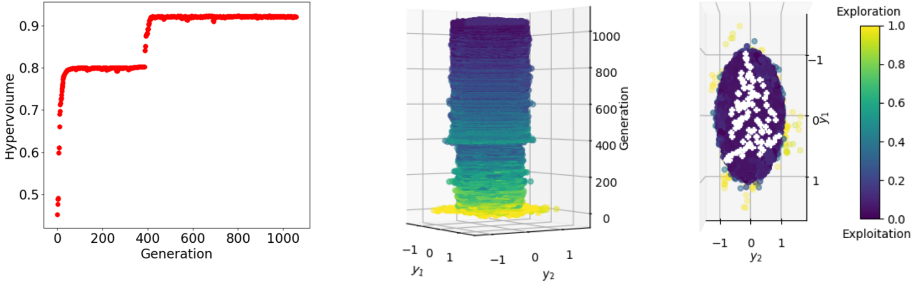


Fig. 6. Clustering coloured MDS reduced search space. For (a), axes  $x, y, z$  correspond to the three objectives. In (b) and (c), axes  $y_i$  correspond to the reduced MDS data axes.



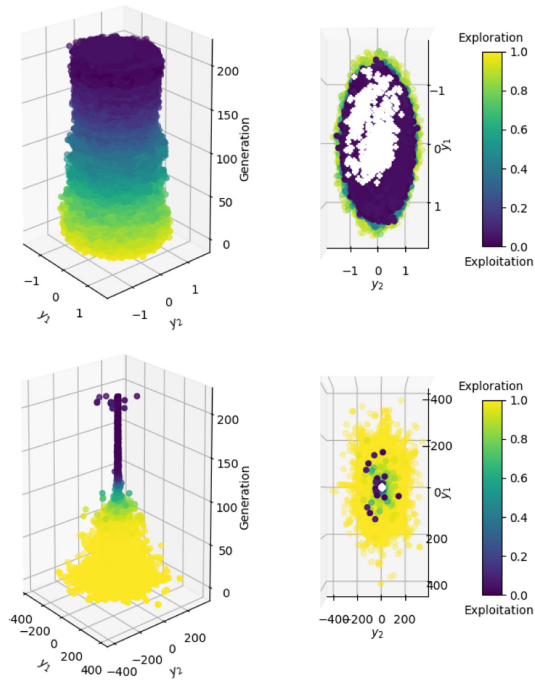
**Fig. 7.** The hypervolume of a three objective DTLZ4 problem, followed by the corresponding MDS reduced objective space plot.

## 5.2 Many-objective Problems

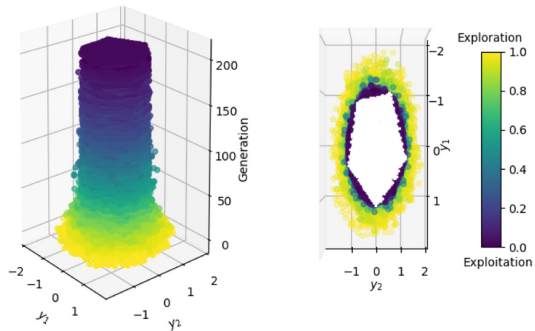
The framework is demonstrated on five objective problems, for which a larger number of function evaluations, and hence generations, are required. In this experiment, 200,000 function evaluations are run.

In Fig. 8, the DTLZ1 test problem appears to converge around an optimum at approximately generation 100. In the final generations the population diverges again after encountering another optimum; this is because some population distances have increased. We can see the population has not converged as quickly as in the three objective problem; this is intuitive, as the five objective problem is more difficult. DTLZ2 (Fig. 9) and DTLZ4 (Fig. 10) have a final generation which shows a very distinctive pentagon in the MDS reduced objective space, which demonstrates the final generation in the MDS visualisation maintains a similar structure to the final generation in the objective space. It should be noted: the pentagon is formed due to the five objective problem nature. For a problem comprising  $M$  objectives, one would expect to find a  $M$ -sided shape from the final generation MDS reduced objective space. DTLZ4 contains a ring around the MDS reduced search space. The population appears to form circular ‘shock-waves’. This is because the search space contains a dense area of solutions next to the  $f_M/f_1$  plane. In the MDS reduced search space of DTLZ7 (Fig. 11), the clusters of points become more difficult to observe than with the same problem in three objectives. The MDS reduced objective space, however, appears to show stacked ‘lines’, and shows how the NSGA-III algorithm operates on the population movements. NSGA-III uses a set of reference directions to maintain diversity among solutions, and the population appears to converge along these reference points. We therefore state, the problem characterisations that can be inferred from the visualisations are highly dependent on the employed algorithm. That is, the visualisations show the search behavior from which the problem characterisations can be seen only indirectly, leading to some visualisation features being an artifact of the algorithm.





**Fig. 8.** DTLZ1 with five objectives, top illustrations showing the MDS reduced search space. The bottom illustrations show the MDS reduced objective space.



**Fig. 9.** DTLZ2 with five objectives, showing the MDS reduced objective space.

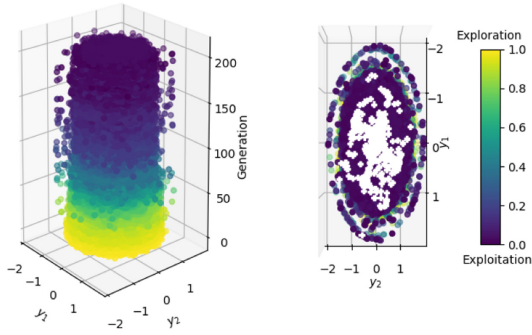


Fig. 10. DTLZ4 with five objectives, showing the MDS reduced search space.

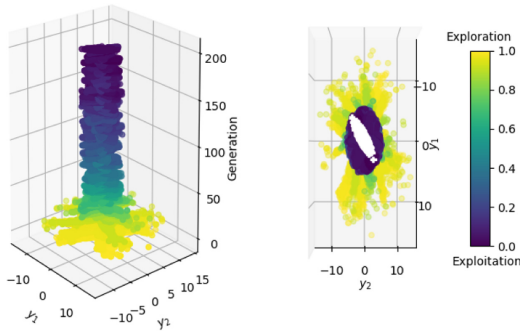


Fig. 11. DTLZ7 with five objectives, showing the MDS reduced objective space.

## 6 Conclusion

By extending the visualisation proposed in [4], we have identified specific problem characteristics through population movements that could be of value to the DM. This provides a better understanding of the problem landscape, and how the algorithm is performing, allowing the DM to make decisions based on how the optimisers are evolving solutions to the problem. We have shown how this framework can be used to locate some of the characteristics of a problem. For example, the framework has identified clusters caused by discontinuities in the Pareto front, can identify local optima, and allows one to see where the population approximately converges to the Pareto front. Ultimately, the visualisation illustrates how the population moves through the search space.

The approach can be used to identify important characteristics of a problem; this is particularly useful if the problem landscape is unknown, and contains unknown features. It is well known that visualising many-objective solutions is a challenge, and this work has shown how using this framework is effective for both multi- and many-objective problems. Work on the proposed method

is ongoing, and we are currently examining techniques for further highlighting problem features within the visualisation. This is in addition to considering a wider range of problem features, and other types of problems (e.g., discrete problems and real-world problems).

It is clear how different information is preserved in the search and objective spaces, and that the both spaces should be used in parallel to maximise the information obtained about the population movements. We are currently exploring interactive visualisations that are based on a linear combination plot of the two, as well as allowing a user to manipulate the combination in an interactive visualisation, allowing them to run through the populations in both the objective/search space and the MDS space simultaneously. Ultimately, the aim of this ongoing work is to help the DM and researchers identify the population movements within their problems and provide a better comprehension of the algorithms and problems. Enabling this kind of transparency within genetic algorithms will make the use of genetic algorithms more accessible to DMs.

## References

1. Brockhoff, D., Auger, A., Hansen, N., Tušar, T.: Quantitative performance assessment of multiobjective optimizers: the average runtime attainment function. In: Trautmann, H., et al. (eds.) EMO 2017. LNCS, vol. 10173, pp. 103–119. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-54157-0\\_8](https://doi.org/10.1007/978-3-319-54157-0_8)
2. Craven, M.J., Jimbo, H.C.: EA stability visualization: perturbations, metrics and performance. In: Proceedings of the Companion Publication of GECCO 2014, pp. 1083–1090 (2014)
3. Črepinšek, M., Liu, S.-H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: a survey. *ACM Comput. Surv. (CSUR)* **45**(3), 1–33 (2013)
4. De Lorenzo, A., Medvet, E., Tušar, T., Bartoli, A.: An analysis of dimensionality reduction techniques for visualizing evolution. In: Proceedings of GECCO 2019, pp. 1864–1872 (2019)
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 (2013)
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 multi-objective optimization test problems. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002) (Cat. No. 02TH8600), vol. 1, pp. 825–830. IEEE (2002)
8. 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)
9. He, Z., Yen, G.G.: Visualization and performance metric in many-objective optimization. *IEEE Trans. Evol. Comput.* **20**(3), 386–402 (2015)
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. Kobayashi, Y., Okamoto, T., Koakutsu, S.: A Pareto optimal solution visualization method using SOM-NG with learning parameter optimization. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 004525–004531 (2016)
12. Torgerson, W.S.: Multidimensional scaling: I. theory and method. *Psychometrika* **17**(4), 401–419 (1952). <https://doi.org/10.1007/BF02288916>
13. Tušar, T., Filipič, B.: Visualization of Pareto front approximations in evolutionary multiobjective optimization: a critical review and the prosection method. *IEEE Trans. Evol. Comput.* **19**(2), 225–245 (2015)
14. Walker, D.J., Craven, M.J.: Toward the online visualisation of algorithm performance for parameter selection. In: Sim, K., Kaufmann, P. (eds.) *EvoApplications 2018*. LNCS, vol. 10784, pp. 547–560. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77538-8\\_38](https://doi.org/10.1007/978-3-319-77538-8_38)
15. Walkerm, D.J., Craven, M.J.: Identifying good algorithm parameters in evolutionary multi-and many-objective optimisation: a visualisation approach. *Appl. Soft Comput.* **88**, 105902 (2020)
16. Walker, D.J., Everson, R.M., Fieldsend, J.E.: Visualizing mutually nondominating solution sets in many-objective optimization. *IEEE Trans. Evol. Comput.* **17**(2), 165–184 (2012)



# Improving Many-Objective Evolutionary Algorithms by Means of Edge-Rotated Cones

Yali Wang<sup>(✉)</sup>, André Deutz, Thomas Bäck, and Michael Emmerich

Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1,  
2333CA Leiden, The Netherlands  
[y.wang@liacs.leidenuniv.nl](mailto:y.wang@liacs.leidenuniv.nl)

**Abstract.** Given a point in  $m$ -dimensional objective space, any  $\varepsilon$ -ball of a point can be partitioned into the incomparable, the dominated and dominating region. The ratio between the size of the incomparable region, and the dominated (and dominating) region decreases proportionally to  $1/2^{m-1}$ , i.e., the volume of the Pareto dominating orthant as compared to all other volumes. Due to this reason, it gets increasingly unlikely that dominating points can be found by random, isotropic mutations. As a remedy to stagnation of search in many objective optimization, in this paper, we suggest to enhance the Pareto dominance order by involving an obtuse convex dominance cone in the convergence phase of an evolutionary optimization algorithm. We propose edge-rotated cones as generalizations of Pareto dominance cones for which the opening angle can be controlled by a single parameter only. The approach is integrated in several state-of-the-art multi-objective evolutionary algorithms (MOEAs) and tested on benchmark problems with four, five, six and eight objectives. Computational experiments demonstrate the ability of these edge-rotated cones to improve the performance of MOEAs on many-objective optimization problems.

**Keywords:** Cone order · Pareto dominance · Many-objective evolutionary algorithm

## 1 Introduction

Multi-objective evolutionary algorithms (MOEAs) have been successfully used in the application area of multi-objective optimization due to their ability to approximate the entire Pareto front in a single run. The Pareto dominance relation, as the most commonly adopted ranking method, plays an essential role in many MOEAs because Pareto dominance is used to compare solutions even when different selection mechanisms are employed in different categories of MOEAs.

---

This work is part of the research programme Smart Industry SI2016 with project name CIMPLO and project number 15465, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

The well-known NSGA-II [1] is a Pareto dominance-based MOEA, using Pareto non-dominated sorting as the first ranking criterion and crowding distance to promote diversity in the population. DI-MOEA [2] is an indicator-based MOEA, it employs the non-dominated sorting as the first ranking criterion and a diversity indicator as the second criterion, which is the Euclidean distance based geometric mean gap indicator. It has been shown to be invariant to the shape of the Pareto front and can achieve evenly spread Pareto front approximations. The NSGA-III [3] is an extension of NSGA-II and it is a decomposition-based MOEA. It employs the Pareto non-dominated sorting to partition the population into a number of fronts, but replaces the crowding distance operator with a clustering operator based on a set of reference points.

Although the Pareto dominance relation usually works well on multi-objective problems with two or three objectives, its ability is often severely degraded when handling many-objective problems (MaOPs) where more than three objectives need to be optimized simultaneously. One major reason of its performance deterioration in many-objective optimization is that individuals are not likely to be dominated by others. Given a point in  $m$ -dimensional objective space, any  $\varepsilon$ -ball of a point can be partitioned into the incomparable, the dominated and dominating region. The ratio between the size of the incomparable region, and the dominated (and dominating) region decreases proportionally to  $1/2^{m-1}$ , i.e., the volume of the Pareto dominating orthant as compared to all other volumes. Due to this reason, it gets increasingly unlikely that dominating points can be found by random, isotropic mutations and classical algorithms do not converge to the Pareto front. The straightforward attempt to overcome the weakness is to use a large population. However, the use of a large population causes other issues. Firstly, the computing time of MOEAs drastically increases because of the increase of the population size. Secondly, the use of a large population size severely degrades the search ability of some MOEAs, (e.g., NSGA-II) [4]. Instead, we propose to extend the Pareto dominance order during the convergence phase by involving the cone order from a convex obtuse dominance cone. The new cone is implemented by rotating the edges of the standard Pareto cone by means of a single parameter. In this way, an individual can dominate larger space, thus, a gradient towards dominating solutions can be followed using relatively small population sizes.

The structure of this paper is as follows. After discussing related work (Sect. 2), Sect. 3 describes the edge-rotated cone dominance approach. Section 4 presents a comparative analysis. Finally, Sect. 5 concludes the paper.

## 2 Related Work

The Pareto dominance relationship is the most commonly adopted ranking method in multi-objective optimization. However, with the increase of the number of objectives, the convergence ability of MOEAs based on Pareto dominance degrades significantly [5]. Recently, some researchers have proposed the use of relaxed forms of Pareto dominance as a way of regulating convergence of MOEAs.

Under these relaxed definitions, a solution has a higher chance to be dominated by other solutions and the selection pressure toward the Pareto front is increased.

**Definition 1** (*Pareto dominance*). An objective vector  $y^{(1)} \in \mathbb{R}^m$  is said to dominate another objective vector  $y^{(2)} \in \mathbb{R}^m$  (denoted by  $y^{(1)} \prec_{\text{pareto}} y^{(2)}$ ) if and only if:  $y_i^{(1)} \leq y_i^{(2)} \forall i = 1, \dots, m$  and  $\exists i \in \{1, \dots, m\} : y_i^{(1)} < y_i^{(2)}$ .

Ikeda et al. proposed  $\alpha$ -dominance [6] to deal with dominance resistant solutions (DRSs), which are solutions far from the Pareto front but are hardly dominated. In  $\alpha$ -dominance, the upper and lower bounds of trade-off rates between two objectives  $f_i$  and  $f_j$ , i.e.,  $\alpha_{ij}$  and  $\alpha_{ji}$ , are pre-defined. Before judging the dominance relations between two individuals  $y$  and  $y'$  in the population, the following definition is considered:  $g_i(y, y') := f_i(y) - f_i(y') + \sum_{j \neq i}^M \alpha_{ij}(f_j(y) - f_j(y'))$ . Solution  $y$  dominates solution  $y'$  if and only if  $\forall i \in \{1, \dots, m\} : g_i(y, y') \leq 0$  and  $\exists i \in \{1, \dots, m\} : g_i(y, y') < 0$ . Using  $\alpha$ -dominance allows a solution to dominate another if it is slightly inferior to the other in one objective, but largely superior in other objectives by setting lower and upper bounds of trade-off rates between objectives.

Laumanns et al. proposed the concept of  $\epsilon$ -dominance [7]. Given two solutions  $y, y' \in \mathbb{R}^m$ , and  $\epsilon > 0$ ,  $y$  is said to  $\epsilon$ -dominate  $y'$  if and only if  $\forall i \in \{1, \dots, m\} : y_i - \epsilon \leq y'_i$ . Cone  $\epsilon$ -dominance [8] has been proposed by Batista et al. to improve  $\epsilon$ -dominance which may eliminate viable solutions. It introduces a parameter  $k$  ( $k \in [0, 1)$ ) to control the shape of the dominance area of a solution using cones. Cone-dominance is also prominently used in multi-criteria decision making (MCDM), in order to formulate user preferences [9].

Sato et al. proposed an approach to control the dominance area of solutions (CDAS) [10]. In CDAS, the objective values are modified and the  $i$ -th objective value of  $x$  after modification is defined as:  $\hat{f}_i(x) = \frac{r \cdot \sin(w_i + S_i \cdot \pi)}{\sin(S_i \cdot \pi)}$ , where  $r$  is the norm of  $f(x)$ ,  $w_i$  is the declination angle between  $f(x)$  and the coordinate axis. The degree of expansion or contraction can be controlled by the parameter  $S_i \in [0.25, 0.25]$ . CDAS controls the aperture of the cone of dominance so that the influence of each point could be increased.

Yang et al. proposed a grid dominance relation [11] in the grid-based evolutionary algorithm (GrEA). The grid dominance adds the selection pressure by adopting an adaptive grid construction. It uses grid-based convergence and diversity measurements to compare non-dominated solutions.

Recently, an angle dominance criterion was proposed in [12]. It designs a parameter  $k$  which works together with the worst point of the current population to control the dominance area of a solution. The angle of a solution (e.g., solution  $y$ ) on one objective (e.g., the  $i$ th objective),  $\alpha_i^y$ , is determined by two lines: the  $i$ th axis; and the line connecting the solution and the farthest point on the  $i$ th axis in the dominance area. Solution  $y$  angle dominates solution  $y'$  if and only if  $\forall i \in \{1, \dots, m\} : \alpha_i^y \leq \alpha_i^{y'}$  and  $\exists i \in \{1, \dots, m\} : \alpha_i^y < \alpha_i^{y'}$ .

Other than these, the  $(1 - k)$ -based criterion [13] has been considered when addressing MaOPs. After comparing a solution to another and counting the number of objectives where it is better than, the same as, or worse than the

other, this criterion uses these numbers to distinguish the relations of domination between solutions. The  $k$ -optimality [14] is a relation based on the number of improved objectives between two solutions. The  $l$ -optimality [15] not only takes into account the number of improved objective values but also considers the values of improved objective functions, if all objectives have the same importance. The concept of volume dominance was proposed by Le and Landa-Silva [16]. This form of dominance is based on the volume of the objective space that a solution dominates.

In this paper, we propose the approach of using the edge-rotated cone to enhance the traditional Pareto dominance. The edge-rotated cone can lead to the same dominance relation as  $\alpha$ -dominance. However, it is interpreted in a more intuitive and geometric way and compared to angle-based method does not require the knowledge of the ideal point or the nadir point.

### 3 Proposed Algorithm

#### 3.1 Proposed Dominance Relation

The Pareto dominance relation or Pareto order ( $\prec_{pareto}$ ) is a special case of cone orders, which are orders defined on vector spaces. The left image of Fig. 1 shows an example of applying the Pareto order cone to illustrate the Pareto dominance relation, i.e.,  $y$  dominates the points in  $y \oplus \mathbb{R}_{>o}^2$  and  $y'$  dominates the points in  $y' \oplus \mathbb{R}_{>o}^2$ . Here,  $\mathbb{R}_{>o}^2$  is the Pareto order cone and  $\oplus$  is the Minkowski sum.

**Definition 2 (Cone).** A set  $C$  is a cone if  $\lambda w \in C$  for any  $w \in C$  and  $\forall \lambda > 0$ .

**Definition 3 (Minkowski Sum).** The Minkowski sum (aka algebraic sum) of two sets  $A \in \mathbb{R}^m$  and  $B \in \mathbb{R}^m$  is defined as  $A \oplus B := \{a + b \mid a \in A \wedge b \in B\}$ .

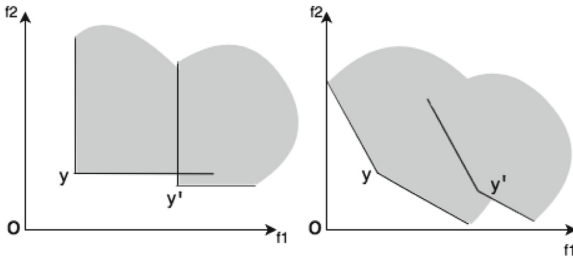


Fig. 1. Pareto and edge-rotated cone dominance.

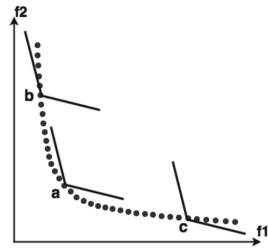


Fig. 2. Trade-off on PF.



In an MOEA, if a solution can dominate more area based on the adopted dominance relation, the algorithm is capable of exploring more solutions and hence accelerating convergence. To this end, we widen the angle of the Pareto order cone and generate the cone which can dominate a larger area. Given a linearly independent vector set  $\{w_1, w_2, \dots, w_m\}$ , a cone can be generated in  $m$ -dimensional space.

**Definition 4** (*Generated  $m$ -dimensional cone*). *The cone generated by the vectors  $w_1, w_2, \dots, w_m$  is the set  $C = \{z : z = \lambda_1 w_1 + \lambda_2 w_2 + \dots + \lambda_m w_m, \forall \lambda_1, \lambda_2, \dots, \lambda_m \geq 0, \lambda \neq 0\}$ ;  $w_1, \dots, w_m$  are linearly independent.*

To be specific, the Pareto order cone is widened by rotating the edges of the standard Pareto order cone around the origin towards the outside. For example, in two-dimensional space, the Pareto order cone is the cone generated by two axes which support an angle of  $90^\circ$ . By rotating two axes towards the opposite direction around the origin, the two axes can reach into the second and fourth quadrants respectively and an edge-rotated cone with an angle larger than  $90^\circ$  is generated. The right image of Fig. 1 shows how the dominance relation has been changed when the edge-rotated cone order is applied. In the left image of Fig. 1,  $y$  and  $y'$  are mutually non-dominated by each other because neither of them is in the dominating space of the other point. However, when an edge-rotated cone is adopted in the right image, the point  $y'$  is dominated by  $y$ . We can see that the edge-rotated cones provide a stricter order compared to the Pareto order. They can guide the search towards the Pareto front better as they establish an ordering among the incomparable solutions (with respect to the Pareto order) in the sense that better incomparable solutions are preferred.

When using the edge-rotated cone order in MOEAs, since the concave cones do not give rise to a strict partial order and the non-dominated points in the order generated by acute-angle cones can be dominated in the Pareto order, we restrict ourselves to convex obtuse cones obtained by rotating each edge of the standard Pareto cone towards the outside with an angle of less than  $45^\circ$ .

**Definition 5** (*Convex Cone*). *A cone  $\mathcal{C}$  is convex if and only if  $\forall c_1 \in \mathcal{C}, c_2 \in \mathcal{C}, \forall \alpha (0 \leq \alpha \leq 1) : \alpha c_1 + (1 - \alpha)c_2 \in \mathcal{C}$ .*

The approach of widening the standard Pareto cone in  $m$ -dimensional space ( $m > 2$ ) is the same. Each edge of the standard Pareto order cone is rotated by an angle less than  $45^\circ$  in the opposite direction of the identity line in the positive orthant. The rotation takes place in the plane determined by the edge and the identity line. In  $m$ -dimensional space, the identity line in the positive orthant is the line passing through the origin and the point  $(1, \dots, 1)$ . The new cone composed of the rotated edges can give rise to a new dominance relation.

### 3.2 Implementation and Integration in MOEAs

In a multi-objective optimization algorithm, solutions that are dominating under the Pareto order are also dominating under the edge-rotated cone order. In this way, it is guaranteed that a minimal element of the edge-rotated cone order is also a minimal element of the Pareto order, and thus algorithms that converge to globally efficient points under the edge-rotated cone order will also converge to globally Pareto efficient points. By using the edge-rotated cone, a solution, especially the solution which is not in the knee region, has a higher chance to be dominated by other solutions. The knee region is the region where the maximum trade-off of objective functions takes place. For the Pareto front in Fig. 2, the knee region is where the Pareto surface bulges the most, i.e., the region near solution  $a$ . When comparing the knee point  $a$  with another solution  $c$ , solution  $c$  has a better (i.e., lower)  $f_2$  value as compared to solution  $a$ . However, this small improvement leads to a large deterioration in the other objective  $f_1$ . Due to the reason that in the absence of explicitly provided preferences, all objectives are considered equally important, solution  $a$ , thus, is more preferable than solution  $c$ . It has been argued in the literature that knee points are the most interesting solutions and preferred solutions [17–20]. Therefore, although not all globally efficient points might be obtained by the edge-rotated cone orders, the edge-rotated cone orders naturally filter out non-preferred solutions. In Fig. 2, when applying the edge-rotated cone, solutions in the knee region can survive, while solutions like  $b$  and  $c$  are on the flat Pareto surface and are more easily to be dominated.

---

**Algorithm 1.** Applying a proper cone order in each iteration.

---

```

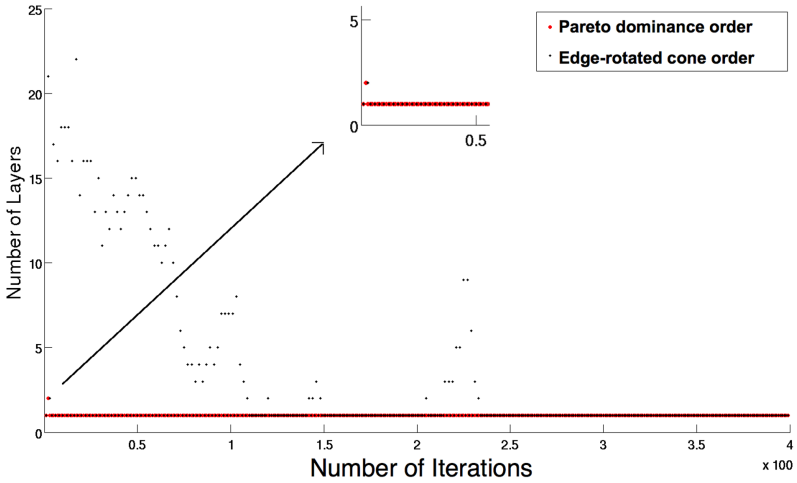
1:  $m \leftarrow$  the number of objectives;
2:  $Degree[m]$ ; // the rotation angle for each edge of the standard Pareto order;
3:  $n\_rank \leftarrow$  Pareto rank number of current population;
4: if  $n\_rank = 0$  then
5:   for each  $i \in \{1, \dots, m\}$  do
6:      $Degree[i] \leftarrow PI/6$ ; // rotation angle is  $30^\circ$ 
7:   end for
8: else
9:   for each  $i \in \{1, \dots, m\}$  do
10:     $Degree[i] \leftarrow 0$ ; // standard Pareto cone
11:   end for
12: end if

```

---

The feature of the edge-rotated cone to eliminate solutions can be appreciated as an advantage especially in the realm of many-objective optimization considering the exponential increase in the number of non-dominated solutions necessary for approximating the entire Pareto front. With the edge-rotated cone, part of the solutions, especially non-preferred solutions, can be excluded. However, this could degrade the diversity of the solution set. Therefore, we propose Algorithm 1 to choose a proper cone order in each iteration of MOEAs in order

to promote diversity in addition to convergence. When running an MOEA, the current population is ranked based on the current cone order at the beginning of each iteration; the edge-rotated cone will be adopted only under the condition that all solutions in the current population are mutually non-dominated by each other. In the case that the current population consists of multiple layers, the standard Pareto cone is used (i.e., the rotation angle is  $0^\circ$ ). The underlying idea is when all the solutions are non-dominated with each other, the edge-rotated cone is adopted to enhance the selection pressure, otherwise, the Pareto order cone is used to maintain the diversity of the population.



**Fig. 3.** The dynamics of the number of layers.

When Algorithm 1 is applied in NSGA-II on the DTLZ1 eight objective problem, Fig. 3 compares the changes of the number of layers between running NSGA-II using only the Pareto dominance and involving the edge-rotated cone order with a rotation angle of  $20^\circ$  within the first 400 iterations (Population size is 100.). When running the original NSGA-II, except that one point lies at level 2 (i.e., the number of fronts is two) at the very beginning, the number of layers always remains one, meaning that all solutions in the current population are non-dominated with each other. As a result, the Pareto dominance relation has no effect on parent selection. That is, an individual with a larger crowding distance is always chosen as a parent in the binary tournament selection since all solutions have the same rank. In this manner, the selection pressure toward the Pareto front is severely weakened. However, when the edge-rotated cone is involved, the layering of the population is very noticeable. In this case, an ordering among the incomparable solutions is established and it can guide the search towards the Pareto front better.

Next we derive a criterion by which one can determine whether a point  $y' \in \mathbb{R}^2$  is dominated by a point  $y \in \mathbb{R}^2$  with respect to the edge-rotated cone

order. Let  $e_1 := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$  and  $e_2 := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$  be the edges of the two-dimensional standard Pareto cone. Then the edges of the edge-rotated cone by a rotation angle  $\alpha$  ( $0 \leq \alpha < \frac{\pi}{4}$ ) are  $Ae_1$  and  $Ae_2$ , where  $A = \begin{bmatrix} \cos(-\alpha) & \frac{\sin(-\alpha)}{\sqrt{2-1}} \\ \frac{\sin(-\alpha)}{\sqrt{2-1}} & \cos(-\alpha) \end{bmatrix}$ .

A point  $y'$  lies in the edge-rotated cone region of  $y$  if and only if for some  $\lambda$ ,  $y' = y + \lambda_1 Ae_1 + \lambda_2 Ae_2$  such that  $\lambda_1, \lambda_2 \geq 0, \lambda \neq 0$ . This is equivalent to: for some  $\lambda$ ,  $A^{-1}(y' - y) = \lambda_1 e_1 + \lambda_2 e_2$  such that  $\lambda_1, \lambda_2 \geq 0, \lambda \neq 0$ . In short,  *$y$  dominates  $y'$  with respect to the edge-rotated cone order if and only if the components of  $A^{-1}(y' - y)$  are non-negative and at least one of them is strictly positive.*

Thus, once the inverse matrix of  $A$  is computed ( $A^{-1} = c \cdot \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$ ,  $c := \frac{1}{(\cos(\alpha))^2 - (\sin(\alpha))^2}$ ), it can readily be determined whether  $y'$  is in the dominating region of  $y$ . Moreover, in case the components are non-zero and have opposite signs, then the points are incomparable. In case the components are non-positive and at least one them negative, then  $y'$  dominates  $y$ .

The approach can easily be applied to three or many objective problems. When the number of objectives is  $m(m > 2)$  and the rotation angle for each edge of the cone is  $\alpha$ , the  $(m \times m)$  matrix (1) gives the coordinates of the unit point on rotated edges: for each unit point on the edge of the standard Pareto cone, each column of the matrix gives its new coordinates after rotation. For example, in three-dimensional space,  $(1, 0, 0)$  is the unit point on one edge of the standard Pareto cone, then  $(\cos(-\alpha), \frac{\sin(-\alpha)}{\sqrt{2}}, \frac{\sin(-\alpha)}{\sqrt{2}})$  are its new coordinates after the edge is rotated by an angle of  $\alpha$  ( $0 \leq \alpha < \frac{\pi}{4}$ ).

$$\begin{bmatrix} \cos(-\alpha) & \frac{\sin(-\alpha)}{\sqrt{m-1}} & \dots & \frac{\sin(-\alpha)}{\sqrt{m-1}} \\ \frac{\sin(-\alpha)}{\sqrt{m-1}} & \cos(-\alpha) & \dots & \frac{\sin(-\alpha)}{\sqrt{m-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\sin(-\alpha)}{\sqrt{m-1}} & \frac{\sin(-\alpha)}{\sqrt{m-1}} & \dots & \cos(-\alpha) \end{bmatrix} \tag{1}$$

When integrating the edge-rotated cone in MOEAs, the inverse matrix needs to be calculated only once. Therefore, almost no extra computing time is involved by Algorithm 1. For a similar cone construction, see [21].

## 4 Experimental Results and Discussion

### 4.1 Experimental Design

The proposed edge-rotated cone order can be integrated in all MOEAs using the Pareto order to select solutions. In this section, Algorithm 1 is combined in NSGA-II, DI-MOEA and NSGA-III to investigate the performance of the proposed approach when different rotation angles (i.e., from 3° to 30°) have been applied. Four, six and eight objective DTLZ1, DTLZ2, DTLZ2\_convex problems have been chosen in the experiments. The optimal Pareto front of DTLZ1 lies

on a linear hyperplane and the optimal Pareto front of DTLZ2 is concave. At the same time, to measure the performance on a convex problem, we transform DTLZ2 problem to DTLZ2\_convex problem with a convex Pareto front by simply decreasing all objective values by 3.5. The other two benchmark problems include UF11 and UF13 [22]. UF11 is a rotated instance of the 5D DTLZ2 test problem, and UF13 is the 5D WFG1 test problem.

The population size is 100 for all problems. We have taken 15 independent runs (with a different seed for each run but the same seed for each of the algorithms) of each algorithm on each problem. For each problem, the number of evaluations (NE) is the computing budget for running the algorithm and it is determined by  $\max\{100000, 10000 \times D\}$ , where  $D$  is the number of decision variables. Two widely-used quality metrics, hypervolume (HV) [23] and inverted generational distance (IGD) [24], have been adopted to compare the performance of the algorithms. All experiments are implemented based on the MOEA Framework 2.12 (<http://www.moeaframework.org/>), which is a Java-based framework for multi-objective optimization. When calculating HV, the objective values of the reference point are 0.6 on DTLZ1, 1.1 on DTLZ2, 5 on DTLZ2\_convex, 2.2 on UF11 and 11 on UF13. The origin is used as the ideal point. When calculating the IGD value, the merged non-dominated solution sets from all runs are used as the reference sets of the DTLZ2\_convex problems and the reference sets of other problems are from the MOEA framework.

## 4.2 Experimental Results

Tables 1 and 2 show the mean hypervolume and IGD from 15 runs of DTLZ2 and UF problems when different edge-rotated cone orders are integrated in NSGA-II, DI-MOEA and NSGA-III. Due to the page limit, tables for DTLZ1 and DTLZ2\_convex problems are in [25]. The “P\_cone” column provides the results obtained by the original MOEAs. The “ $\frac{\pi}{6}$  (= 30°)” column gives the results when each edge of the standard Pareto order cone has been rotated by 30° in the algorithm, similar remark for the other columns. The mean hypervolume and IGD values obtained by the original NSGA-II, DI-MOEA and NSGA-III have been used as the reference values to be compared with the results achieved by the algorithms involving the edge-rotated cone orders. For the algorithms combining the edge-rotated cone, the mean hypervolume and IGD values better than the values obtained by the original MOEAs have been highlighted in bold (i.e., a larger hypervolume value and lower IGD value); the largest respectively lowest value for each algorithm among them is printed in red and underlined. At the same time, the standard deviation of each algorithm is also given under each mean hypervolume and IGD. Tables for the DTLZ benchmark problems consist of four parts, namely four objective, six objective, eight objective with full budget, and eight objective with half budget. The behaviours of UF11 and UF13 with full budget and half budget are given in Table 2. Furthermore, the ranking of these algorithms has been calculated based on the mean hypervolume and shown in [25].

We can draw the following conclusions from the data in these tables.

**Table 1.** The mean Hypervolume (HV) and IGD on DTLZ2 (concave).

Four objective (NE = 130000)								
Algorithms	Metrics	P_cone	$\frac{\pi}{6} (=30^\circ)$	$\frac{\pi}{9} (=20^\circ)$	$\frac{\pi}{12} (=15^\circ)$	$\frac{\pi}{18} (=10^\circ)$	$\frac{\pi}{30} (=6^\circ)$	$\frac{\pi}{60} (=3^\circ)$
NSGA-II	Mean HV	0.5953	0.1971	0.5458	<b>0.6760</b>	<b>0.6525</b>	<b>0.6388</b>	<b>0.6333</b>
	std	0.0089	0.1182	0.0535	0.0041	0.0048	0.0080	0.0077
DI-MOEA	Mean HV	0.6471	0.0913	0.5639	<b>0.6944</b>	<b>0.6897</b>	<b>0.6755</b>	<b>0.6688</b>
	std	0.0094	0.0012	0.0406	0.0038	0.0026	0.0066	0.0039
NSGA-III	Mean HV	0.6597	0.2508	0.5749	<b>0.6863</b>	<b>0.6821</b>	<b>0.6652</b>	0.6592
	std	0.0054	0.1265	0.0362	0.0017	0.0040	0.0031	0.0066
NSGA-II	Mean IGD	0.1634	0.8352	0.4037	0.1867	<b>0.1492</b>	<b>0.1536</b>	<b>0.1542</b>
	std	0.0045	0.2290	0.0794	0.0056	0.0040	0.0055	0.0041
DI-MOEA	Mean IGD	0.1363	1.0405	0.3810	0.1731	<b>0.1264</b>	<b>0.1295</b>	<b>0.1279</b>
	std	0.0045	0.0183	0.0661	0.0049	0.0022	0.0061	0.0028
NSGA-III	Mean IGD	0.1501	0.7553	0.3510	0.1749	<b>0.1361</b>	<b>0.1477</b>	<b>0.1490</b>
	std	0.0046	0.2196	0.0705	0.0039	0.0034	0.0054	0.0026
Six objective (NE = 150000)								
NSGA-II	Mean HV	0.1224	0.0000	<b>0.4304</b>	<b>0.8156</b>	<b>0.7608</b>	<b>0.7284</b>	<b>0.6490</b>
	std	0.0701	0.0000	0.0254	0.0036	0.0067	0.0119	0.0221
DI-MOEA	Mean HV	0.0000	0.0000	<b>0.4488</b>	<b>0.8397</b>	<b>0.8016</b>	<b>0.7479</b>	<b>0.6543</b>
	std	0.0000	0.0000	0.0126	0.0055	0.0055	0.0117	0.0347
NSGA-III	Mean HV	0.8052	0.0000	0.4411	<b>0.8446</b>	<b>0.8185</b>	<b>0.8127</b>	<b>0.8111</b>
	std	0.0076	0.0000	0.0130	0.0048	0.0038	0.0056	0.0041
NSGA-II	Mean IGD	0.7278	2.5612	0.7003	<b>0.3447</b>	<b>0.2856</b>	<b>0.2887</b>	<b>0.3137</b>
	std	0.0758	0.0090	0.0380	0.0119	0.0051	0.0046	0.0091
DI-MOEA	Mean IGD	1.9390	2.5824	<b>0.6961</b>	<b>0.2913</b>	<b>0.2774</b>	<b>0.2898</b>	<b>0.3335</b>
	std	0.3246	0.0059	0.0285	0.0074	0.0026	0.0058	0.0172
NSGA-III	Mean IGD	0.3125	2.5596	0.7260	<b>0.3073</b>	<b>0.3061</b>	<b>0.3092</b>	<b>0.3095</b>
	std	0.0105	0.0154	0.0283	0.0145	0.0071	0.0065	0.0080
Eight objective (NE = 170000)								
NSGA-II	Mean HV	0.0168	0.0000	<b>0.4947</b>	<b>0.8850</b>	<b>0.8193</b>	<b>0.7068</b>	<b>0.4062</b>
	std	0.0355	0.0000	0.0576	0.0068	0.0068	0.0487	0.0754
DI-MOEA	Mean HV	0.0000	0.0000	<b>0.4250</b>	<b>0.9002</b>	<b>0.8011</b>	<b>0.4619</b>	<b>0.0138</b>
	std	0.0000	0.0000	0.1260	0.0033	0.0196	0.1500	0.0516
NSGA-III	Mean HV	0.8543	0.0000	0.3151	<b>0.9079</b>	<b>0.8727</b>	<b>0.8632</b>	0.8522
	std	0.0121	0.0000	0.0643	0.0044	0.0074	0.0078	0.0138
NSGA-II	Mean IGD	1.2941	2.4798	<b>0.7887</b>	<b>0.5247</b>	<b>0.3955</b>	<b>0.4332</b>	<b>0.6433</b>
	std	0.1867	0.0422	0.0507	0.0210	0.0068	0.0201	0.0687
DI-MOEA	Mean IGD	2.4722	2.5704	<b>0.8728</b>	<b>0.4483</b>	<b>0.4425</b>	<b>0.6013</b>	<b>2.3017</b>
	std	0.0430	0.0129	0.1118	0.0054	0.0088	0.0682	0.4257
NSGA-III	Mean IGD	0.4594	1.9278	0.9662	0.4936	0.4659	0.4638	0.4680
	std	0.0105	0.1043	0.0491	0.0130	0.0099	0.0093	0.0175
Eight objective - Half budget (NE = 85000)								
NSGA-II	Mean HV	0.0001	0.0000	<b>0.4674</b>	<b>0.8859</b>	<b>0.8161</b>	<b>0.7145</b>	<b>0.4251</b>
	std	0.0003	0.0000	0.0847	0.0047	0.0083	0.0334	0.0851
DI-MOEA	Mean HV	0.0000	0.0000	<b>0.4196</b>	<b>0.9000</b>	<b>0.8061</b>	<b>0.5432</b>	<b>0.0213</b>
	std	0.0000	0.0000	0.1254	0.0050	0.0207	0.0931	0.0606
NSGA-III	Mean HV	0.8526	0.0000	0.3223	<b>0.9063</b>	<b>0.8728</b>	<b>0.8616</b>	<b>0.8548</b>
	std	0.0084	0.0000	0.0553	0.0048	0.0054	0.0085	0.0116
NSGA-II	Mean IGD	1.6856	2.4963	<b>0.8125</b>	<b>0.5167</b>	<b>0.3939</b>	<b>0.4295</b>	<b>0.6116</b>
	std	0.1949	0.0202	0.0763	0.0091	0.0060	0.0126	0.0869
DI-MOEA	Mean IGD	2.4858	2.5688	<b>0.8765</b>	<b>0.4520</b>	<b>0.4391</b>	<b>0.5633</b>	<b>2.0740</b>
	std	0.0272	0.0276	0.1149	0.0073	0.0072	0.0403	0.5132
NSGA-III	Mean IGD	0.4611	1.9307	0.9590	0.4923	0.4691	0.4630	<b>0.4597</b>
	std	0.0178	0.1646	0.0433	0.0127	0.0115	0.0101	0.0152

**Table 2.** The mean Hypervolume (HV) and IGD on UF11 & UF13.

UF11 Five objective (NE = 300000)								
Algorithms	Metrics	P_cone	$\frac{\pi}{6} (=30^\circ)$	$\frac{\pi}{9} (=20^\circ)$	$\frac{\pi}{12} (=15^\circ)$	$\frac{\pi}{18} (=10^\circ)$	$\frac{\pi}{30} (=6^\circ)$	$\frac{\pi}{60} (=3^\circ)$
NSGA-II	Mean HV	0.0000	0.0000	<b>0.0211</b>	<b>0.0291</b>	<b>0.0306</b>	<b>0.0218</b>	<b>0.0104</b>
	std	0.0000	0.0000	0.0024	0.0058	0.0012	0.0011	0.0014
DI-MOEA	Mean HV	0.0029	0.0000	<b>0.0191</b>	<b>0.0336</b>	<b>0.0256</b>	<b>0.0188</b>	<b>0.0138</b>
	std	0.0018	0.0000	0.0035	0.0008	0.0012	0.0015	0.0024
NSGA-III	Mean HV	0.0147	0.0000	<b>0.0266</b>	<b>0.0350</b>	<b>0.0278</b>	<b>0.0201</b>	<b>0.0171</b>
	std	0.0016	0.0000	0.0034	0.0017	0.0016	0.0014	0.0015
NSGA-II	Mean IGD	1.5208	14.6626	<b>0.3890</b>	<b>0.2990</b>	<b>0.2685</b>	<b>0.3119</b>	<b>0.4531</b>
	std	0.2173	0.2878	0.0368	0.0374	0.0171	0.0241	0.0289
DI-MOEA	Mean IGD	0.7304	15.1690	<b>0.6152</b>	<b>0.2807</b>	<b>0.3339</b>	<b>0.3946</b>	<b>0.4621</b>
	std	0.0944	0.2054	0.1997	0.0210	0.0228	0.0352	0.0545
NSGA-III	Mean IGD	0.4517	15.0785	<b>0.4190</b>	<b>0.2795</b>	<b>0.3188</b>	<b>0.3848</b>	<b>0.4166</b>
	std	0.0388	0.2105	0.0697	0.0247	0.0235	0.0324	0.0183
UF11 Five objective - Half budget (NE = 150000)								
NSGA-II	Mean HV	0.0000	0.0000	<b>0.0205</b>	<b>0.0269</b>	<b>0.0288</b>	<b>0.0201</b>	<b>0.0082</b>
	std	0.0000	0.0000	0.0025	0.0055	0.0014	0.0016	0.0017
DI-MOEA	Mean HV	0.0012	0.0000	<b>0.0237</b>	<b>0.0316</b>	<b>0.0244</b>	<b>0.0185</b>	<b>0.0126</b>
	std	0.0011	0.0000	0.0030	0.0020	0.0010	0.0014	0.0017
NSGA-III	Mean HV	0.0148	0.0000	<b>0.0268</b>	<b>0.0342</b>	<b>0.0270</b>	<b>0.0199</b>	<b>0.0170</b>
	std	0.0020	0.0000	0.0029	0.0013	0.0018	0.0016	0.0010
NSGA-II	Mean IGD	1.7202	14.7243	<b>0.3951</b>	<b>0.3031</b>	<b>0.2731</b>	<b>0.3208</b>	<b>0.4846</b>
	std	0.2541	0.1769	0.0392	0.0343	0.0164	0.0289	0.0312
DI-MOEA	Mean IGD	0.8730	15.1172	<b>0.4910</b>	<b>0.2939</b>	<b>0.3418</b>	<b>0.4061</b>	<b>0.4831</b>
	std	0.1485	0.2099	0.0619	0.0269	0.0244	0.0329	0.0439
NSGA-III	Mean IGD	0.4606	15.0148	<b>0.3897</b>	<b>0.2752</b>	<b>0.3204</b>	<b>0.4009</b>	<b>0.4314</b>
	std	0.0433	0.1881	0.0615	0.0186	0.0265	0.0393	0.0335
UF13 Five objective (NE = 300000)								
NSGA-II	Mean HV	0.6937	0.5041	<b>0.7410</b>	<b>0.7424</b>	<b>0.7177</b>	<b>0.7065</b>	<b>0.6994</b>
	std	0.0079	0.1742	0.0096	0.0070	0.0091	0.0084	0.0084
DI-MOEA	Mean HV	0.6611	0.4625	<b>0.7343</b>	<b>0.7152</b>	0.6590	0.6567	0.6589
	std	0.0063	0.1580	0.0064	0.0119	0.0073	0.0067	0.0071
NSGA-III	Mean HV	0.6498	0.4523	<b>0.7164</b>	<b>0.7226</b>	<b>0.7023</b>	<b>0.6703</b>	<b>0.6532</b>
	std	0.0130	0.1017	0.0048	0.0108	0.0085	0.0106	0.0077
NSGA-II	Mean IGD	1.4761	<b>1.3108</b>	<b>1.4316</b>	<b>1.3805</b>	<b>1.4656</b>	<b>1.4391</b>	<b>1.4181</b>
	std	0.1315	0.2267	0.0565	0.0857	0.0664	0.1572	0.1029
DI-MOEA	Mean IGD	1.5448	<b>1.5031</b>	<b>1.5151</b>	1.5481	1.7512	1.6351	1.5934
	std	0.0473	0.4180	0.0533	0.0646	0.0384	0.0667	0.0399
NSGA-III	Mean IGD	1.8698	<b>1.6030</b>	<b>1.6324</b>	<b>1.5813</b>	<b>1.6675</b>	<b>1.7950</b>	<b>1.8527</b>
	std	0.1842	0.1835	0.0285	0.0658	0.0969	0.1457	0.1245
UF13 Five objective - Half budget (NE = 150000)								
NSGA-II	Mean HV	0.6687	0.5016	<b>0.7259</b>	<b>0.7170</b>	<b>0.6915</b>	<b>0.6831</b>	<b>0.6738</b>
	std	0.0041	0.1749	0.0092	0.0058	0.0042	0.0047	0.0057
DI-MOEA	Mean HV	0.6457	0.3427	<b>0.7254</b>	<b>0.7002</b>	<b>0.6513</b>	<b>0.6481</b>	<b>0.6497</b>
	std	0.0045	0.2041	0.0044	0.0133	0.0056	0.0053	0.0057
NSGA-III	Mean HV	0.6432	0.4702	<b>0.7073</b>	<b>0.7045</b>	<b>0.6770</b>	<b>0.6579</b>	0.6417
	std	0.0086	0.0996	0.0074	0.0076	0.0103	0.0071	0.0056
NSGA-II	Mean IGD	1.5720	1.3736	<b>1.5455</b>	<b>1.5074</b>	1.5968	1.5746	<b>1.5262</b>
	std	0.0946	0.1703	0.0638	0.0649	0.0786	0.1135	0.0860
DI-MOEA	Mean IGD	1.6609	<b>1.5321</b>	<b>1.5939</b>	<b>1.6311</b>	1.8048	1.7286	1.6403
	std	0.0557	0.3781	0.0268	0.0781	0.0509	0.0794	0.0613
NSGA-III	Mean IGD	1.8931	1.7553	<b>1.6824</b>	<b>1.6832</b>	<b>1.8163</b>	1.8976	1.9725
	std	0.1238	0.2361	0.0456	0.0376	0.0924	0.1200	0.0562

1. The algorithms do not work well when a large rotation angle is adopted (e.g.,  $30^\circ$ ); only the mean rank of the algorithm involving the cone with a  $30^\circ$  rotation angle is worse than the mean rank of the original MOEA.
2. The algorithms show similar performance to the original MOEAs when the rotation angle is very small (e.g.,  $3^\circ$ ).
3. When an intermediate rotation angle is adopted, the performance of the algorithms (both hypervolume and IGD values) shows a significant improvement except for a few cases which display values close to the original MOEAs.
4. Although it differs depending on the specific problems, the best performance is usually obtained when the rotation angle is  $15^\circ$ . Also, the mean rank of the algorithm involving the cone with a  $15^\circ$  rotation angle is the best and a  $10^\circ$  rotation angle is the second best.
5. It can be seen that the edge-rotated cone can improve the performance of all three adopted MOEAs (i.e., NSGA-II, DI-MOEA and NSGA-III) in most cases when an intermediate rotation angle is used. Even though NSGA-III is assumed to be powerful enough to handle these benchmark problems, its performance can still be improved by the edge-rotated cone approach.
6. The edge-rotated cone can benefit MOEAs even more with the increase of the number of objectives. For example, when a  $15^\circ$  rotation angle is applied on the DTLZ2 (concave) four objective problem, the hypervolume of NSGA-II is improved from 0.5953 to 0.6760; for the six objective problem, the hypervolume is improved from 0.1224 to 0.8156; and for the eight objective problem, the hypervolume is improved from 0.0168 to 0.8850.
7. The edge-rotated cone can benefit the algorithm with a small computing budget more than the algorithm with a large budget. For example, when using half of the computing budget on UF13 five objective problem and the rotation angle is set to  $20^\circ$ , the hypervolume values of the Pareto fronts from NSGA-II, DI-MOEA and NSGA-III can be improved to 0.7259, 0.7254, 0.7073, which are already larger than the hypervolume values obtained by the original MOEAs with full budget, namely 0.6937, 0.6611 and 0.6497.
8. Even though we did not show the median values of the hypervolume and IGD values in the tables, they show similar values as the mean values. At the same time, the standard deviations show a stable behavior of the edge-rotated cone order when it is integrated in MOEAs.

## 5 Conclusions and Further Work

In this paper, we enhance the standard Pareto dominance relationship from the geometric perspective. By rotating the edges of the standard Pareto order cone, the incomparable solutions can be ranked into different layers, hence, the selection pressure toward the Pareto front can be strengthened and the convergence of the algorithm can be accelerated. To avoid neglecting the diversity, the edge-rotated cone order is designed to work together with the standard Pareto order in our algorithm. After testing various angles on different many-objective optimization problems, we show the ability of improving the performance of original



MOEAs by the edge-rotated cone and suggest that the rotation angle of  $15^\circ$  can be adopted in the absence of specific experiments or knowledge of the application domain. Our method of implementing the integration of the edge-rotated cone barely needs more computing time compared to the original MOEAs, moreover, with a small computing budget, it can promote the performance of the algorithm to the effect of using a large budget without using the edge-rotated cone orders.

Our implementation of enhancing the Pareto dominance is straightforward and effective, we think it is a good direction to improve any MOEA using the Pareto dominance to select solutions. In future, the mechanism that relates the properties of the problem with the rotation angle should be researched. Another interesting direction of future work could be to investigate and compare different schemes of alternating between the cone orders in order to promote diversity and convergence. For instance, it could be investigated whether using acute cones can be of benefit to promote diversity even more, or to use again the Pareto cone in the final stage of the evolution to make sure that no solutions are excluded from the Pareto front which might happen when using the edge-rotated cone.

## References

1. 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)
2. Wang, Y., Emmerich, M., Deutz, A., Bäck, T.: Diversity-indicator based multi-objective evolutionary algorithm: DI-MOEA. In: Deb, K., et al. (eds.) EMO 2019. LNCS, vol. 11411, pp. 346–358. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-12598-1\\_28](https://doi.org/10.1007/978-3-030-12598-1_28)
3. 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)
4. Ishibuchi, H., Sakane, Y., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization by NSGA-II and MOEA/D with large populations. In: 2009 IEEE International Conference on Systems, Man and Cybernetics, pp. 1758–1763. IEEE, October 2009
5. Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L., Deb, K. (eds.) EMO 2003. LNCS, vol. 2632, pp. 376–390. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36970-8\\_27](https://doi.org/10.1007/3-540-36970-8_27)
6. Ikeda, K., Kita, H., Kobayashi, S.: Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal? In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), vol. 2, pp. 957–962. IEEE, May 2001
7. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. *Evol. Comput.* **10**(3), 263–282 (2002)
8. Batista, L.S., Campelo, F., Guimarães, F.G., Ramírez, J.A.: Pareto cone *epsilon*-dominance: improving convergence and diversity in multiobjective evolutionary algorithms. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 76–90. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-19893-9\\_6](https://doi.org/10.1007/978-3-642-19893-9_6)

9. Wiecek, M.M.: Advances in cone-based preference modeling for decision making with multiple criteria. *Decis. Mak. Manuf. Serv.* **1**(1/2), 153–173 (2007)
10. Sato, H., Aguirre, H., Tanaka, K.: Controlling dominance area of solutions in multi-objective evolutionary algorithms and performance analysis on multiobjective 0/1 knapsack problems. *IPSN Digital Courier* **3**, 703–718 (2007)
11. 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)
12. Liu, Y., Zhu, N., Li, K., Li, M., Zheng, J., Li, K.: An angle dominance criterion for evolutionary many-objective optimization. *Inf. Sci.* **509**, 376–399 (2020)
13. Farina, M., Amato, P.: On the optimal solution definition for many-criteria optimization problems. In: 2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings (NAFIPS-FLINT 2002) (Cat. No. 02TH8622), pp. 233–238. IEEE, June 2002
14. Farina, M., Amato, P.: A fuzzy definition of “optimality” for many-criteria optimization problems. *IEEE Trans. Syst. Man Cybern.-Part A: Syst. Hum.* **34**(3), 315–326 (2004)
15. Zou, X., Chen, Y., Liu, M., Kang, L.: A new evolutionary algorithm for solving many-objective optimization problems. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **38**(5), 1402–1412 (2008)
16. Le, K., Landa-Silva, D.: Obtaining better non-dominated sets using volume dominance. In: 2007 IEEE Congress on Evolutionary Computation, pp. 3119–3126. IEEE, September 2007
17. Das, I.: On characterizing the “knee” of the Pareto curve based on normal-boundary intersection. *Struct. Optim.* **18**(2–3), 107–115 (1999). <https://doi.org/10.1007/BF01195985>
18. Branke, J., Deb, K., Dierolf, H., Osswald, M.: Finding knees in multi-objective optimization. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 722–731. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30217-9\\_73](https://doi.org/10.1007/978-3-540-30217-9_73)
19. Deb, K., Gupta, S.: Towards a link between knee solutions and preferred solution methodologies. In: Panigrahi, B.K., Das, S., Suganthan, P.N., Dash, S.S. (eds.) SEMCCO 2010. LNCS, vol. 6466, pp. 182–189. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17563-3\\_22](https://doi.org/10.1007/978-3-642-17563-3_22)
20. Braun, M.A., Shukla, P.K., Schmeck, H.: Preference ranking schemes in multi-objective evolutionary algorithms. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 226–240. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-19893-9\\_16](https://doi.org/10.1007/978-3-642-19893-9_16)
21. Emmerich, M., Deutz, A., Krusselbrink, J., Shukla, P.K.: Cone-based hypervolume indicators: construction, properties, and efficient computation. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) EMO 2013. LNCS, vol. 7811, pp. 111–127. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-37140-0\\_12](https://doi.org/10.1007/978-3-642-37140-0_12)
22. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the CEC 2009 special session and competition (2008)
23. While, L., Hingston, P., Barone, L., Huband, S.: A faster algorithm for calculating hypervolume. *IEEE Trans. Evol. Comput.* **10**(1), 29–38 (2006)
24. 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)
25. Wang, Y., Deutz, A., Bäck, T., Emmerich, M.: Improving many-objective evolutionary algorithms by means of edge-rotated cones. *arXiv preprint arXiv:2004.06941* (2020)

# **Real-World Applications**



# Human-Like Summaries from Heterogeneous and Time-Windowed Software Development Artefacts

Mahfouth Alghamdi<sup>(✉)</sup>, Christoph Treude<sup>(✉)</sup>, and Markus Wagner<sup>(✉)</sup>

School of Computer Science, University of Adelaide, Adelaide, Australia  
{mahfouth.a.alghamdi, christoph.treude, markus.wagner}@adelaide.edu.au

**Abstract.** Automatic text summarisation has drawn considerable interest in the area of software engineering. It is challenging to summarise the activities related to a software project, (1) because of the volume and heterogeneity of involved software artefacts, and (2) because it is unclear what information a developer seeks in such a multi-document summary. We present the first framework for summarising multi-document software artefacts containing heterogeneous data within a given time frame. To produce human-like summaries, we employ a range of iterative heuristics to minimise the cosine-similarity between texts and high-dimensional feature vectors. A first study shows that users find the automatically generated summaries the most useful when they are generated using word similarity and based on the eight most relevant software artefacts.

**Keywords:** Extractive summarisation · Heuristic optimisation · Software development

## 1 Introduction and Motivation

Modern-day rapid software development produces large amounts of data, e.g., GitHub [4] now hosts more than 100 million repositories, with over 87 million pull requests merged in the last year, making it the largest source code hosting service in the world. The corresponding software development involves a lot of communication: developers create many types of software artefacts – such as pull requests, commits, and issues – and the amount can be overwhelming. For example, the Node<sup>1</sup> project contains more than 11k issues, more than 20k pull requests, and over 29k commits. It also contains other software artefacts, such as wiki entries and readme files created by the developers during the project development life-cycle. In addition, these artefacts are frequently updated. For instance, in the week from January 1 to January 7, 2020, developers created 17 new issues, closed 12 issues and submitted 82 commits. Let us now consider two scenarios: (1) a developer has been on holidays during this period and would like to be updated, and (2) a new developer joins the team after this period and

<sup>1</sup> <https://github.com/nodejs/node>, accessed on February 2, 2020.

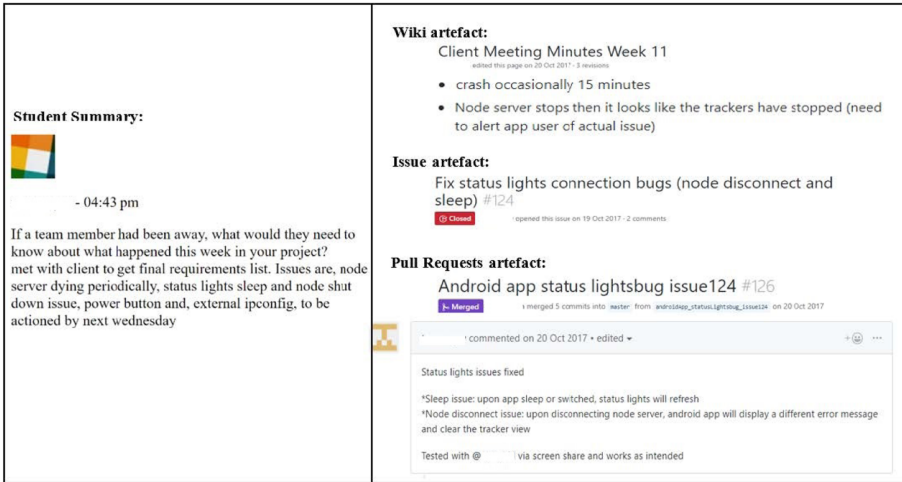


Fig. 1. An example of an anonymised student summary (left) linked to the content of related software artefacts (right).

would like to know what has happened recently. In both cases, going through the artefacts and collecting the most useful information from them can be tedious and time-consuming. It is scenarios like these that we are targeting in our study, as solutions to these can ultimately increase the productivity of software developers and reduce information overload [16]. To make these scenarios more tangible, Fig. 1 shows a summary written by a student software developer, as well as the various artefacts that contain parts of the information conveyed in this manually written summary.

To offer solutions in such cases, we employ a combination of methods from Data-Driven Search-Based Software Engineering (DSE) [8]. DSE combines insights from Mining Software Repositories (MSR) and Search-based Software Engineering (SBSE). While MSR formulates software engineering problems as data mining problems, SBSE reformulates SE problems as optimisation problems and use meta-heuristic algorithms to solve them. Both MSR and SBSE share the common goal of providing insights to improve software engineering. In this present paper, we suggest to improve software engineering – in particular the creation of software development activities – by mining the created artefacts for summaries.

In recent years, several approaches have been developed to summarise software artefacts. Rastkar et al. [12] summarised bug reports using a supervised machine learning method. Rigby et al. [13] summarised code elements from Stack Overflow using four classifiers based on context, location, text type, tf-idf, and element type. Furthermore, Nazar et al. [9] and Ying et al. [18] utilised naive Bayes and support vector machine classifiers to generate summaries of code fragments taken from the official Eclipse FAQ. Interestingly, these techniques have

mostly focused on summarising a single type of artefact, and they have not taken into consideration the production of summaries of content in a given time frame. To address these issues, we present a first framework to create multi-document summaries from heterogeneous software artefacts within a given time frame. In particular, we aim at an extractive approach, which generates a new summary from the relevant documents without creating new sentences [10, 17].

The remainder of this paper is structured as follows. First, we describe the creation of the necessary gold-standard based on 503 human-written summaries in Sect. 2. In Sect. 3, we define the problem of summary-generation as an optimisation problem based on cosine-similarity and on 26 text-based metrics. In Sects. 4 and 5, we report on the results of our computational study and expert annotation of the results. We then discuss threats to validity in Sect. 6 and outline future work in Sect. 7.

## 2 Human-Written Summaries – Creation of a Gold Standard

To better understand what human-written summaries of time-windowed software development artefacts look like, it has been necessary to create our own *gold standard*. The basis of our gold standard is formed by a total of 503 summaries that were produced (mostly) on a weekly basis by 50 students over 14 weeks and for 14 (university-internal) GitHub projects. The students were working in teams of three or four on their capstone projects with clients from local industry toward a Bachelor degree (43 students in total) or with clients from academia toward their Masters degree (7 students in total). To ensure the usefulness of the students' summaries, each of the students' summaries was assessed as part of the student assessments during the particular semester. The summaries were anonymised before conducting this work to ensure confidentiality and anonymity of the students.

We make use of these summaries to understand the general properties of human-written summaries, such as the summaries' typical length and the amount of information these summaries may contain. Additionally, the students' summaries can provide us with an understanding of the common types of artefacts related to development activities mentioned in the students' summaries and to help us identify which sentences from which artefacts should be selected for our extractive summarisation approach.

To automatically collect the summaries, we used a Slack bot that asked the students to write summaries on a weekly basis about their projects development activity (see again Fig. 1). The written summaries were automatically recorded and collected in response to the question: *If a team member had been away, what would they need to know about what happened this week in your project?*

We show an example in Fig. 1: the question and the student's summary are on the left, and the relevant artefacts on the right. Note that the students only provide the summary, i.e., they do not provide a list of the relevant artefacts.

**Table 1.** Number of artefacts contained in the 14 GitHub projects, resulting in 56,152 sentences.

Type	Number	Sentences
Issue titles (IT)	1,885	1,885
Issue bodies (IB)	1,885	5,650
Issue body comments (IBC)	3,280	8,754
Pull requests titles (PRT)	1,103	1,103
Pull requests bodies (PRB)	1,103	5,176
Pull requests body comments (PRBC)	897	1,811
Pull requests reviews (PRRv)	2,019	2,762
Pull requests reviews' comments (PRRvC)	1,286	1,737
Commit messages (CM)	4,562	7,856
Commit comments (CMC)	30	55
Milestone titles (MT)	103	103
Milestone description (MD)	103	142
Readme files (RMe)	14	2,678
Wiki files (Wiki)	492	16,436
Releases (Rel)	1	4

We considered the following 15 types of textual artefacts in the GitHub repositories: issues (titles, bodies, and comments), pull requests (titles, bodies, comments, reviews, and reviews' comments), commits (messages and comments), milestones (titles and descriptions), releases, wiki entries, and readme files. It is worth noting that, after manually inspecting the students' summaries, there was no evidence that any of these summaries contained a reference to a particular source code file.

The documents to be summarised as well as the summaries needed to first undergo various pre-processing steps – including sentence splitting, stop-word removal, tokenisation, and stemming – to reduce noise in the data. Also, we remove source code blocks from the software artefacts due to lack of evidence of student summaries citing code from actual files. Table 1 lists the total amount of artefacts per type in our gold standard, as well as the total number of extracted sentences for each type.

### 3 Methodology

In our approach, we intend to extract text from a set of heterogeneous software artefacts so that the resulting summaries are similar in style to those found in gold-standard summaries. In the following, we introduce two ways of measuring similarity, we revisit the definition of cosine similarity, and we define the iterative search heuristics used later on.

### 3.1 Generating Summaries Based on Word-Similarity and Feature Vector Similarity

Historically, the selection of important sentences for inclusion in a summary is based on various features represented in the sentences such as sentence position [2], sentence length and title similarity [5], sentence centrality [3], and word frequency [6]. Determining these features in the selection of important sentences is not simple and depends largely on the type of documents to be summarised [15].

We consider two ways of characterising sentences: (1) based on the similarity of words, and (2) based on the similarity of feature vectors. In both cases, the goal is to select sentences from the collection of software artefacts so that the characteristics of the resulting summary are close to the characteristics of a target.

First, word similarity between two texts is defined by the number of times a term occurs in both texts, after the aforementioned stemming and removal of stop words. To achieve this, we use a vector-based representation, where each element denotes the number of times a particular word has occurred in the sentence.

**Table 2.** Features used to represent each of the sentences.

No.	Feature	No.	Feature
F1.	Word count	F14.	No. of long words ( $\geq 7$ chars)
F2.	Chars count including spaces	F15.	Longest sentence (chars)
F3.	Chars without spaces	F16.	Longest words (chars)
F4.	No. of syllables in a word	F17.	Longest words by number of syllables
F5.	Sentence length	F18.	Estimated reading time
F6.	Unique words	F19.	Estimated speaking time
F7.	Avg. word length (chars)	F20.	Dale-Chall readability index
F8.	Avg. sentence Length (words)	F21.	Automated readability index
F9.	No. of monosyllabic words	F22.	Coleman-Liau index
F10.	No. of polysyllabic words	F23.	Flesch reading ease score
F11.	Syllables per word	F24.	Flesch-Kincaid grade level
F12.	Difficult words	F25.	Gunning fog index
F13.	No. of short words ( $\leq 3$ chars)	F26.	Shannon entropy

Second, as an alternative to the word-similarity and for situations where a reference text is unavailable, we consider a total of 26 text-based features of sentences, which aim at capturing different aspects of readability metrics, information-theoretic entropy and other lexical features (see Table 2). Each sentence is represented as a 26-dimensional vector of the feature values. For an initial characterisation of this high-dimensional dataset, we refer the interested reader to [1].



### 3.2 Cosine Similarity

The most popular similarity measure used in the field of text summarisation is cosine similarity [7] as it has advantageous properties for high dimensional data [14].

To measure the cosine similarity between two sentences  $x$  and  $y$  – respectively their representation as a vector of word counts or the 26-dimensional representation – we first normalise the respective feature values (each independently) based on the observed minimum and maximum values, and then calculate the cosine similarity:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{xy}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^n \mathbf{x}_i \mathbf{y}_i}{\sqrt{\sum_{i=1}^n (\mathbf{x}_i)^2} \sqrt{\sum_{i=1}^n (\mathbf{y}_i)^2}} \quad (1)$$

We employ the cosine similarity in our optimisation algorithms as the fitness function to guide the search toward summaries that are close to a target vector.

### 3.3 Algorithmic Approaches

Extractive multi-document summarisation can be seen as an optimisation problem where the source documents form a collection of sentences, and the task is to select an optimal subset of the sentences under a length constraint [11]. In this study, we aim to generate summaries with up to five sentences as this is approximately the length of the summaries that the students have written.

We now present our optimisation algorithms to automatically produce summaries from heterogeneous artefacts for a given time frame. We utilise five algorithms, and we also create summaries at random to estimate a lower performance bound. We use the aforementioned cosine similarity as the scoring function, which computes either the word-similarity or the feature-similarity with respect to a given target. In our case, the targets are the summaries in the gold standard. By doing so, we aim at capturing the developers' activities found in the software artefacts that were created or updated in the given time frame and that are cited in the gold-standard summaries to generate human-like summaries.

Our first approach is a brute force algorithm, which exhaustively evaluates all subsets of up to a given target size. We will use this as a performance reference, because we do not know a-priori what good cosine-similarity values are.

The second algorithm is a greedy approach (Algorithm 1). It iteratively builds up a summary sentence-by-sentence: in each iteration, it determines the best-suited additional sentence and then adds it – unless the addition of even the best-suited sentence would result in a worsening of the cosine similarity.

In addition, we use three variations of random local search (RLS) algorithms. First, RLS-unrestricted (see Algorithms 2) can create summaries without being restricted by a target length. Second, RLS-restricted is like RLS-unrestricted, but it can only generate summaries of at most a given target length. Third, RLS-unrestricted-subset runs RLS-unrestricted first, but it then runs the brute force approach to find the best summary of at most a given target length. These

---

**Algorithm 1:** Greedy algorithm

---

**Input:**  $AS$  - artefacts' sentences,  $SS$  - student summary, and  $TLGS$  - targeted length of the generated summary.  
**Output:**  $GS$  generated summary  
 $GS \leftarrow \emptyset$   
**while** ( $len(GS) \leq TLGS$ ) **do**  
   $K \leftarrow \emptyset \{K: \text{unused sentences in } AS\}$   
   $K_{best} \leftarrow \emptyset \{\text{best single sentence to add in this iteration}\}$   
  **for all** ( $K_i \in K$ ) **do**  
    **if**  $cosSimilarity(GS + K_i, SS) \geq cosSimilarity(GS + K_{best}, SS)$  **then**  
       $K_{best} \leftarrow K_i$   
    **if**  $cosSimilarity(GS + K_{best}, SS) < cosSimilarity(GS, SS)$  **then**  
      **return**  $GS$  {do not add  $K_{best}$  if it worsens the similarity}  
**return**  $GS$

---



---

**Algorithm 2:** Random Local Search with unrestricted summary length (RLS-unrestricted)

---

**Input:**  $AS$  - artefacts' sentences and  $SS$  - student summary  
**Output:**  $GS$  generated summary  
 $GS \leftarrow \emptyset$   
**while** (running time < 10 seconds) **do**  
   $GS_{temp} \leftarrow GS$   
  select a sentence  $AS_r$  from  $AS$  u.a.r. and flip its inclusion status in  $GS_{temp}$   
  **if**  $cosSimilarity(GS_{temp}, SS) \geq cosSimilarity(GS, SS)$  **then**  
     $GS \leftarrow GS_{temp}$   
**return**  $GS$

---

algorithms share common characteristics, such as the execution time limit and the ability to explore the search space by including and excluding sentences. One notable characteristic of RLS-unrestricted is that it can produce summaries that exceed the target length. We have done this to provide an indication of whether five sentences were enough to create close summaries.

As the sixth approach, we use a random search as a naive approach to provide a lower performance bound: it iteratively creates summaries of five sentences, and it returns (when the time is up) the best randomly created five-sentence summary.

Note that the student summary ( $SS$ ) that we provide as an input to all approaches can either be an actual summary (i.e., the words) in which case the co-occurrence is calculated, or it can be a summary represented as a feature vector in the high-dimensional feature space.

Lastly, to investigate the impact of the individual artefacts on the summaries, we consider three scenarios as input source to generate summaries by each of the algorithms at a given time window: 1) each of the artefacts listed in Table 1 is considered individually as a source, 2) combining all the 15 artefacts in a single

source, and 3) assuming we know a developer’s preferences for particular types of artefacts, we only consider sentences coming from these types.

*Implementation Note.* We remove a-posteriori all the cases when we encountered at least one empty summary for two reasons: (1) the word similarity between a generated summary and the student’s summary can be zero, and (2) we encountered co-linear vectors even in the 26-dimensional space. Generating summaries from all artefacts as an input source, we detected 670 and 1065 empty summaries generated from all algorithms using the word similarity and feature similarity, respectively. On the other hand, we found 845 and 980 empty summaries generated by all algorithms using word similarity and feature similarity, respectively, when the most relevant artefacts considered as an input source.

## 4 Computational Study and Discussion

In our experiments, we consider the 503 summaries written by students, 6 algorithms, and three scenarios (i.e., the sentences’ sources).

For both similarity measures, we use the gold standard as the target, i.e., the students’ original summaries either as bags of words or as high-dimensional feature vectors. An alternative for the feature similarity is to use, e.g., the average vector across all students to aim at the “average style”, however, then it would not be clear anymore if it can be approximated. As this is the first such study, and in order to study the problem and the behaviour of the algorithms in this extractive setting under laboratory conditions, we aim for the solutions defined in the gold standard.

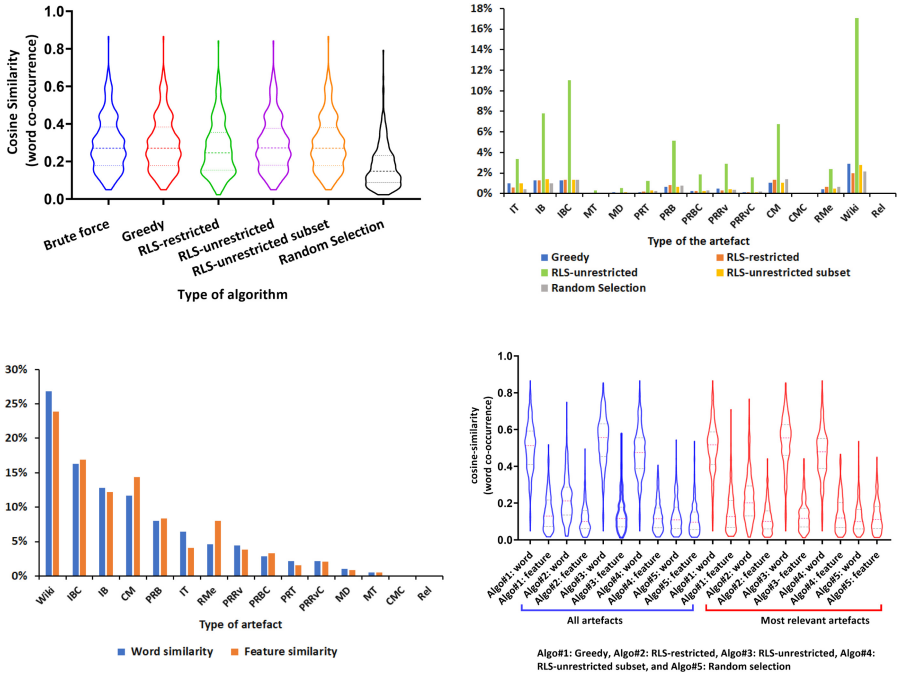
*A Comparison with Brute Force.* To better understand what quality we can expect from our five randomised approaches, we compare these approaches with our brute force approach to extractive summarisation. The artefact type for this first investigation is “issue title”. The maximum number of sentences here per project and summary combination was 35. For our brute force approach, this resulted in a manageable number of  $324,632 + 52,360 + 6,545 + 595 + 35 = 384,167$  subsets of up to five sentences for that particular week. The computational budget that we give each RLS variant is 10s.

Comparing the results obtained by these algorithms (see Fig. 2a), we can observe that the Greedy algorithm has the ability to generate summaries whose overall distribution is close to the distribution of summaries generated by brute force.<sup>2</sup> Similarly, the two RLS-unrestricted approaches also produce comparable summaries. RLS-restricted performs worse, but still better than the Random Selection.<sup>3</sup> From this first comparison, we conclude that Greedy is a very good approach, as it achieves a performance comparable to that of brute force (which is our upper performance bound), while it requires only 0.49s on average to form

<sup>2</sup> Based on a two-sided Mann-Whitney U test, there is no statistically significant difference at  $p=0.05$  between Greedy and Brute Force.

<sup>3</sup> Let us recall let Random Selection does not generate only one summary at random, but many until the time limit is reached, and it then returns the best.

a summary compared to other algorithms.<sup>4</sup> We can moreover conclude that a maximum summary length of five is acceptable, as the *RLS-unrestricted subset* does not perform differently from the others that were restricted.



**Fig. 2.** Results of the computational study. (a): Cosine similarity based on word co-occurrence of the generated summaries. (b): Average contribution of artefacts to summaries, aggregated across the two similarity measures. (c): Average contribution of artefacts to summaries, aggregated across all algorithms. (d): Similarities: when all artefacts are used (blue, overall average 0.266) and when only the relevant eight are used (red, overall average 0.258). (Color figure online)

To explain Greedy’s performance, and to explain that the performances of Greedy and of some of the RLS variants is very comparable, we conjecture that the problem of maximising the cosine-similarity w.r.t. a target vector given a set of vectors is largely equivalent to a submodular pseudo-Boolean function without many local optima. A formal proof of this, however, remains future work.

*Used Types of Artefacts.* Next, use each algorithm to create a weekly summary for the cases where we have student summaries. In particular, we investigate from which artefact types the sentences are taken from in these generated summaries.

<sup>4</sup> The average running time per algorithm (in seconds) to generate a summary is, from left: 151.92, 0.49, 10.0, 10.0, 10.20, 6.67 s.

In total, there are 22,313 (39.73% of the total) sentences found in the source input linked to the students' summaries. Note that while this number appears to be very large, it includes the very large summaries produced by RLS-unrestricted (average length 29.6), and we are nevertheless aiming at hundreds of different target summaries for one-week time-windows, which thus appear to require very different sentences from the artefacts.

In Fig. 2b, we can see that the generated summaries by each of the algorithms are composed of sentences from almost all of the artefact types. In particular, we can note that sentences from wiki pages are most commonly used. Possible reasons for this include that (1) wiki pages make up the largest fraction of the source sentences, and (2) developers might have best described their activities on the wiki pages.

In Fig. 2c, we can see that content from wiki artefacts contributed around 27% to the summaries generated by all algorithms. Also, sentences found in issue bodies (IB), issue body comments (IBC), and commit messages (CM) contributed 13%–17%. On the other hand, artefacts such as pull requests reviews (PRRv), pull requests title (PRT) and milestone titles (MT) have among the lowest contributions, which indicates that the students did not commonly use these artefacts – or at least mention them and related content – during their project's development life cycle.

*Generating Summaries Based on the Most Relevant Artefacts.* By generating summaries based on the most relevant artefacts found in the students' original summaries, we aim at generating more human-like summaries that better reflect the developers' preferences for certain artefact types. To achieve this, we consider the generated summaries as a starting point, as each of them was generated to be similar to a particular student summary, and hence it can indirectly reflect a student's preference. Then, we identify the most relevant ones by using the median as the cut-off (i.e., based on Fig. 2c). As a result of this selection, the eight most commonly referred to artefacts are (from most common to least common): wiki, issue title, issue bodies, issue body comments, commit messages, pull request bodies, readme files, and pull requests reviews. In total, this reduces the number of candidate sentences by 10.5% to 50,246.

We now investigate the performance of the subset of artefacts in terms of being able to generate good summaries. Fig. 2d shows the cosine word co-occurrence similarity and feature similarity achieved by each of the algorithms. Blue violin plots show the distributions of similarities achieved when all 15 artefacts were considered, and the red violin pots show the same for the eight most relevant artefacts. As we can see, focusing on only eight artefact types appears to have little to no negative impact.

## 5 Expert Annotation

To evaluate the extent to which the summaries that the different approaches generate matched the summaries written by the students *in the perception of*

**Table 3.** Average rating from each annotator for output produced by the different approaches.

Approach	Annotator 1	Annotator 2
Word (all)	3.7	3.3
Word (subset)	<b>3.8</b>	<b>3.5</b>
Feature (all)	3.7	2.0
Feature (subset)	3.7	2.0
Random (all)	3.5	1.8
Random (subset)	3.0	1.8

*software developers*, we asked two expert annotators to evaluate the results – both were in their first year of study of a Computer Science PhD, and both not affiliated with this study. Both annotators indicated that developing software is part of their job, and they have 4–6 years of software development experience. Annotator 1 stated that they had 1–2 years of experience using GitHub for project development, Annotator 2 answered the same question with 2–4 years.

The selection of algorithms to be used for expert annotation is based on the highest median value of the cosine similarities between the gold standard summaries and the generated summaries from each of the algorithms. Therefore, summaries generated by the Greedy algorithm were chosen for the annotation.

For the study, we randomly selected ten out of the total of fourteen weeks, and for each week, we randomly selected one project. For each of these ten, we then produced six different summaries in relation to the gold standard (i.e., the summaries written by the students):

1. the best summary based on *word similarity* between sentences contained in all artefacts in the input data (issues, pull requests, etc.) and the gold standard student summary,
2. same as (1), but only using the eight most relevant artefacts as input data,
3. the best summary based on *feature similarity* between sentences contained in all artefacts in the input data and the gold standard student summary,
4. same as (3), but only using the eight most relevant artefacts as input data,
5. a random baseline by randomly selecting five sentences from all artefacts,
6. same as (5), but only using the eight most relevant artefacts as input data.

We created a questionnaire, which asked the annotators first to produce a summary for the ten selected weeks after inspecting the corresponding GitHub repositories (to ensure that annotators were familiar with the projects), and then to rate each summary on a Likert-scale from 1 (strongly disagree) to 5 (strongly agree) in response to the question “Please indicate your agreement with the following statement: The summary mentions all important project activities present in the gold standard summary”.

Table 3 shows the results, separately per annotator. While it is apparent from the data that Annotator 1 generally gave out higher scores than Annota-

tor 2, both annotators perfectly agreed on the (partial) order of the different approaches: Word (subset)  $\geq$  Word (all)  $\geq$  Feature (subset)  $\geq$  Feature (all)  $\geq$  Random (all)  $\geq$  Random (subset).

In summary, approaches based on text similarity achieve the best result in terms of human perception, followed by approaches based on feature similarity, and the random baselines.

## 6 Threats to Validity

Our study, like many other studies, has a number of threats that may affect the validity of our results.

First, our research subjects involved summaries written by graduate and undergraduate students. Although it is possible that Master students are more knowledgeable about interacting with the GitHub platform than the Bachelor students, the difference in the experiences of both subjects should not affect the results. This is because the students require an intermediate level of skills to work with the GitHub platform.

Our result, illustrated in Table 3, shows that the eight most relevant artefacts are found to be sufficient when generating summaries containing developers' activities. These types – such as issues, pull requests, and commits – are essential elements of a GitHub repository. However, as these are essential elements of probably any software repository, we expect this finding to be transferable to other repositories.

Also, evaluating the automatically generated summaries relies on human experts. Subjectivity and bias are likely to be issues when the number of human experts involved to assess the generated summaries is small. Hence, we plan, for future work, to include more experts to mitigate these issues.

## 7 Conclusion and Future Work

Software engineering projects produce many artefacts over time, ranging from wiki pages, to pull request and issue comments. Summarising these can be helpful to a developer, for example, when they return from a holiday, or when they try to get an overview of the project's background in order to move forward with their team.

In this article, we have presented the first framework to summarise the heterogeneous artefacts produced during a given time window. We have defined our own gold standard and ways of measuring similarity on a text-based level. Then, we proceeded to compare various optimisation heuristics using different input scenarios, and have found that a greedy algorithm can generate summaries that are close to the human-written summaries in less running time compared to other algorithms. A study then has found that experts preferred the combination that used word similarity to generate summaries based on the eight most relevant artefacts.

Interestingly, the generated summaries have been found useful even though the optimisation approaches have not yet considered temporal connections between the sentences and also not yet the actual meaning. In the next steps, we will focus on these two to further improve the quality of the summaries. An additional, larger study with GitHub users will aim at the use of averaged and personalised target vectors.

**Acknowledgements.** Mahfouth has been sponsored by the Institute of Public Administration (IPA), Saudi Arabia. Christoph's and Markus' work has been supported by the Australian Research Council projects DE180100153 and DE160100850, and by the 2019 Google Faculty Research Award "Rewriting software documentation for non-native speakers".

## References

1. Alghamdi, M., Treude, C., Wagner, M.: Toward human-like summaries generated from heterogeneous software artefacts. In: Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, pp. 1701–1702. ACM (2019). ISBN 9781450367486
2. Baxendale, P.B.: Machine-made index for technical literature—an experiment. *IBM J. Res. Dev.* **2**(4), 354–361 (1958)
3. Erkan, G., Radev, D.R.: LexRank: graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.* **22**, 457–479 (2004)
4. GitHub. The State of the octoverse, February 2020. <https://octoverse.github.com/>
5. Kupiec, J., Pedersen, J., Chen, F.: A trainable document summarizer. In: 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 68–73 (1995)
6. Luhn, H.P.: The automatic creation of literature abstracts. *IBM J. Res. Dev.* **2**(2), 159–165 (1958)
7. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
8. Nair, V., et al.: Data-driven search-based software engineering. In: 15th International Conference on Mining Software Repositories (MSR), Gothenburg, Sweden, pp. 341–352. ACM (2018). ISBN 9781450357166
9. Nazar, N., et al.: Source code fragment summarization with small scale crowd-sourcing based features. *Front. Comput. Sci.* **10**(3), 504–517 (2016)
10. Nenkova, A., McKeown, K.: Automatic summarization. *Found. Trends Info. Retr.* **5**(2–3), 103–233 (2011)
11. Peyrard, M., Eckle-Kohler, J.: A general optimization framework for multi-document summarization using genetic algorithms and swarm intelligence. In: 26th International Conference on Computational Linguistics: Technical Papers (COLIN), pp. 247–257 (2016)
12. Rastkar, S., Murphy, G.C., Murray, G.: Automatic summarization of bug reports. *IEEE Trans. Softw. Eng.* **40**(4), 366–380 (2014)
13. Rigby, P.C., Robillard, M.P.: Discovering essential code elements in informal documentation. In: 35th International Conference on Software Engineering (ICSE), pp. 832–841. IEEE (2013)
14. Sohangir, S., Wang, D.: Improved sqrt-cosine similarity measurement. *J. Big Data* **41**, 25 (2017)



15. Torres-Moreno, J.-M.: *Automatic Text Summarization*. Wiley, Boca Raton (2014)
16. Treude, C., Filho, F.F., Kulesza, U.: Summarizing and measuring development activity. In: 10th Joint Meeting on Foundations of Software Engineering (FSE), pp. 625–636 (2015)
17. Verma, P., Om, H.: Extraction based text summarization methods on user’s review data: a comparative study. In: Unal, A., Nayak, M., Mishra, D.K., Singh, D., Joshi, A. (eds.) *SmartCom 2016*. CCIS, vol. 628, pp. 346–354. Springer, Singapore (2016). [https://doi.org/10.1007/978-981-10-3433-6\\_42](https://doi.org/10.1007/978-981-10-3433-6_42)
18. Ying, A.T.T., Robillard, M.P.: Code fragment summarization. In: 9th Joint Meeting on Foundations of Software Engineering (FSE), pp. 655–658 (2013)



# A Search for Additional Structure: The Case of Cryptographic S-boxes

Claude Carlet<sup>1,2</sup>, Marko Djurasevic<sup>3</sup>, Domagoj Jakobovic<sup>3</sup>,  
and Stjepan Picek<sup>4</sup>(✉)

<sup>1</sup> Department of Informatics, University of Bergen, Bergen, Norway  
claude.carlet@gmail.com

<sup>2</sup> Department of Mathematics, Universities of Paris VIII and XIII, Paris, France

<sup>3</sup> Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3,  
Zagreb, Croatia

{marko.durasevic, domagoj.jakobovic}@fer.hr

<sup>4</sup> Cyber Security Research Group, Delft University of Technology, Mekelweg 2,  
Delft, The Netherlands  
s.picek@tudelft.nl

**Abstract.** We investigate whether it is possible to evolve cryptographically strong S-boxes that have additional constraints on their structure. We investigate two scenarios: where S-boxes additionally have a specific sum of values in rows, columns, or diagonals and the scenario where we check that the difference between the Hamming weights of inputs and outputs is minimal. The first case represents an interesting benchmark problem, while the second one has practical ramifications as such S-boxes could offer better resilience against side-channel attacks.

We explore three solution representations by using the permutation, integer, and cellular automata-based encoding. Our results show that it is possible to find S-boxes with excellent cryptographic properties (even optimal ones) and reach the required sums when representing S-box as a square matrix. On the other hand, for the most promising S-box representation based on trees and cellular automata rules, we did not succeed in finding S-boxes with small differences in the Hamming weights between the inputs and outputs, which opens an interesting future research direction. Our results for this scenario and different encodings inspired a mathematical proof that the values reached by evolutionary algorithms are the best possible ones.

## 1 Introduction

S-boxes are functions with an important role in cryptography as they are the only source of nonlinearity for many cryptographic algorithms. Without S-boxes with good cryptographic properties, many designs would be easy to break by running cryptanalyses [1, 2]. A common option to obtain cryptographically strong S-boxes is to use algebraic constructions [3]. Still, there are alternatives to algebraic constructions. If the construction constraints are not too difficult, the random

search can work well. Besides algebraic constructions and random search, various heuristic techniques showed their potential [4]. Heuristics play an important role, especially in the cases where one requires S-boxes with cryptographic properties that are not obtainable by the known algebraic constructions (note, that there are not too many known algebraic constructions [3]). We explore whether we can design cryptographically strong S-boxes (with good cryptographic properties) that have additional properties of structure. We consider two settings: obtaining cryptographic S-boxes that have an arrangement of elements that result in 1) the same sum for every row, column, or diagonal, or 2) the minimal difference between the Hamming weights of the inputs and outputs.

Finding additional structure in S-boxes while keeping very good mandatory cryptographic features is not an easy task. We do not know if there exists such structure, or even if it does, whether it occurs for various S-box sizes. Next, if we assume there is additional structure, there is no prior knowledge about how difficult it is to reach it with heuristics. What is more, we may not be able to find such solutions depending on the selection of solution encoding or fitness functions. Finally, it is not known if there are trade-offs between cryptographic properties and the properties denoting additional structure.

Exploring the S-boxes depicted as square matrices with sums of rows, columns, or diagonals equal to a specific value does not have practical cryptographic applications to the best of our knowledge. This is because what we search for is not affine invariant, contrary to many notions in cryptography [3]. Still, we consider it an interesting benchmark problem as now, for smaller S-box sizes, we can easily obtain optimal cryptographic properties with heuristics, while for larger S-boxes, heuristics cannot give results close to algebraic constructions. We consider this additional constraint an interesting bridging option to allow more fine-grained evaluation of heuristics for S-box construction. By imposing additional constraints, we enable heuristics to reduce the search space size. Note that this structure constraint requires that an S-box is also a magic square [5].

The second constraint ensuring that the Hamming weights of S-box inputs ( $x$ ) and outputs ( $F(x)$ ) are as close as possible, has more practical ramifications. S-boxes are common targets for side-channel attacks [6]. In such attacks, one needs to consider an appropriate leakage model that the cryptographic device follows. A common model is the Hamming weight model, where the power consumption is proportional to the Hamming weight of a processed value. Minimizing the difference in the Hamming weights could potentially make the S-box more resilient against side-channel attacks, as explained in more detail in Sect. 2.

To the best of our knowledge, there are no previous works that consider evolving cryptographically strong S-boxes with additional constraints on the arrangement of elements to result in a specific sum, or to minimize the differences in the Hamming weights. In the rest of this paper, whenever talking about S-boxes, we consider those that are cryptographically strong (details are given in Sect. 2). While there is not much previous work on exploring additional structure in S-boxes, there are works that use evolutionary algorithms to evolve S-boxes.

For example, Clark et al. used the principles from the evolutionary design of Boolean functions to evolve S-boxes for sizes up to  $8 \times 8$  [7]. Picek et al. developed an improved fitness function that enables evolutionary algorithms to find higher nonlinearity values for several S-box sizes [8]. Mariot et al. investigated the genetic programming approach to evolve cellular automata rules that are then used to generate S-boxes [9]. The results obtained with GP used to evolve CA rules outperform any other metaheuristics for sizes  $5 \times 5$  up to  $7 \times 7$ . Jakobovic et al. presented a fitness landscape analysis for S-boxes of various sizes [10]. Finally, Picek et al. used evolutionary algorithms to find S-boxes that have good cryptographic properties but also good side-channel resilience [11].

In this paper, we explore how evolutionary algorithms can be used to construct S-boxes with additional structure. Toward that goal, we first define several S-box sizes we investigate with both single-objective and multi-objective approaches. Our results indicate that it is possible to construct S-boxes with sums of rows and columns equal to a specific value. We find a more difficult condition when adding the constraint on the sum of diagonals. We could not find any such S-box with optimal cryptographic properties but also having the sum of all rows, columns, and diagonals equal to a specific value (thus, producing S-box that is also a magic square).

Our experiments show it is possible to obtain an S-box with optimal cryptographic properties and a small difference between the Hamming weights of inputs and outputs. Then, our experimental results inspire mathematical research proving that the nonlinearity must be equal or smaller than the sum of differences of the Hamming weights. Finally, our results show that the solution encoding (tree-based) that works the best for cryptographic properties performs the worst for these additional, structure-based properties.

## 2 Background

Let  $n, m$  be positive integers.  $\mathbb{F}_2^n$  is the  $n$ -dimensional vector space over  $\mathbb{F}_2$  and by  $\mathbb{F}_{2^n}$  the finite field with  $2^n$  elements. The set of all  $n$ -tuples of elements in the field  $\mathbb{F}_2$  is denoted by  $\mathbb{F}_2^n$ , where  $\mathbb{F}_2$  is the finite field with two elements. For any set  $S$ , we denote  $S \setminus \{0\}$  by  $S^*$ . The addition of elements of the finite field  $\mathbb{F}_{2^n}$  is represented with “+” while the inner product of  $a$  and  $b$  equals  $a_1x_1 + \dots + a_nx_n$ .

### 2.1 S-boxes – Representations and Properties

An S-box (Substitution box) is a mapping  $F$  from  $n$  bits into  $m$  bits (thus, S-box is an  $(n, m)$  function). An  $(n, m)$ -function  $F$  can be defined as a vector  $F = (f_1, \dots, f_m)$ , where the Boolean functions  $f_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$  for  $i \in \{1, \dots, m\}$  are called the coordinate functions of  $F$ . The component functions of an  $(n, m)$ -function  $F$  are all the linear combinations of the coordinate functions with non all-zero coefficients. For every  $n$ , there exists a field  $\mathbb{F}_{2^n}$  of order  $2^n$ , so we can endow the vector space  $\mathbb{F}_2^n$  with the structure of that field when convenient.

In Table 1, we give the best known/possible results for two relevant cryptographic properties and S-box dimensions we consider.

An  $(n, m)$ -function  $F$  is balanced if it takes every value of  $\mathbb{F}_2^m$  the same number  $2^{n-m}$  of times. If a function  $F$  is balanced, then it is a permutation (the function is bijective, i.e.,  $n = m$ ).

The Walsh-Hadamard transform of an  $(n, m)$ -function  $F$  is (see, e.g., [3]):

$$W_F(a, v) = \sum_{x \in \mathbb{F}_2^n} (-1)^{v \cdot F(x) + a \cdot x}, \quad a, v \in \mathbb{F}_2^m. \tag{1}$$

The nonlinearity  $nl_F$  of an  $(n, m)$ -function  $F$  equals the minimum nonlinearity of all its component functions  $v \cdot F$ , where  $v \in \mathbb{F}_2^{m*}$  [3, 12]:

$$nl_F = 2^{n-1} - \frac{1}{2} \max_{\substack{a \in \mathbb{F}_2^n \\ v \in \mathbb{F}_2^{m*}}} |W_F(a, v)|. \tag{2}$$

Let  $F$  be a function from  $\mathbb{F}_2^n$  into  $\mathbb{F}_2^m$  with  $a \in \mathbb{F}_2^n$  and  $b \in \mathbb{F}_2^m$ . We write:

$$D_F(a, b) = \{x \in \mathbb{F}_2^n : F(x) + F(x + a) = b\}. \tag{3}$$

The entry at the position  $(a, b)$  corresponds to the cardinality of the delta difference table  $D_F(a, b)$  and we write it as  $\delta_F(a, b)$ . The differential uniformity  $\delta_F$  is then defined as [13]:

$$\delta_F = \max_{a \neq 0, b} \delta_F(a, b). \tag{4}$$

**Table 1.** Best known values for bijective S-boxes. For  $8 \times 8$ , we give the best known results while for smaller sizes, we give the optimal values. For bijective S-boxes (and in  $\mathbb{F}_2$ ), both nonlinearity and differential uniformity can be even values only. The worst possible values are 0 for nonlinearity (i.e., the S-box is linear), and  $2^n$  for differential uniformity.

Property	$3 \times 3$	$4 \times 4$	$5 \times 5$	$6 \times 6$	$7 \times 7$	$8 \times 8$
$nl_F$	2	4	12	24	56	112
$\delta_F$	2	4	2	2	2	4

## 2.2 Side-Channel Attacks

Besides various cryptanalysis techniques, another category of attacks on cryptographic targets is side-channel attacks (SCAs). In such attacks, one does not aim at the weakness of the algorithm, but on the weaknesses of implementation [6]. For instance, one could observe the power consumption of a device while running the cryptographic algorithm, and compare it with hypotheses for every possible key (or, more precisely, subkey). When a particular statistic technique

(for instance, Pearson correlation) shows the best absolute value of correlation, we assume that the key hypothesis is correct, and we use it to break the target. For this to work, we need to assume the leakage model in which a device leaks, where one of the standard models is the Hamming weight model. If the input and output of the S-box have the same Hamming weights, this will result in several equally likely key hypotheses (they will have the same correlation). As such, the side-channel attack in this leakage model would become somewhat more difficult to succeed.

### 2.3 Magic Squares

A magic square is a  $d \times d$  square grid that consists of distinct positive integer values in the range  $[1, d^2]$ . The sum of values in every row, column, and diagonal is equal. That sum value is called the magic constant of the magic square. There is no strict requirement on the value of the magic constant. Usually, the sum is determined as the sum of all elements occurring in the magic square and divided by the number of cells on each side ( $d$ ), i.e.,  $d(d^2 + 1)/2$ .

## 3 Experiments

We consider S-boxes of sizes  $n \times n$  only. As a consequence, our S-boxes can be bijective, but there is no constraint on this for all encodings. The smallest S-box size we work with is  $3 \times 3$ , while the largest is  $8 \times 8$  (the smallest and largest practically used S-box sizes). The set of experiments is divided into two groups. The first group of experiments deals with the evolution of S-boxes with a constraint that is imposed on the sums of its rows, columns, and diagonals. More precisely, we can depict an S-box as a square matrix. Then, each row, column, or diagonal in that matrix must sum up to the same value (i.e., an S-box is magic square). As an example, let us consider an S-box of size  $4 \times 4$  that has 16 elements. We can depict them in a matrix of size  $4 \times 4$  where we fill it with S-box values column by column. Finally, we can easily check the sum of each row, column, or diagonal in such a matrix. To ensure that it is possible to obtain such S-boxes, we place the following constraints in our experiments:

1. We consider S-boxes of even dimensions only, and more precisely, dimensions 4, 6, and 8. Recall from Sect. 2 that S-boxes are defined as elements of finite fields where the underlying field is the finite field with two elements, i.e.,  $\mathcal{F}_2$ . As such, the number of elements in an S-box equals  $2^n$ , where  $n$  is the size of an S-box. Simultaneously, the number of elements in a magic square is equal to  $d^2$ , where  $d$  is the size of the magic square. For every odd dimension, the number of elements in a magic square with size  $d$  is not the same as the number of elements in a finite field of size  $2^n$ , regardless of the fact if  $d = n$ .
2. Commonly, the elements in a magic square are denoted from 1 to  $d^2$  while the elements in an S-box with elements from 0 to  $2^n - 1$ . This does not represent a problem as in the finite field; all elements are calculated *modulo*  $2^n$ , which means that 0 and  $2^n$  ( $d^2$ ) represent the same values.

The second group of experiments will focus on evolving S-boxes where the difference in the Hamming weights for every pair of S-box input and output is minimal. Unlike the previous scenario, where the experiments were constrained to only even-sized S-boxes, now, there is no such constraint. Consequently, for this set of experiments, we test S-boxes from  $3 \times 3$  up to  $8 \times 8$  size.

### 3.1 Experimental Setup

The first set of experiments consider the case in which the additional constraints are placed on the sum of elements in rows, columns, and diagonals. The second round of experiments, in addition to cryptographic properties, also places the constraint on the difference in Hamming weights between the inputs and outputs of the S-box. The experiments evaluate both the bijective and non-bijective S-boxes. For that purpose, three solution representations are applied, out of which the first two are used with a genetic algorithm (GA) and the third with genetic programming (GP):

1. The integer genotype, consisting of a vector of integer values of size  $2^n$  and values in the range  $[0, 2^n - 1]$ .
2. The permutation genotype of size  $2^n$ , where each value in  $[0, 2^n - 1]$  occurs only once.
3. The tree genotype, which represents the transition function of cellular automata (CA).

In our experiments, the integer genotype is used to evolve non-bijective S-boxes, while the permutation and tree genotype are used to evolve bijective S-boxes. The integer representation is a super-set of the permutation representation, and it is possible, although very unlikely, to obtain bijective S-boxes with integer genotype. The integer genotype mutation selects a random position in the vector and assigns it a new value in the defined range. The crossover operators for integer vector include a simple one-point and two-point recombination, as well as an averaging crossover which defines all the elements of the child vector as an arithmetic mean of the corresponding values in the parent vectors. For the permutation genotype, we use three mutation operators and five crossover operators where we chose among the most common ones in practice. The mutation operators are insert mutation, inversion mutation, and swap mutation [14]. We used partially mapped crossover (PMX), position based crossover (PBX), order crossover (OX), uniform like crossover (ULX), and cyclic crossover [15]. For each new individual, an operator is selected uniformly at random between all operators within a class (both mutation and crossover).

Finally, we use the tree representation, where GP is used to evolve a suitable cellular automata function in the form of a tree. The input bits of the S-box are used as terminal nodes of the tree, where their number is equal to  $n$ . The function set consists of Boolean primitives with 1) two inputs: NOT, which inverts its argument, XOR, AND, OR, NAND, and XNOR, and 2) three inputs: IF (it takes three arguments and returns the second one if the first evaluates to

*true*, and the third one otherwise). The evolved function is used as a transition that, based on the input bits of the current state, calculates the output bits which act as the next state. For details about the cellular automata approach for S-box evolution, we refer readers to [9]. The recombination operators for this representation are simple tree crossover, uniform crossover, size fair, one-point, and context preserving crossover [16] (selected at random) and subtree mutation.

Both GA and GP use a steady-state tournament algorithm with tournament size  $k = 3$  (select three individuals and remove the worst one, from the remaining two make an offspring and mutate it), and with individual mutation probability of 30%. All parameters were selected after a tuning phase, where the selected combinations showed good performance. The experiments for each considered configuration were executed 30 times (independent runs). The algorithms optimize a single-objective function, which is defined as a linear combination of individual criteria, as defined in the next section. In addition to the single-objective case, the multi-objective case using the well known NSGA-II algorithm [17] was also tested. Since the results achieved in the multi-objective case were equally good or worse than in the single-objective case, these results are not further discussed in the paper.

### 3.2 Fitness Functions

In addition to the cryptographic properties, the fitness function has to include an additional term to ensure that the algorithm can evolve S-boxes of the desired structure. When considering structures that have the sums on row, column, and diagonal elements equal to some predefined value, we use:

$$\begin{aligned}
 msq = \sum_i^n \left| mc - \sum_j^n sq[i][j] \right| + \sum_i^n \left| mc - \sum_j^n sq[i][j] \right| \\
 + \left| mc - \sum_i^n sq[i][i] \right| + \left| mc - \sum_i^n sq[i][n-i] \right|.
 \end{aligned} \tag{5}$$

Here,  $mc$  represents the magic constant, i.e., the number to which the elements in rows, columns, and diagonals need to be equal to when summed up, while  $sq$  represents a square consisting of elements of the S-box. The  $msq$  property calculates the distance for all relevant elements (rows, columns, diagonals) from the predefined constant. The previous equation is the most general version, which considers the sums on all the relevant elements of the square. Depending on the experiment, some elements in that expression will not be calculated when not all relevant elements need to adhere to the specified sum constraint. Although the constant to which the rows, columns, and diagonals need to be equal to can be freely specified, we use values to which the elements sum up in magic squares of the corresponding sizes [18]. For these constants, we know it is possible to construct a square with the requested structure, whereas choosing a random value could mean it would not be possible to obtain a square of numbers where



the sum of elements equals the selected number. For the  $4 \times 4$  S-box, the constant is 34. For the  $6 \times 6$  S-box, it equals 260, and for the  $8 \times 8$  S-box, it equals 2056.

The fitness function to be minimized is defined as:

$$f = msq - \alpha \frac{nl_F}{nl_{best}} - \beta \frac{(\delta_{worst} - \delta_F)}{\delta_{worst}}, \quad (6)$$

where  $nl_{best}$  represents the best-known values for nonlinearity as defined in Table 1,  $\delta_{worst}$  is the worst possible value for differential uniformity equal to  $2^n$  with  $n$  representing the number of inputs in the S-box, while  $\alpha$  and  $\beta$  represent scaling factors for the cryptographic properties. In the experiments, the emphasis is put on the  $sq$  property in a way that the other two properties are normalized and scaled with additional factors. This is because we want to see whether there are S-boxes that adhere to the additional structures, but still have good cryptographic properties.

Based on preliminary experiments, the values for both  $\alpha$  and  $\beta$  were set to the value of 0.5, which results in the  $msq$  property being treated as a primary objective, while the cryptographic properties are the secondary. In this way, the algorithm will always prefer a solution with a better structure, while the other two properties will then force the algorithm to search for those solutions that have better cryptographic properties.

The second set of experiments places a constraint on the difference in the Hamming weights of the inputs and outputs of the S-box. This property is calculated as:

$$HWD = |w_H(x) - w_H(F(x))|, \quad (7)$$

where  $x$  represents an input value to the S-box,  $F(x)$  is the S-box function that transforms the input, and  $w_H$  is the function returning the number of ones in the argument.  $HWD$  is simply defined as the absolute difference between the Hamming weights of the inputs in the S-box and outputs from the S-box. In this case, the fitness function which has to be minimized is:

$$f = HWD - \alpha \frac{nl_F}{nl_{best}} - \beta \frac{(\delta_{best} - \delta_F)}{\delta_{best}}. \quad (8)$$

If the  $HWD$  property would be optimized primarily by giving a smaller weight to the cryptographic properties, then the algorithms would always obtain the optimal solution for this property. This would lead to poor results for cryptographic properties, and as such, the evolved S-box would not have much use. When focusing on S-boxes that have such a structure, the primary focus is placed on evolving S-boxes with good cryptographic properties and then adjusting the solutions to conform to the desired structure. After executing some preliminary tests with different weight values, the  $\alpha$  and  $\beta$  coefficients are set to 10, which we evaluated to be enough for the algorithm to focus primarily on cryptographic properties, and only then on the difference of the Hamming weights.

### 3.3 Results

Table 2 shows the results obtained for the experiments in which additional constraints are placed on the sum of elements in rows, columns, and diagonals.

For each configuration, the S-box with the best fitness was selected, and the individual properties that constitute the fitness function are denoted. The *constrained elements* column denotes for which elements of the S-box the sum was calculated to be of the specified constant. For the size of  $4 \times 4$ , the algorithm obtained S-boxes that mostly have good cryptographic properties. For S-boxes in which the sum in rows or columns had to be equal to a certain sum, it was almost trivial for the algorithm to find the square with the optimal cryptographic properties. Still, when it is enforced that the sum on more elements (e.g., rows and columns) has to be equal to the desired constant, we found optimal solutions for the permutation encoding only. Interestingly, although the tree representation obtained S-boxes with optimal cryptographic properties, it was unable to obtain S-boxes that adhered to any of the additionally placed constraints.

For S-boxes of size  $6 \times 6$ , the algorithm demonstrated an interesting behavior. When the permutation and integer genotypes were applied, the algorithm had no problem with optimizing the additional constraints, since it evolved S-boxes that satisfied these constraints in all cases. On the other hand, in any of the experiments, were we able to obtain an S-box with optimal cryptographic properties. It is not surprising that the algorithm could not reach optimal cryptographic properties since this is a difficult task for EA when using this type of encoding. Additionally, the integer genotype achieved inferior results compared to the permutation genotype. The cellular automata rules evolved by GP demonstrated the opposite behavior since there, we found S-boxes with optimal cryptographic properties, but which did not satisfy the additional constraints.

For the  $8 \times 8$  S-boxes, the algorithm exhibited difficulties in obtaining S-boxes with an additional structure. For the permutation genotype, the algorithm evolved S-boxes that satisfy only the most simple constraint in which either the row or column sums are equal to the desired constant. For the integer genotype, the algorithm had fewer problems and obtained the desired structure for each constraint, which is expected since, in that case, it is much easier to construct an S-box with the desired structure. The S-boxes constructed by the permutation genotype have better cryptographic properties than those constructed by the integer genotype. The obtained objects have relatively good cryptographic properties, which are in line with the results for EA and S-boxes of that size [8]. The results obtained by the GP evolved CA are quite poor since this representation was neither able to evolve S-boxes with good cryptographic properties, nor which satisfied the constraints that were additionally placed upon the S-boxes for its structure.

It is not always easy to discern what are strong cryptographic properties as that depends on the whole cipher and not only the S-box part. For all tested sizes, it is possible to find S-boxes whose structure adheres to certain constraints. For the S-box size of  $4 \times 4$ , GA obtained S-boxes both with the desired structure and optimal cryptographic properties. This was not possible in GP's case since it was not possible to find an optimal S-box even for the least restrictive structure constraint. The cryptographic properties are further away from the optimal values for larger sizes, but they are still relatively good. For  $6 \times 6$  and  $8 \times 8$

and permutation encoding, cryptographic properties are the same for the most restrictive structure in which the elements in rows, columns, and diagonals have to sum up to a certain value, and in the structure where this is not required for the diagonal elements.

Table 3 presents the results obtained when, in addition to optimizing the cryptographic properties, the difference in the Hamming weights is used as an additional constraint. For the S-box size of  $3 \times 3$ , the algorithm obtained the same value for the integer and permutation genotypes. By additionally using exhaustive search, it was also proven that this is the optimal solution that can be obtained for this size. On the other hand, the algorithm did not obtain the optimal value by using the tree representation. This further shows that the evolved CA, although very powerful with dealing with only cryptographic properties, exhibits difficulties when S-boxes of certain structures are being evolved. For all other S-box sizes, the permutation and integer genotypes achieve a similar performance, which seems to demonstrate that both can be used for the considered problem. The only significant difference happens for  $8 \times 8$  S-box, where the integer genotype obtained a slightly better result for the nonlinearity property, but it also obtained a much worse value for the difference in Hamming weights.

In many cases, the obtained difference in Hamming weights between inputs and outputs is equal to the obtained nonlinearity value. This inspires us to ask a question of whether we found a lower bound on the sum of the difference of the Hamming weights. More precisely, whether for bijective S-boxes,  $\sum_{x \in F_2^n} |w_H(F(x)) - w_H(x)| \geq \min_{u, v \in F_2^n, \epsilon \in F_2, v \neq 0} d_H(v \cdot F, u \cdot x + \epsilon)$ ? It turns out this is true and can be mathematically proven. We have  $\sum_{x \in F_2^n} |w_H(F(x)) - w_H(x)| \geq d_H(v \cdot F(x), u \cdot x + \epsilon)$  when  $u = v$  equals the all-1 vector and  $\epsilon = 0$ , where  $d_H$  denotes the Hamming distance. Indeed, for each  $x$  such that  $v \cdot F(x) \neq u \cdot x$ , we have  $w_H(F(x)) \neq w_H(x)$ . Observe this is true as the nonlinearity of an S-box is the minimal nonlinearity of all its components. Then,  $\sum_{x \in F_2^n} |w_H(F(x)) - w_H(x)|$  is at least the distance between the component function  $v \cdot F(x)$  and the linear function  $u \cdot x$ . *A fortiori* it is at least the nonlinearity. To the best of our knowledge, the characterization between the maximal nonlinearity and the differences between the Hamming weights of input/output pairs is new. As such, we see that EAs not only managed to reach the optimal results for several S-box sizes, but they also inspired the new characterization of the differences of the Hamming weights concerning nonlinearity.

The CA evolved by GP once again demonstrates its superiority when considering only cryptographic properties. Nevertheless, the results obtained for the difference in Hamming weights are poor compared to the results for the other two genotypes. Thus, CA again does not seem to be fit to handle the evolution of S-boxes with additional structure. Although it could be said that this representation achieves poor results because for the S-boxes with better cryptographic properties it is not even possible to obtain good values for the difference in Hamming weights, the results obtained for the S-box of size  $8 \times 8$  disprove this, since, for that size, GP obtained poor results for both the cryptographic properties and the difference in Hamming weights. It simply seems that the restricted search

**Table 2.** Best results obtained with additional constraints imposed on the sums of rows, columns, and diagonals, single-objective optimization.

S-box size	Genotype	Constrained elements	msq	$nl_F$	$\delta_F$
$4 \times 4$	Integer	Rows	0	4	2
$4 \times 4$	Integer	Columns	0	4	4
$4 \times 4$	Integer	Rows and columns	0	3	4
$4 \times 4$	Integer	Rows, columns, and diagonals	0	2	6
$4 \times 4$	Permutation	Rows	0	4	4
$4 \times 4$	Permutation	Columns	0	4	4
$4 \times 4$	Permutation	Rows and columns	0	4	4
$4 \times 4$	Permutation	Rows, columns, and diagonals	0	2	8
$4 \times 4$	Tree	Rows	6	4	4
$4 \times 4$	Tree	Columns	6	4	4
$4 \times 4$	Tree	Rows and columns	26	4	4
$4 \times 4$	Tree	Rows, columns, and diagonals	49	4	4
$6 \times 6$	Integer	Rows	0	20	8
$6 \times 6$	Integer	Columns	0	19	8
$6 \times 6$	Integer	Rows and columns	0	7	8
$6 \times 6$	Integer	Rows, columns, and diagonals	0	7	10
$6 \times 6$	Permutation	Rows	0	20	6
$6 \times 6$	Permutation	Columns	0	20	6
$6 \times 6$	Permutation	Rows and columns	0	18	8
$6 \times 6$	Permutation	Rows, columns, and diagonals	0	18	8
$6 \times 6$	Tree	Rows	88	24	4
$6 \times 6$	Tree	Columns	88	24	4
$6 \times 6$	Tree	Rows and columns	220	24	4
$6 \times 6$	Tree	Rows, columns, and diagonals	422	24	4
$8 \times 8$	Integer	Rows	0	33	12
$8 \times 8$	Integer	Columns	0	29	12
$8 \times 8$	Integer	Rows and columns	0	18	12
$8 \times 8$	Integer	Rows, columns, and diagonals	0	11	12
$8 \times 8$	Permutation	Rows	0	100	8
$8 \times 8$	Permutation	Columns	0	100	10
$8 \times 8$	Permutation	Rows and columns	2	96	10
$8 \times 8$	Permutation	Rows, columns, and diagonals	4	96	10
$8 \times 8$	Tree	Rows	476	86	28
$8 \times 8$	Tree	Columns	5 760	84	36
$8 \times 8$	Tree	Rows and columns	8 440	80	32
$8 \times 8$	Tree	Rows, columns, and diagonals	10 169	82	26

space of CA is quite suitable for evolving S-boxes of good cryptographic properties (up to the size of  $8 \times 8$ ), but it exhibits problems when the S-boxes have to include an additional structure in them.

**Table 3.** Best results obtained by optimizing the difference in Hamming weights simultaneously with cryptographic properties.

S-box size	Genotype	HWD	$nl_F$	$\delta_F$
$3 \times 3$	Integer	2	2	2
$4 \times 4$	Integer	6	4	2
$5 \times 5$	Integer	10	10	4
$6 \times 6$	Integer	22	22	6
$7 \times 7$	Integer	46	46	8
$8 \times 8$	Integer	156	102	10
$3 \times 3$	Permutation	2	2	2
$4 \times 4$	Permutation	4	4	4
$5 \times 5$	Permutation	10	10	4
$6 \times 6$	Permutation	20	20	6
$7 \times 7$	Permutation	46	46	8
$8 \times 8$	Permutation	102	100	8
$3 \times 3$	Tree	6	2	2
$4 \times 4$	Tree	8	4	4
$5 \times 5$	Tree	20	12	2
$6 \times 6$	Tree	60	24	4
$7 \times 7$	Tree	84	48	8
$8 \times 8$	Tree	224	84	30

## 4 Conclusions and Future Work

The results we obtain suggest that the problem of evolving S-boxes with additional structure can be rather difficult. This should not come as a surprise as we know that S-boxes' design is a difficult task for evolutionary algorithms. The choice of genotype significantly influences the algorithm's ability to obtain S-boxes with additional structure. We managed to find cryptographically optimal S-boxes with sums of rows and columns equal to a specified constant. We also managed to obtain cryptographically optimal S-boxes with a small difference in the Hamming weights between inputs and outputs. We mathematically prove that the smallest sum of differences cannot be smaller than the nonlinearity value, which is a previously unknown result, inspired by evolutionary algorithms experiments and observations. In Fig. 1, we give examples of two evolved  $4 \times 4$

S-boxes for scenarios 1 and 2, respectively. In future work, we plan to explore whether we can obtain magic S-boxes if we consider some other patterns, e.g., looking at broken rows/columns/diagonals. Besides the difference in Hamming weights, we also plan to consider the Hamming distance metric.

7	6	10	11
1	15	2	16
14	8	9	3
12	5	13	4

(a) S-box with optimal cryptographic properties where the elements in rows and columns sum up to the value 34.

0	8	2	4
10	6	5	13
1	3	12	9
14	7	11	15

(b) S-box with optimal cryptographic properties where the difference of Hamming weights equals 4

**Fig. 1.** Examples of  $4 \times 4$  S-boxes with additional structure.

## References

1. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993). [https://doi.org/10.1007/3-540-47555-9\\_7](https://doi.org/10.1007/3-540-47555-9_7)
2. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991). [https://doi.org/10.1007/3-540-38424-3\\_1](https://doi.org/10.1007/3-540-38424-3_1)
3. Carlet, C.: Boolean Functions for Cryptography and Coding Theory. Cambridge University Press, Cambridge (2020)
4. Picek, S., Mariot, L., Yang, B., Jakobovic, D., Mentens, N.: Design of S-boxes defined with cellular automata rules. In: Proceedings of the Computing Frontiers Conference, CF 2017, New York, NY, USA, pp. 409–414. ACM (2017)
5. Chabert, J.L.: Magic Squares, pp. 49–81. Springer, Heidelberg (1999). <https://doi.org/10.1007/978-3-030-17993-9>
6. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Boston, MA (2007). <https://doi.org/10.1007/978-0-387-38162-6>
7. Clark, J.A., Jacob, J.L., Stepney, S.: The design of S-boxes by simulated annealing. *New Gener. Comput.* **23**(3), 219–231 (2005)
8. Picek, S., Cupic, M., Rotim, L.: A new cost function for evolution of s-boxes. *Evol. Comput.* **24**(4), 695–718 (2016). PMID: 27482748
9. Mariot, L., Picek, S., Leporati, A., Jakobovic, D.: Cellular automata based s-boxes. *Cryptogr. Commun.* **11**(1), 41–62 (2019)
10. Jakobovic, D., Picek, S., Martins, M.S.R., Wagner, M.: A characterisation of s-box fitness landscapes in cryptography. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, New York, NY, USA, pp. 285–293. ACM (2019)

11. Picek, S., Ege, B., Batina, L., Jakobovic, D., Chmielewski, L., Golub, M.: On using genetic algorithms for intrinsic side-channel resistance: the case of AES s-box. In: Proceedings of the First Workshop on Cryptography and Security in Computing Systems, CS2 2014, New York, NY, USA, pp. 13–18. ACM (2014)
12. Nyberg, K.: On the construction of highly nonlinear permutations. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 92–98. Springer, Heidelberg (1993). [https://doi.org/10.1007/3-540-47555-9\\_8](https://doi.org/10.1007/3-540-47555-9_8)
13. Nyberg, K.: Perfect nonlinear S-boxes. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 378–386. Springer, Heidelberg (1991). [https://doi.org/10.1007/3-540-46416-6\\_32](https://doi.org/10.1007/3-540-46416-6_32)
14. Eiben, A.E., Smith, J.E., et al.: Introduction to Evolutionary Computing, vol. 53. Springer, Heidelberg (2003). <https://doi.org/10.1007/978-3-662-05094-1>
15. Kellegöz, T., Toklu, B., Wilson, J.: Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem. *Appl. Math. Comput.* **199**(2), 590–598 (2008)
16. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk> (2008) (With contributions by J. R. Koza)
17. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Trans. Evol. Comput.* **6**(2), 182–197 (2002)
18. Abbott, S.: Most-perfect pandiagonal magic squares: their construction and enumeration, by kathleen ollerenshaw and david brée, p. 172. £19.50 (1998). ISBN 0 905091 06 x. (Institute of mathematics and its applications, 16 nelson st, southend-on-sea, ss1 1ef). *The Mathematical Gazette* 82(495), 535–536(1998)



# Evolutionary Multi-objective Design of SARS-CoV-2 Protease Inhibitor Candidates

Tim Cofala<sup>1</sup>, Lars Elend<sup>1</sup>, Philip Mirbach<sup>1</sup>, Jonas Prellberg<sup>1</sup>, Thomas Teusch<sup>2</sup>,  
and Oliver Kramer<sup>1</sup>(✉)

<sup>1</sup> Computational Intelligence Lab, Department of Computer Science,  
University of Oldenburg, Oldenburg, Germany  
{tim.cofala,lars.elend,philip.mirbach,jonas.prellberg,  
oliver.kramer}@uni-oldenburg.de

<sup>2</sup> Theoretical Chemistry Group, Department of Chemistry, University of Oldenburg,  
Oldenburg, Germany  
thomas.teusch@uni-oldenburg.de

**Abstract.** Computational drug design based on artificial intelligence is an emerging research area. At the time of writing this paper, the world suffers from an outbreak of the coronavirus SARS-CoV-2. A promising way to stop the virus replication is via protease inhibition. We propose an evolutionary multi-objective algorithm (EMOA) to design potential protease inhibitors for SARS-CoV-2's main protease. Based on the SELFIES representation the EMOA maximizes the binding of candidate ligands to the protein using the docking tool QuickVina 2, while at the same time taking into account further objectives like drug-likeness or the fulfillment of filter constraints. The experimental part analyzes the evolutionary process and discusses the inhibitor candidates.

**Keywords:** Evolutionary multi-objective optimization · Computational drug design · SARS-CoV-2

## 1 Introduction

At the time of writing this paper, researchers around the globe are searching for a vaccine or an effective treatment against the 2019 novel coronavirus (SARS-CoV-2). One strategy to limit virus replication is protease inhibition. A biomolecule called ligand binds to a virus protease enzyme and inhibits its functional properties. For SARS-CoV-2 the crystal structure of its main protease M<sup>Pro</sup> has been solved, e.g. by Jin *et al.* [16]. The search for a valid protease inhibitor can be expressed as optimization problem. As not only the binding of the ligand is an important objective, but also further properties like drug-likeness or filter properties, we comprise the molecule search problem as multi-objective optimization problem, which we aim to solve with evolutionary algorithms.

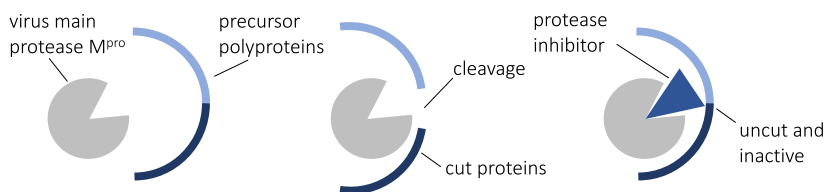


This paper is structured as follows. In Sect. 2 we shortly repeat the basics of protease inhibition and the connection to the novel coronavirus. Section 3 gives an overview of related work on evolutionary molecule design. In Sect. 4 we introduce molecule metrics, which we aim to optimize with the EMOA that is presented in Sect. 5. The experimental part in Sect. 6 presents our experimental results and discusses the evolved molecules. Conclusions are drawn in Sect. 7, where also prospective future research directions are presented.

## 2 Virus Protease Inhibition

As of late 2019, a novel respiratory disease named COVID-19 spread worldwide. COVID-19 is caused by SARS-CoV-2, which belongs to the coronavirus family like the well-known severe acute respiratory syndrome coronavirus (SARS-CoV). As RNA virus SARS-CoV-2's replication mechanism hijacks the cell mechanisms for replication. An essential part of the virus replication process is a cleavage process, in which the virus protease enzyme cuts long precursor polyproteins into mature non-structural proteins, see Fig. 1. If a ligand biomolecule binds to the protease it can prevent and inhibit this cleavage process. A ligand binds to the target protein in a so-called pocket based on various non-covalent interactions like hydrophobic interactions, hydrogen bonding,  $\pi$ -stacking, salt bridges, and amide stacking [14]. With the proper ligand, the protease cleavage process is inhibited, in practice measured by the half maximal inhibitory concentration  $IC_{50}$  corresponding to the inhibitory substance quantity needed to inhibit 50% of the protease process. The protease inhibitor is the target of the drug design process, which we aim to find with evolutionary search.

Computational modeling of protein-ligand binding is a complex process depending on protein-ligand geometry, chemical interactions as well as various constraints and properties like hydration and quantum effects. Complex molecular dynamics computations are often too expensive in computational drug design. Instead, docking tools like AutoDock [26], see Sect. 4, are supposed to be sufficient for a coarse binding affinity estimation based on a simplification of the physical reality.



**Fig. 1.** Illustration of (left) protease enzyme with uncut precursor polyproteins, (middle) the cleavage progress, and (right) protease inhibition preventing the cleavage.

For SARS-CoV-2 the crystal structure of its main protease  $M^{pro}$  is known, e.g., [7, 16, 38]. Various attempts to design inhibitors have been made recently,

e.g., based on known protease inhibitors for other viruses [6, 18], based on virtual screening [13], and computational drug design [25]. It has to be noted, that potential inhibitors discovered by the latter require extensive testing, as they are often completely new molecules. However, computational drug design can form an important starting point in the search of optimal drug candidates.

### 3 Related Work

Methods for de novo drug design can be categorized in different ways [4, 9]. Some works construct molecules directly from atoms [10, 29], while others use chemical fragments as their smallest building block [30]. The goal also varies among publications. Sometimes you want to find drugs that bind to a specific protein binding site like in our work or [30, 37]. Other times the goal is to generate any drug-like molecules as in [10, 31].

ADAPT [30] is a fragment-based method that optimizes for molecules that bind to a specific binding site using a genetic algorithm on an acyclic graph-representation consisting of chemical fragments. The fitness of a resulting compound is evaluated through a docking simulation with a target protein binding site and common drug-likeness indicators. Peptide ligands can be successfully optimized with single-target EAs [20, 32]. The fitness of the individuals was determined experimentally in-vitro. On the other hand, Douguet *et al.* [10] use the SMILES representation and as such work on the level of atoms instead of fragments. In contrast to our work their genetic algorithm optimizes for drug-likeness only instead of binding to a specific ligand. Furthermore, the algorithm is single-objective and simply weighs the different properties in a fitness function using constant coefficients. Similarly, Nigam *et al.* [29] present a genetic algorithm on the SMILES representation for general molecule design. The method increases diversity by using a deep neural network as an adaptive fitness function to penalize long-surviving molecules. In contrast to methods like ours that try to stay inside the distribution of drug-like molecules, the genetic algorithm is free to explore the chemical space in its entirety.

Finally, LigBuilder [37] is a software tool for drug design that is based on a genetic algorithm. It allows optimizing for the interesting quality of binding to multiple targets, which enables tackling more complex diseases with a single drug without the risk of drug-drug interactions that comes with combination drugs (treatment with multiple compounds). Lameijer *et al.* have developed a program called Molecule Evoluator [22]. This program uses an atomic-based evolutionary approach, where the fitness function is an evaluation of the user. A selection of molecules was further investigated experimentally.

Brown *et al.* [5] have utilized an approach for multi-objective optimization of molecules applying a graph-based representation of molecules. The multi-objective evolutionary algorithm applies a Pareto ranking scheme for the optimization process. Wagner *et al.* [35] have developed a tool which identifies potential CNS drugs by means of a multi-objective optimization. The molecules have been optimized for six physical properties. In contrast to the approach presented

here, this tool is not based on evolutionary algorithms but on medical knowledge. A multi-objective evolutionary algorithm for the design of adenosine receptor ligands was developed by van der Horst *et al.* based on a pharmacophore model and three support vector machines [34]. The results have also been verified experimentally. Nicolaou and Brown [28] present a short review, which focuses on the multi-objective optimization of drugs. In this context, different problem definitions and various Multi-objective optimization methods are summarized.

## 4 Molecule Design Metrics

In computational drug design, molecule metrics define the optimization objectives. This section introduces the five metrics our optimization approach is based on. Table 1 shows the value ranges and the optima of the five used metrics. For our experiments we unify these values to a range of  $[0, 1]$ , where 0 is the optimum, as we will describe in Sect. 5.2.

**Table 1.** Value ranges and optimum for used metrics

	Docking score [kcal/mol]	SA	QED	NP	Filters
Value range	$\mathbb{R}$	$[1, 10]$	$[0, 1]$	$[-5, 5]$	$\{0, 1\}$
Optimum	$-\infty$	1	1	5	1

**Binding Affinity Scores.** The major objective in protease inhibitor search is the protein-ligand binding affinity. A widespread tool for this metric is the automated docking tool AutoDock [26], which will also be used by the OpenPandemics<sup>1</sup> activities to fight COVID-19. AutoDock performs very fast calculations of the binding energy by using grid-based look-up tables. For this purpose, the protein is embedded in a grid. The binding energy of all individual atoms of the ligand is calculated at all positions of the grid using semi-empirical force field methods. Using a Lamarckian genetic algorithm, the best binding position and binding energy of the complete ligand can be determined with the help of the look-up tables.

Through various improvements, the accuracy and especially the performance of AutoDock has been significantly improved. In AutoDock Vina [33] a hybrid scoring function based on empirical and knowledge-based data is used instead of the force field method. QuickVina [33] and QuickVina 2 [1] mainly improve the search algorithm by performing the most complex part of the optimization only for very promising ligand positions. We use QuickVina 2 for the calculation of the binding energies of our proposed ligands, as it provides very good results at high performance. For the sake of simplicity, we will use binding affinity score and docking score synonymously.

<sup>1</sup> <https://www.ibm.org/OpenPandemics>.

The informative value of QuickVina 2 binding scores may be limited due to a simplification of various physical properties, such as the neglect of water molecules and the changing electrical properties of ligand and protein when they interact with each other. However, it has been shown by Gaillard [15] that AutoDock Vina binding scores outperform various computational docking methods and Quickvina 2 achieves very comparable results with Autodock Vina [1].

**Synthetic Accessibility (SA).** For drug design it is not only important to find a molecule with the desired properties, but also a synthesizable one. Ertl and Schuffenhauer [12] created a method to estimate the synthetic accessibility of drug-like molecules on a continuous scale and achieve a high agreement with manual estimations by experts. Such a method can easily be incorporated into a search process and we use it as one of our optimization goals, too.

**Quantitative Estimate of Drug-Likeness (QED).** To estimate whether a molecule can be used as a drug, its similarity to other existing drugs can be considered. This is based on the fact that many important physiochemical properties of drugs follow a certain distribution. Lipinski's rule of five [23] which specifies ranges of values for different molecular properties such as size, is frequently used. A major disadvantage, however, is that this rule is only a rule of thumb and only checks whether its criteria are met or not. Among modern drugs there are molecules that violate more than one of Lipinski's rules. A modern approach by Bickerton *et al.* [3] is based on multi-criteria optimization and the principle of desirability. Instead of a fixed value range, all relevant molecular properties are evaluated by an individual desirability function. A single score (QED) is then determined by geometrically averaging all desirability functions. In this work we use this continuous QED score to estimate drug-likeness.

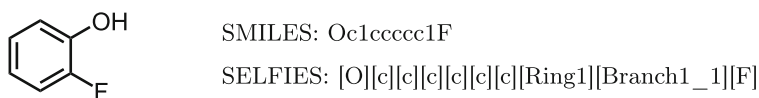
**Natural Product-Likeness (NP).** In addition to the similarity to known drugs, the similarity to naturally occurring biomolecules (natural products) is also an important metric. Natural products have numerous bioactive structures that were created and validated by nature in an evolutionary process. Ertl *et al.* [11] have studied the key differentiating features of natural and synthetic molecules and developed a measure of similarity to natural products. This score is based on structural characteristics of the molecules, such as the number of aromatic rings and the distribution of nitrogen and oxygen atoms.

**Medical Chemical Filters.** Medical chemical filters can be used to exclude molecules that are toxic due to their structural nature. Potentially unstable molecules whose metabolites may be toxic are also not suitable as drugs. We use the MCFs and PAINS filters described by Polykovskiy [31] as a Boolean indicator metric.

## 5 Evolutionary Molecule Search

This section presents the evolutionary approach for the protease inhibitor design. For searching in the design space of biomolecules we use evolutionary algorithms (EAs), which are biologically inspired population-based search heuristics. We employ the evolution strategy oriented ( $\mu + \lambda$ ) population model [2].

A solution is defined by a string based on the self-referencing embedded strings (SELFIES) representation [21], which is an advancement of the simplified molecular-input line-entry system (SMILES) [36] representation. Figure 2 pictures an exemplary molecule with its structural formula and the corresponding SMILES and SELFIES representations. Each string consist of symbols, encoding the occurring atoms, bindings, branches and ring sizes. SELFIES implements a formal grammar, and the interpretation of a symbol depends on derivation rules and state of derivation. In contrast to SMILES, SELFIES strings are always syntactically correct and therefore always yield valid molecules [21].



**Fig. 2.** Molecular structure formula, SMILES, and SELFIES of 2-fluorophenol.

The EA's initial population consists of individuals with randomly generated strings representation of a fixed length. Since multiple SELFIES strings can be translated to the same SMILES string, the resulting SMILES string is compared to a global list of all previously generated individuals. Individuals with a representation that already occurred are discarded and a new individual is generated. This process is repeated until the population consist of unique individuals and also applies for the generation of offspring individuals.

### 5.1 Mutation

Since every SELFIES string corresponds to a valid molecule and every molecule can be expressed in SELFIES representation, the design space can be explored by applying random mutations to the strings – more precisely the SELFIES symbols of which the string is composed. Offspring solutions are created by choosing a random individual from the parental population. Each child is mutated with the following mutation operations with defined probabilities:

**Replacement** is applied independently for every symbol with a probability of  $p_r$ . The symbol is replaced by a random SELFIES symbol.

**Insertion** is applied with probability  $p_i$ . A random symbol is inserted at a random position in the individual's representation.

**Deletion** is applied with probability  $p_d$  and deletes a randomly chosen symbol of the individual's representation.

The new symbols are drawn from a set of symbols inspired by [21]. This set has been extended with benzene as a separate, composed symbol, to increase the likelihood of its occurrence and ease the generation of complex molecules. Additionally, each symbol is assigned a weighting parameter to adjust the probability with which it is randomly selected. This weighting can be used to increase the likelihood of more common symbols (e.g., [C]) in contrast to more complex ones (e.g., branches and ring structures).

## 5.2 Fitness Evaluation

For the selection operator the fitness  $f(\mathbf{x})$  of each solution candidate is evaluated based on the molecule metrics binding affinity score, QED, filters, NP, and SA introduced in Sect. 4. To increase the comparability, each metric is scaled to the range between 0 (best possible score) and 1 (worst possible score). The binding affinities are scaled with regard to the experimentally chosen minimum of  $-15$  kcal/mol and maximum of 1 kcal/mol and clipped to the range between 0 and 1 with soft clipping [19].

For the single-objective baseline experiments each individual is assigned a single composed fitness value. We use a weighted sum fitness of the  $n$  introduced metrics:

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

with weights  $\mathbf{w} = (0.4, 0.15, 0.15, 0.15, 0.15)$  with  $i$  corresponding to 1: docking, 2: SA, 3: QED, 4: NP, and 5: filters. The choice of weights is based on preliminary experiment with the objective of putting the highest attention on the docking score, while at the same time considering the other properties.

The evaluation of individuals of one generation is executed concurrently. During evaluation SELFIES are converted to the SMILES representation. MOSES [31] is then used for the calculation of QED, NP, and SA as well as for the application of the PAINS and MCF filters. The docking score for each compound is determined by QuickVina 2. Therefore, RDKit<sup>2</sup> and MGLTools<sup>3</sup> are used to generate PDB and PDBQT files for the respective SMILES representation. The binding energy is calculated in regards to the COVID-19 M<sup>Pro</sup> (PDB ID: 6LU7 [24])<sup>4</sup> with the search grid being centered around the native ligand position and sized to  $22 \times 24 \times 22 \text{ \AA}^3$ . The exhaustiveness is maintained at its default value of 8, resulting in a execution time of just a few minutes per molecule.

## 5.3 NSGA-II

The objectives presented in Sect. 4 may be contradictory. For example, in preliminary experiments, we discovered that molecules with high AutoDock binding

<sup>2</sup> <https://www.rdkit.org>.

<sup>3</sup> <http://mgltools.scripps.edu>.

<sup>4</sup> PDB: protein data base, <https://www.rcsb.org>.

scores suffer from low QED scores. As the choice of predefined weights for objectives is difficult in advance, a multi-objective approach may be preferable in practice. In our multi-objective optimization setting in molecule space  $\mathcal{M}$  with fitness functions  $f_1, \dots, f_n$  to minimize we seek for a Pareto set  $\{\mathbf{x}^* \mid \nexists \mathbf{x} \in \mathcal{M} : \mathbf{x} \prec \mathbf{x}^*\}$  of non-dominated solutions, where  $\mathbf{x} \prec \mathbf{x}^*$  means  $\mathbf{x}$  dominates  $\mathbf{x}^*$ , i.e.,  $\forall i \in \{1, \dots, n\} : f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ , while  $\exists i \in \{1, \dots, n\} : f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ . NSGA-II [8] is known to be able of approximating a Pareto set with a broad distribution of solutions in objective space, i.e., of the Pareto front. After non-dominated sorting,  $\mu$  non-dominated solutions maximizing the crowding distance. For comparison of different multi-objective runs we also employ the hypervolume indicator (S-metric) measuring the dominated hypervolume in objective space with regards to a dominated reference point [39].

The five metrics described in Sect. 4 form the dimensions of the objective space. Although the fulfilment of Medical Chemical Filters is a binary criterion, it is included as an objective. Since all objectives are computationally determined, they are just an approximation of the real molecule properties. Molecules performing poorly in one of the objectives may still turn out to be potent drug candidates or can potentially lead the algorithm into new areas of the search space.

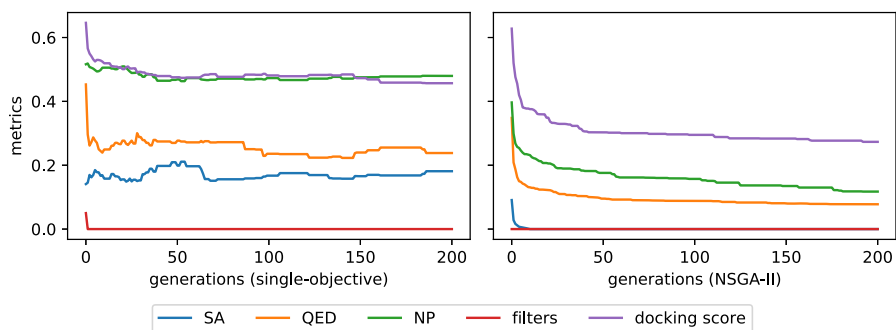
## 6 Experiments

In this section we experimentally analyze the single-objective and the NSGA-II approaches for the protease inhibitor candidate search. For the experimental analyses, the following settings are applied. A (10+100)-EA is used for the single-objective run i.e., in each generation from 10 parents 100 offspring candidate molecules are generated with the mutation operators introduced in Sect. 5.1 with mutation probabilities  $p_r = 0.05$ ,  $p_i = 0.1$ , and  $p_d = 0.1$  applying plus selection. For multi-objective runs the number of parents is increased to 20 to achieve a broader distribution of solutions in objective space. No crossover is applied. Individuals are limited to a length of 80 SELFIES tokens oriented to the setting by Krenn *et al.* [21]. All runs are terminated after 200 generations and are repeated 20 times.

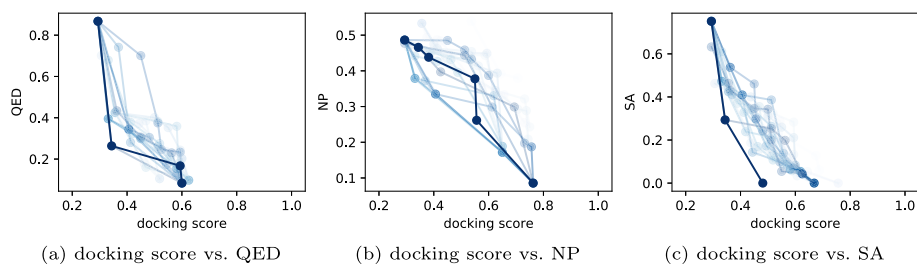
### 6.1 Metric Development

Figure 3 shows the development of the previously explained normalized metrics in single- and multi-objective runs. For the single-objective runs, the best individuals according to fitness are chosen in each generation and their metrics are averaged over all runs. The optimization process concentrates on improving docking score, QED, and NP. As expected, an improvement of one metric may result in a deterioration of another, e.g., as of generation 140, when QED and NP deteriorate in favor of SA and docking score.

For multi-objective runs, the best individuals for each metric are chosen in each generation and then averaged over all runs. A steady improvement with



**Fig. 3.** Development of all metrics during (left) single-objective and (right) multi-objective NSGA-II optimization runs.



**Fig. 4.** Visualization of typical Pareto fronts evolved with NSGA-II: (a) docking score vs. QED, (b) docking score vs. NP, and (c) docking score vs. SA.

regard to all objectives is achieved here, but has to be paid with regard to deteriorations in other objectives that are not shown here.

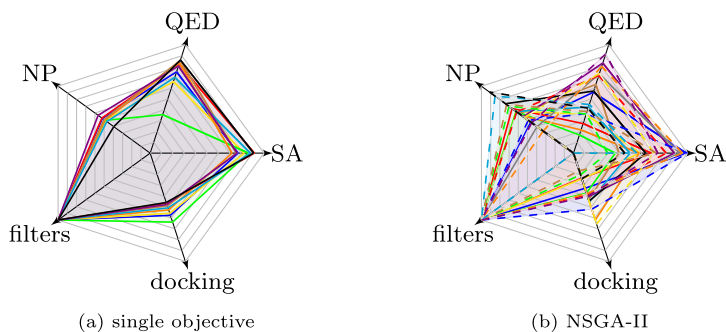
Figure 4 shows three different two-dimensional slices of the Pareto front that compare docking score to QED, NP, and SA. A Pareto front is shown for every 10th generation and their colors start at light blue for the first generation and end at dark blue for the final generation. The plots illustrate NSGA-II's ability to generate solutions with different degrees of balance between docking score and the plotted metric. In the course of the optimization process the front of non-dominated solutions has the expected tendency to move towards the lower left. This is also reflected by the hypervolume indicator, which, in average over all runs improves from  $0.10 \pm 0.03$  in the first to  $0.20 \pm 0.05$  in the last generation. In the slice plots deteriorations are possible due to improvements in the remaining three objectives.

A comparison of final experimental results of the single-objective and NSGA-II runs is presented in Table 2. For NSGA-II the best achieved values for each objective are shown corresponding to the corner points of the Pareto front approximation. For comparison, corresponding metric values are shown for N3 proposed as ligand in the PDB database as well as for Lopinavir, the HIV main protease inhibitor [17]. Docking scores achieved by the single objective



**Table 2.** Experimental results of weighted-sum single-objective approach, the best values per objective for NSGA-II, the N3 ligand (from PDB 6LU7), and Lopinavir (a prominent drug candidate). Statistical evaluation for the NSGA-II method is calculated based on the best 20 individuals per objective. ▼ marks a minimization objective, while ▲ marks a maximization objective.

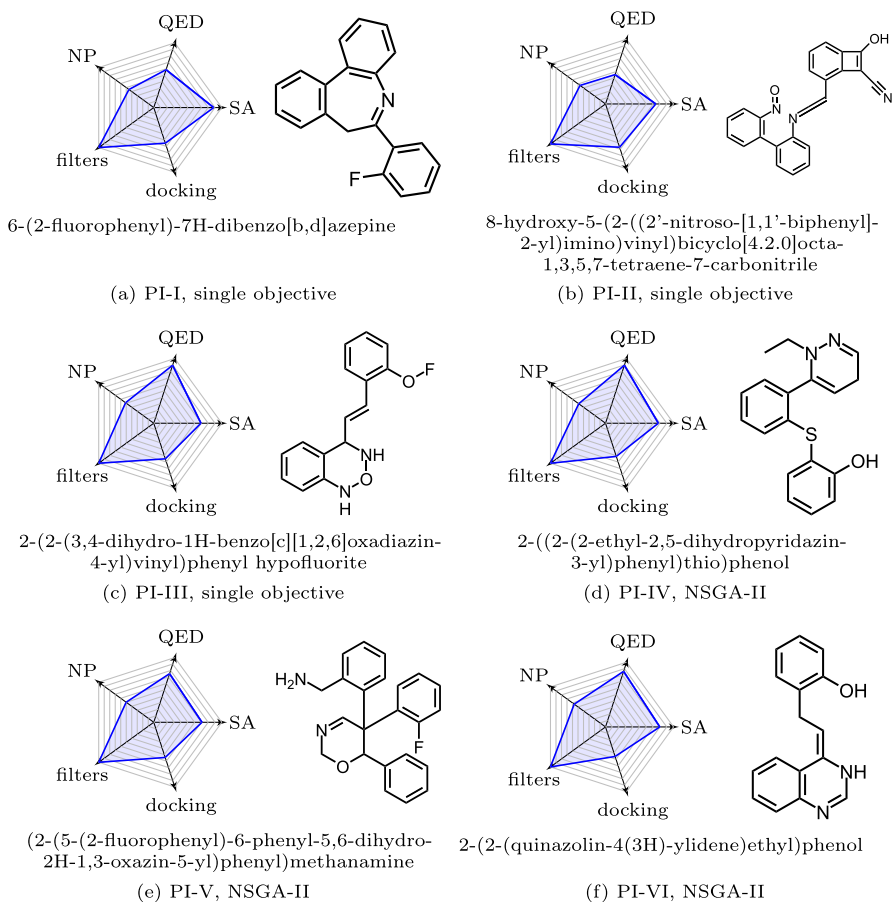
Objective	Single-objective		NSGA-II		N3	Lopinavir
	Best	Avg $\pm$ std	Best	Avg $\pm$ std	Value	Value
Fitness ▼	0.30	$0.32 \pm 0.01$	0.31	$0.39 \pm 0.06$	0.43	0.41
Docking score ▼	-9.30	$-7.68 \pm 0.90$	-13.30	$-10.63 \pm 1.18$	-8.40	-8.40
SA ▼	3.04	$2.63 \pm 0.59$	1.00	$1.00 \pm 0.00$	4.29	3.90
QED ▲	0.66	$0.76 \pm 0.10$	0.94	$0.92 \pm 0.01$	0.12	0.20
NP ▲	0.33	$0.20 \pm 0.54$	4.27	$3.82 \pm 0.24$	-0.18	-0.04
Filters ▲	1.00	$1.00 \pm 0.00$	1.00	$1.00 \pm 0.00$	1.00	1.00



**Fig. 5.** Comparison of population of the last generation of exemplary single-objective (10 molecules) and NSGA-II (20 molecules) runs. Each line represents a molecule candidate.

optimization process show that the best values even overcome the scores of N3 and Lopinavir. Lopinavir and N3 bind similarly strong to  $M^{Pro}$ . NSGA-II achieves promising values for all metrics. The broad coverage of objective function values offers the practitioner a huge variety of interesting candidates. However, some of the extreme metric values may sometimes be unpractical, e.g., the outstanding score of the best NSGA-II molecule (docking score  $-13.3$  kcal/mol) has been achieved by a chemically unrealistic candidate.

From our observations we conclude that the SELFIES representation with our mutation operators are able to robustly achieve molecules of a certain quality. However, we expect the quality of the results to improve with mechanisms that allow the development of larger molecules to overcome fitness plateaus and local optima. Figure 5 compares the populations of the last generation of a typical single-objective and NSGA-II run. The solutions in the single-objective population are similar to each other, while the solutions in the last NSGA-II population maintain a higher diversity of molecule properties.



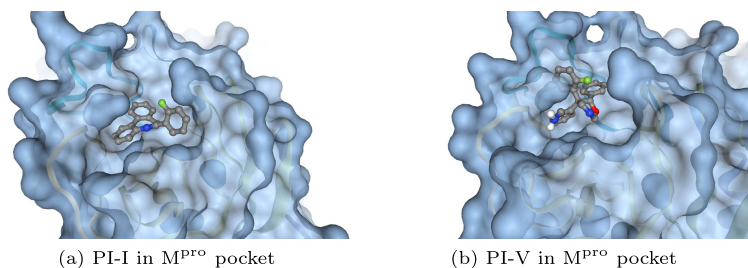
**Fig. 6.** Exemplary protease inhibitors with properties presented as radar plot, structural formula, and chemical name, a-c: single-objective, d-f: NSGA-II results.

## 6.2 Candidate Comparison

In the following we present interesting protein inhibitor candidates evolved with the single- and multi-objective approaches. In our experiments we made three main observations. The molecules generated have a strong tendency to contain aromatic ring structures. Candidates with good drug-likeness are comparatively short. Candidates with high docking scores often have unrealistic geometries.

In Fig. 6 we present a list of six promising protease inhibitors (PI) candidates with properties as radar plots, structural formulas, and chemical names. PI-I (a) to PI-III (c) are results from single-objective runs, while PI-IV (d) to PI-VI (f) show candidates generated by NSGA-II. Points near the border of the radar plot represent better values, e.g., a zero value lies at the corner of a plot. All candidates fulfill the filter condition. PI-1 achieves a high SA value

with a reasonable docking score. PI-II achieves an excellent docking score with  $-9.7$  kcal/mol. PI-III, PI-IV, and PI-VI achieve excellent drug-likeness QED with good docking results around  $-7.0$  kcal/mol. An interesting candidate balancing all objectives is PI-V with docking score  $-7.7$  kcal/mol and QED value of 0.75. Last, we visualize how the ligand candidates are located in the  $M^{\text{Pro}}$  protein pocket optimized by QuickVina 2. Figure 7 shows candidates (a) PI-I and (b) PI-V in their  $M^{\text{Pro}}$  pockets.



**Fig. 7.** Visualization of PI-I and PI-V docked to the pocket of SARS-CoV-2's  $M^{\text{Pro}}$  using NGLview [27].

## 7 Conclusion

In this paper we introduced an evolutionary multi-objective approach to evolve protein inhibitor candidates for the  $M^{\text{Pro}}$  of SARS-CoV-2, which could be a starting point for drug design attempts, aiming at optimizing the QuickVina 2-based protein-ligand binding scores and further important objectives like QED and filter properties. In the experimental part we have shown that the evolutionary processes are able to evolve interesting inhibitor candidates. Many of them achieve promising metrics with ordinary structures, but also unconventional candidates have been evolved that may be worth for a deeper analysis. As the informative value of QuickVina 2 binding scores and also the further metrics may be limited in practice, we understand our approach as AI-assisted virtual screening of the chemical biomolecule space.

Future research will focus on the improvement of protein-ligand models for more detailed and more efficient binding affinity models. Further, we see potential to improve the SELFIES representation in terms of bloated strings that represent comparatively small molecules and mechanisms to guarantee their validity. Moreover, we want to use further multi objective evolutionary algorithms.

**Acknowledgements.** We thank Ahmad Reza Mehdipour, Max Planck Institute of Biophysics, Frankfurt, Germany, for useful comments improving this manuscript. Furthermore, we thank the German Research Foundation (DFG) for supporting our work within the Research Training Group SCARE (GRK 1765/2).

## References

1. Alhossary, A., Handoko, S.D., Mu, Y., Kwok, C.K.: Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics* **31**(13), 2214–2216 (2015). <https://doi.org/10.1093/bioinformatics/btv082>
2. Beyer, H.G., Schwefel, H.P.: Evolution strategies – a comprehensive introduction. *Nat. Comput.* **1**(1), 3–52 (2002). <https://doi.org/10.1023/A:1015059928466>
3. Bickerton, G.R., Paolini, G.V., Besnard, J., Muresan, S., Hopkins, A.L.: Quantifying the chemical beauty of drugs. *Nat. Chem.* **4**(2), 90–98 (2012). <https://doi.org/10.1038/nchem.1243>
4. Brown, N., Fiscato, M., Segler, M.H., Vaucher, A.C.: GuacaMol: benchmarking models for de novo molecular design. *J. Chem. Inf. Model.* **59**(3), 1096–1108 (2019). <https://doi.org/10.1021/acs.jcim.8b00839>
5. Brown, N., McKay, B., Gilardoni, F., Gasteiger, J.: A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. *J. Chem. Inf. Comput. Sci.* **44**(3), 1079–1087 (2004). <https://doi.org/10.1021/ci034290p>
6. Caly, L., Druce, J.D., Catton, M.G., Jans, D.A., Wagstaff, K.M.: The FDA-approved Drug Ivermectin inhibits the replication of SARS-CoV-2 in vitro. *Antiviral Res.* 104787 (2020). <https://doi.org/10.1016/j.antiviral.2020.104787>
7. Dai, W., et al.: Structure-based design of antiviral drug candidates targeting the SARS-CoV-2 main protease. *Science* (2020). <https://doi.org/10.1126/science.abb4489>
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). <https://doi.org/10.1109/4235.996017>
9. Devi, R.V., Sathya, S.S., Coumar, M.S.: Evolutionary algorithms for de novo drug design – a survey. *Appl. Soft Comput.* **27**, 543–552 (2015). <https://doi.org/10.1016/j.asoc.2014.09.042>
10. Douguet, D., Thoreau, E., Grassy, G.: A genetic algorithm for the automated generation of small organic molecules: drug design using an evolutionary algorithm. *J. Comput. Aided Mol. Des.* **14**(5), 449–466 (2000). <https://doi.org/10.1023/A:1008108423895>
11. Ertl, P., Roggo, S., Schuffenhauer, A.: Natural product-likeness score and its application for prioritization of compound libraries. *J. Chem. Inf. Model.* **48**(1), 68–74 (2008). <https://doi.org/10.1021/ci700286x>
12. Ertl, P., Schuffenhauer, A.: Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminform.* **1**(1), 8 (2009). <https://doi.org/10.1186/1758-2946-1-8>
13. Fischer, A., Sellner, M., Naranjan, S., Lill, M.A., Smieško, M.: Inhibitors for Novel Coronavirus Protease Identified by Virtual Screening of 687 Million Compounds (2020). <https://doi.org/10.26434/chemrxiv.11923239.v1>
14. de Freitas, R.F., Schapira, M.: A systematic analysis of atomic protein–ligand interactions in the PDB. *Med. Chem. Commun.* **8**(10), 1970–1981 (2017). <https://doi.org/10.1039/C7MD00381A>
15. Gaillard, T.: Evaluation of AutoDock and AutoDock Vina on the CASF-2013 Benchmark. *J. Chem. Inf. Model.* **58**(8), 1697–1706 (2018). <https://doi.org/10.1021/acs.jcim.8b00312>
16. Jin, Z., et al.: Structure of M pro from COVID-19 virus and discovery of its inhibitors. *Nature* 1–9 (2020). <https://doi.org/10.1038/s41586-020-2223-y>

17. Kaplan, S.S., Hicks, C.B.: Safety and antiviral activity of lopinavir/ritonavir-based therapy in human immunodeficiency virus type 1 (HIV-1) infection. *J. Antimicrob. Chemother.* **56**(2), 273–276 (2005). <https://doi.org/10.1093/jac/dki209>
18. Khaerunnisa, S., Kurniawan, H., Awaluddin, R., Suhartati, S., Soetjipto, S.: Potential Inhibitor of COVID-19 Main Protease (Mpro) From Several Medicinal Plant Compounds by Molecular Docking Study (2020). <https://doi.org/10.20944/preprints202003.0226.v1>
19. Klimek, M.D., Perelstein, M.: Neural network-based approach to phase space integration. [arXiv:1810.11509](https://arxiv.org/abs/1810.11509) [hep-ex, physics:hep-ph, physics:physics, stat] (2018)
20. Krause, T., et al.: Breeding cell penetrating peptides: optimization of cellular uptake by a function-driven evolutionary process. *Bioconjugate Chem.* **29**(12), 4020–4029 (2018). <https://doi.org/10.1021/acs.bioconjchem.8b00583>
21. Krenn, M., Häse, F., Nigam, A., Friederich, P., Aspuru-Guzik, A.: Self-referencing embedded strings (SELFIES): a 100% robust molecular string representation. [arXiv:1905.13741](https://arxiv.org/abs/1905.13741) [physics, physics:quant-ph, stat] (2020)
22. Lameijer, E.W., Kok, J.N., Bäck, T., IJzerman, A.P.: The molecule evaluator. an interactive evolutionary algorithm for the design of drug-like molecules. *J. Chem. Inf. Model.* **46**(2), 545–552 (2006). <https://doi.org/10.1021/ci050369d>
23. Lipinski, C.A., Lombardo, F., Dominy, B.W., Feeney, P.J.: Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Deliv. Rev.* **23**(1), 3–25 (1997). [https://doi.org/10.1016/S0169-409X\(96\)00423-1](https://doi.org/10.1016/S0169-409X(96)00423-1)
24. Liu, X., Zhang, B., Jin, Z., Yang, H., Rao, Z.: 6LU7: The crystal structure of COVID-19 main protease in complex with an inhibitor N3 (2020). <https://www.rcsb.org/structure/6lu7>
25. Macchiagodena, M., Pagliai, M., Procacci, P.: Inhibition of the Main Protease 3CL-pro of the Coronavirus Disease 19 via Structure-Based Ligand Design and Molecular Modeling. [arXiv:2002.09937](https://arxiv.org/abs/2002.09937) [q-bio] (2020)
26. Morris, G.M., et al.: AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility. *J. Comput. Chem.* **30**(16), 2785–2791 (2009). <https://doi.org/10.1002/jcc.21256>
27. Nguyen, H., Case, D.A., Rose, A.S.: NGLview—interactive molecular graphics for Jupyter notebooks. *Bioinformatics* **34**(7), 1241–1242 (2018). <https://doi.org/10.1093/bioinformatics/btx789>
28. Nicolaou, C.A., Brown, N.: Multi-objective optimization methods in drug design. *Drug Discov. Today Technol.* **10**(3), e427–e435 (2013). <https://doi.org/10.1016/j.ddtec.2013.02.001>
29. Nigam, A., Friederich, P., Krenn, M., Aspuru-Guzik, A.: Augmenting genetic algorithms with deep neural networks for exploring the chemical space. [arXiv:1909.11655](https://arxiv.org/abs/1909.11655) [physics] (2020)
30. Pegg, S.C.H., Haresco, J.J., Kuntz, I.D.: A genetic algorithm for structure-based de novo design. *J. Comput. Aided Mol. Des.* **15**(10), 911–933 (2001). <https://doi.org/10.1023/A:1014389729000>
31. Polykovskiy, D., et al.: Molecular sets (MOSES): a benchmarking platform for molecular generation models. [arXiv:1811.12823](https://arxiv.org/abs/1811.12823) [cs, stat] (2019)
32. Röckendorf, N., Borschbach, M., Frey, A.: molecular evolution of peptide ligands with custom-tailored characteristics for targeting of glycostructures. *PLOS Comput. Biol.* **8**(12), e1002800 (2012). <https://doi.org/10.1371/journal.pcbi.1002800>
33. Trott, O., Olson, A.J.: AutoDock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.* **31**(2), 455–461 (2010). <https://doi.org/10.1002/jcc.21334>

34. van der Horst, E., et al.: Multi-objective evolutionary design of adenosine receptor ligands. *J. Chem. Inf. Model.* **52**(7), 1713–1721 (2012). <https://doi.org/10.1021/ci2005115>
35. Wager, T.T., Hou, X., Verhoest, P.R., Villalobos, A.: central nervous system multiparameter optimization desirability: application in drug discovery. *ACS Chem. Neurosci.* **7**(6), 767–775 (2016). <https://doi.org/10.1021/acschemneuro.6b00029>
36. Weininger, D.: SMILES, a chemical language and information system. I. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **28**(1), 31–36 (1988). <https://doi.org/10.1021/ci00057a005>
37. Yuan, Y., Pei, J., Lai, L.: LigBuilder V3: a multi-target de novo drug design approach. *Front. Chem.* **8** (2020). <https://doi.org/10.3389/fchem.2020.00142>
38. Zhang, L., et al.: Crystal structure of SARS-CoV-2 main protease provides a basis for design of improved  $\alpha$ -ketoamide inhibitors. *Science* **368**(6489), 409–412 (2020). <https://doi.org/10.1126/science.abb3405>
39. 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>



# Generic Relative Relations in Hierarchical Gene Expression Data Classification

Marcin Czajkowski<sup>(✉)</sup>, Krzysztof Jurczuk, and Marek Kretowski

Faculty of Computer Science, Bialystok University of Technology,  
Wiejska 45a, 15-351 Bialystok, Poland  
{m.czajkowski,k.jurczuk,m.kretowski}@pb.edu.pl

**Abstract.** Relative Expression Analysis (RXA) plays an important role in biomarker discovery and disease prediction from gene expression profiles. It deliberately ignores raw data values and investigates only the relative ordering relationships between a small group of genes. The classifiers constituted on that concept are therefore robust to small data perturbations and normalization procedures, but above all, they are easy to interpret and analyze.

In this paper, we propose a novel globally induced decision tree in which node splits are based on the RXA methodology. We have extended a simple ordering with a more generic concept that also explores fractional relative relations between the genes. To face up to the newly arisen computational complexity, we have replaced the typical brute force approach with an evolutionary algorithm. As this was not enough, we boosted our solution with the OpenMP parallelization, local search components calculated on the GPU and embedded ranking of genes to improve the evolutionary convergence. This way we managed to explore in a reasonable time a much larger solution space and search for more complex but still comprehensible gene-gene interactions. An empirical investigation carried out on 8 cancer-related datasets shows the potential of the proposed algorithm not only in the context of accuracy improvement but also in finding biologically meaningful patterns.

**Keywords:** Evolutionary data mining · Relative Expression Analysis · Decision trees · Gene Expression Data

## 1 Introduction

Data mining is an umbrella term covering a broad range of tools and techniques for extracting hidden knowledge from large quantities of data. Biomedical data can be very challenging due to the enormous dimensionality, biological and experimental noise as well as other perturbations. Unfortunately, many traditional machine learning algorithms use complex predictive models, which impede biological understanding and are an obstacle for mature applications [1]. Most of the research effort tends to focus almost exclusively on the prediction accuracy of core data mining tasks (e.g., classification and regression), and far less

effort has gone into understand and interpret the discovered knowledge. It is not enough to simply produce good outcomes but to provide logical reasoning just as clinicians do for medical treatments.

There is a strong need for ‘white box’ computational methods to effectively and efficiently carry out the predictions using biomedical data. One of the example approaches which may actually help in understanding and identifying relationships between specific features and improve biomarker discovery is the Relative Expression Analysis (RXA) [9]. It is a powerful collection of easily interpretable algorithms that plays an important role in genomic data classification [11]. RXA’s key novelty is the use of interactions between a small collection of genes by examining the relative order of their expressions rather than their raw values. The influence of RXA solutions could be even greater, however, the simplicity of model decisions which is based only on the plain ordering comparisons strongly limits the search for other gene-gene relations. Additionally, a typical exhaustive search performed by most of RXA solutions limits the number of genes that can be analyzed [16] due to computational complexity.

In this paper, we introduce a new approach for RXA called Evolutionary Relative Expression Decision Tree (Evo-REDT). We have extended the simple ordering relations between the genes proposed in RXA with a new more generic concept. It explores relative fraction comparison in the gene pairs, therefore, it can identify percent changes in their relations between different expression profiles. To include also the hierarchical relations between the gene pairs, we have adapted an evolutionary induced decision tree system called Global Decision Tree (GDT) [15]. It allows performing a simultaneous search for the tests in the internal nodes as well as the overall tree structure. In each splitting node of a tree, we use a test consisting of two genes and a fraction which represents the ratio (weight) of their relations. Originally, the selection of a top pair in RXA performs an exhaustive search for all possible order relations between two genes. Using brute force within the proposed approach is computationally infeasible, on the other hand, relying only on the evolutionary search may result in a very slow algorithm convergence. Therefore, we have proposed several improvements in order to boost our solution, mainly:

- several specialized variants of mutation and crossover operators;
- local search components calculated on the GPU;
- embedded ranking of genes in order to consider the relations based on top genes more often;
- parallel processing of the individuals of the population using shared memory (OpenMP) paradigm.

Our main objective is to find in a reasonable time more advanced relations in comparison to RXA that are more accurate and still easy to understand and interpret.



## 2 Background

Genomic data is still challenging for computational tools and mathematical modeling due to the high ratio of features to observations as well as enormous gene redundancy and ubiquitous noise. Nearly all off-the-shelf techniques applied to genomics data [1], such as neural networks, random forests and SVMs are ‘black box’ solutions which often involve nonlinear functions of hundreds or thousands of genes and complex prediction models. Currently, deep learning approaches have been getting attention as they can better recognize complex features through representation learning with multiple layers. However, we know very little about how such results are derived internally. In this section, we focus on two concepts which are the main elements of the proposed approach.

### 2.1 RXA Classification Algorithms

Relative Expression Analysis focuses on finding interactions among a small group of genes and studies the relative ordering of their expression values. In the pioneer research [10], authors used ranks of genes instead of their raw values and introduced the Top Scoring Pair (TSP) classifier. It is a straightforward prediction rule that makes a pairwise comparison of gene expression values and searches for a single pair of genes with the highest rank. Let  $x_i$  and  $x_j$  ( $0 \leq i, j < N$ ) be the expression values of two different genes from available set of genes and there are only two classes: *normal* and *cancer*. First, the algorithm calculates the probability of the relation  $x_i < x_j$  between those two genes in the objects from the same class:

$$P_{ij}(\textit{normal}) = \textit{Prob}(x_i < x_j | Y = \textit{normal}) \quad (1)$$

and

$$P_{ij}(\textit{cancer}) = \textit{Prob}(x_i < x_j | Y = \textit{cancer}), \quad (2)$$

where  $Y$  denotes the class of the objects. Next, the score for this pair of genes  $(x_i, x_j)$  is calculated:

$$\Delta_{ij} = |P_{ij}(\textit{normal}) - P_{ij}(\textit{cancer})|. \quad (3)$$

This procedure is repeated for all distinct pairs of genes and the pair with the highest score becomes the top-scoring pair. In the case of a draw, a secondary ranking that relies on gene expression differences is used [19]. Finally, for a new test sample, the relation between expression values of the top pair of genes is checked. If the relation holds, then the TSP predictor votes for the class that has higher probability  $P_{ij}$  in the training set, otherwise it votes for the class with smaller probability.

There are many extensions of the TSP classifier. The main ones focused on increasing the number of gene pairs in the predictive model (k-TSP [19]) or analyzing the order of relationships for more than two genes (TSN [16]). Those methods were also combined with a typical decision tree algorithm (TSPDT [3])

in which each non-terminal node of the tree divides instances according to a splitting rule that is based on TSP or k-TSP accuracy. As one of the main drawbacks of the aforementioned solutions was the enormous computational complexity resulting from the exhaustive search, various optimization techniques were proposed. Some of them were based on parallel computing using GPGPU [16], others used the heuristic approach involving evolutionary algorithms (EA) like EvoTSP [4]. Finally, there are many variations of ranking and grouping the gene pairs [9, 13] but all the systems inherited the standard RXA methodology based on the ordering relations.

## 2.2 Decision Trees

Decision trees have a knowledge representation structure made up of nodes and branches, where: each internal node is associated with a test on one or more attributes; each branch represents the test outcome, and each leaf (terminal node) is designed by a class label. Induction of optimal DT for a given dataset is a known NP-complete problem. As a consequence, practical DT learning algorithms must be heuristically enhanced. The most popular type of tree induction is based on a top-down greedy search [14]. It starts from the root node, where the locally optimal split (test) is searched according to the given optimality measure. Next, the training instances are redirected to the newly created nodes, and this process is repeated for each node until a stopping condition is met. Inducing the DT through a greedy strategy is fast and generally efficient in many practical problems, but it usually produces overgrown solutions.

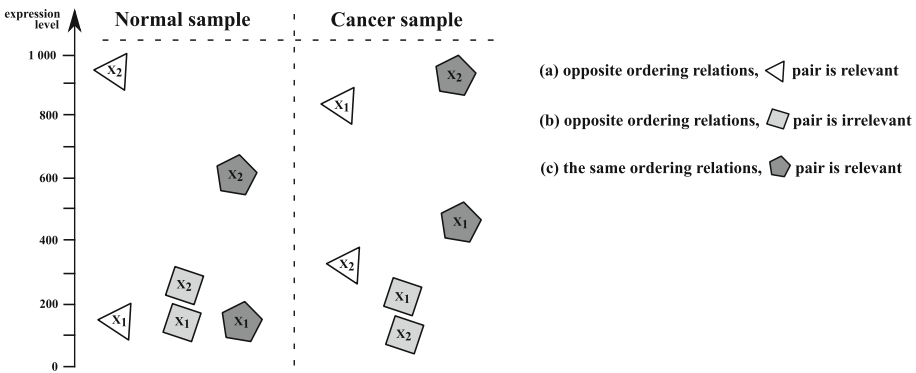
Evolutionary induction of decision trees is an alternative to greedy top-down approaches as it mitigates some of the negative effects of locally optimal decisions [15]. The strength of such an approach lies in a global search for the tree structure and the tests in the internal nodes. This global induction is much more computationally complex; however, it can reveal hidden regularities that are often undetectable by greedy methods. Unfortunately, there are not so many new solutions in the literature that focus on the classification of genomic data with comprehensive DT models. In the literature, there is far more interest in trees as sub-learners of an ensemble learning approach, such as Random Forests. These solutions alleviate the problem of low accuracy by averaging or adaptive merging of multiple trees. However, when modeling is aimed at understanding basic processes, such methods are not so useful due to the complexity of the generated rules.

## 2.3 Motivation

RXA solutions deliberately replace the raw expression data values with simple ordering relationships between the features. However, in a nutshell, limiting knowledge to the information that expression of one gene  $x_i$  is larger than another  $x_j$  which has a form of a pair:  $(x_i > x_j)$  may result in a large loss of potentially important data. We propose an additional fractional component

called relational weight  $w$ , which is the ratio of the genes relation in a pair:  $(x_i > w * x_j)$ .

Let us hypothetically assume that the two genes  $x_1$  and  $x_2$  have constant expression values among the instances from the same classes. Figure 1 shows three simple scenarios (a), (b), (c) of possible relations between genes  $x_1$  and  $x_2$  in a normal and cancer class. The RXA algorithms will detect only the pairs  $(x_1, x_2)$  from the (a) and (b) scenario as “top pairs” because only there the relation between genes changes between classes. However, the pair from the scenario (b) should not be considered as a biological switch due to small change of the genes expression level between classes. Unfortunately, the undoubtedly relevant pair from the scenario (c) will not be considered by any currently available RXA-family algorithms despite significant variations in the expression values of genes in normal and cancer classes. It might choose them together with other genes, by making multiple top pairs, but besides potential interpretability problems, lower accuracy issues may also arise. Evo-REDT solution is capable not only of selecting relevant pairs (scenario (a) and (c)) but also ignoring the ones with small weight perturbations.



**Fig. 1.** Possible relations between two genes X1 and X2 in normal and cancer sample together with biological importance of the pair constituted from that genes

Additionally, RXA enormous computational complexity strongly limits the number of features and inter-relations that can be analyzed [13]. For regular RXA exhaustive search, it equals  $O(T * M * N^2)$ , where T is the number of splitting nodes of DT, M is the number of instances and N is the number of analyzed genes. Evo-REDT has much higher complexity due to additional search for the relations weight. For this newly arisen level of complexity, even a standard evolutionary approach might be not sufficient.

### 3 Evolutionary Relative Expression Decision Tree

The proposed solution has been integrated into a system called the Global Decision Tree (GDT). Its overall structure is based on a typical evolutionary

algorithm (EA) schema [17] with an unstructured population and generational selection. The GDT framework [15] can be used to induce various types of trees and its applications also cover biomedical data [6]. We have proposed several changes in the original GDT solutions, involving the node representation and overall evolutionary search. The general flowchart of the Evo-REDT solution is illustrated in Fig. 2.

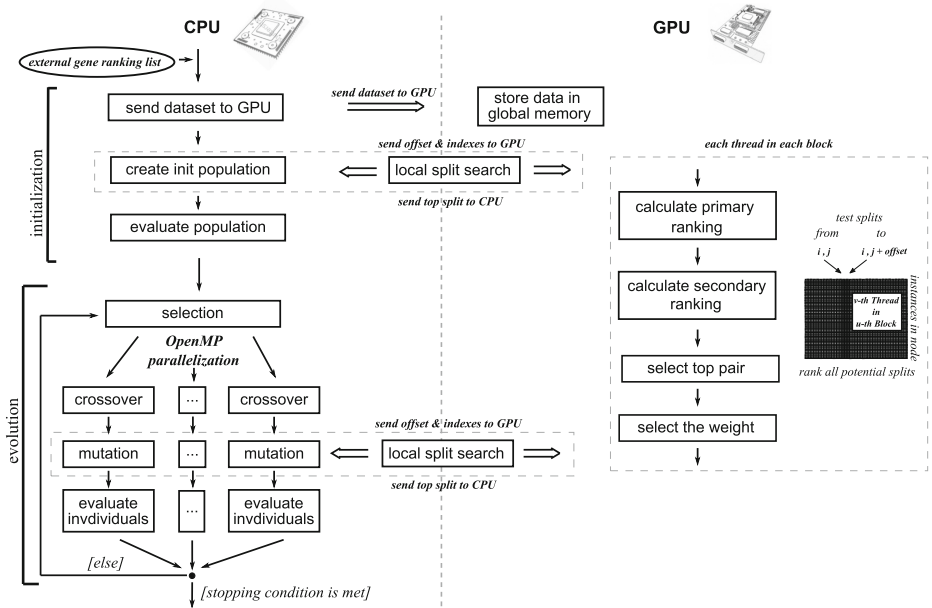


Fig. 2. General flowchart of the Evo-REDT solution

### 3.1 Representation, Initialization, Selection

Decision trees are quite complicated structures, in which a number of nodes, type of the tests and even number of test outcomes are not known in advance. The GDT system uses a tree-encoding schema in which individuals are represented in their actual form as potential tree-solutions. A new type of tests in the splitting nodes is applied. It is constituted from a single pair of genes together with the weight and has the form  $(x_i > w * x_j)$ . Additionally, each node stores information about training instances related to the node. This allows the algorithm to perform more effectively local modifications of the structure and tests during the application of genetic operators. Finally, we have embedded information about the discriminative power of genes calculated by the external tool (algorithm Relief-F was used [18]) in a form of ranked list. It is submitted as an

additional input to Evo-REDT and can be manually modified, for example, to focus on biomarker genes for a given disease.

In the GDT system, to maintain a balance between exploration and exploitation, initial individuals are created by using a simple top-down algorithm with randomly selected sub-samples of original training data. Before initialization, the dataset is first copied from the CPU main memory to the GPU device memory so each thread block can access it (see Fig. 2). It is performed only once before starting the tree induction as later only the indexes of the instances that are located in a calculated node are sent.

The selection mechanism is based on a ranking linear selection [17] with the elitist strategy, which copies the best individual founded so far to the next population. Evolution terminates when the fitness of the best individual in the population does not improve during a fixed number of generations (default: 100) or a maximum number of generations is reached (default: 1000).

### 3.2 Genetic Operators

To preserve genetic diversity, the GDT system applies two specialized genetic meta-operators corresponding to the classical mutation and crossover. Both operators may have a two-level influence on the individuals as either decision tree structure or a test in the splitting node can be modified. Depending on the position in the tree, different aspects are taken into account to determine the crossover or mutation point. If the change considers the overall structure, the level of the tree is taken into account. The modification of the top levels is performed less frequently than the bottom parts as the change would have a much bigger, global impact. The probability of selection is proportional to the rank in a linear manner. Examples of such variants are adding/deleting a node in the case of mutation and tree-branch crossover.

If the change considers the tests in the splitting nodes their quality is taken into account like the ones with the higher error, per instance, are more likely to be changed. In the case of mutation, it can be replacing a pair of genes with a new one or changing a single gene in a test. The first two variants require updating the weight between two genes that constitute a test. Additionally, in both variants, we use the gene ranking that determines which new genes will appear in the test. This way top genes from the dataset are considered more often in the population. Crossover variants allow whole tests to exchange as well as randomly selected genes from the pairs between the individuals.

### 3.3 Fitness Function

DTs are at some extent prone to overfitting [14]. In typical top-down induction, this problem is partially mitigated by performing a stop condition and applying post-pruning. In the case of evolutionary induced DT, this problem may be controlled by a multi-objective fitness function in order to maximize the accuracy and minimize the complexity of the output tree. In this work, we decided to use

a simple weight formula, but measure the tree complexity in a different way. The Evo-REDT system maximizes the following fitness function:

$$Fitness(T) = Q(T) - \alpha * Rank(T), \quad (4)$$

where:  $Q(T)$  is the accuracy calculated on the training set,  $Rank(T)$  is the sum of the ranks of attributes constituting tests and  $\alpha$  is the relative importance of the complexity term (default value is 0.05) and a user supplied parameter. As we can see, instead of using the number of leaves or nodes, we measure the sum of the ranks of the attributes that constitute the tests in the internal nodes provided by the external Relief-F algorithm. This way the attributes with the higher rank are more likely to be used in the prediction model.

### 3.4 Parallelization

The GDT system supports various parallelization techniques [5,15]. However, in the context of biomedical data mining where the number of instances is low, using only the data-parallel decomposition strategy will not be effective [12]. We propose a hybrid approach with shared address space (OpenMP) paradigm and graphics processing units (GPU)-based parallelization. The individuals from the population are spread over the CPU cores using OpenMP threads. Each OpenMP thread is responsible for subsequent algorithm blocks (genetic operator, evaluation, etc.) for the assigned pool of individuals. This way, the individual are processed in parallel on the CPU.

The GPU parallelization is applied in a different way. When the mutation operator updates or calculates a new test in a splitting node, a local search for the top gene pair is performed. Each thread on the device is assigned an equal amount of relations (called offset) to compute so it ‘knows’ which relations of genes it should analyze and where it should store the result. However, finding a relation  $x_i > w * x_j$  for a given set of instances that reached a particular node is still computationally demanding. That is why the first attribute is selected by the CPU which together with offset and indexes to the instances are sent to the GPU. Each thread in each block calculates the primary ranking which involves the number of times the relation holds in one of the classes and not in another one. The secondary ranking is a draw breaker, which is based on the differences in the weight relations in each class and object. The weight  $w$  of the top pair equals to  $x_i/x_j$  of the instance in which relation simultaneously distinguishes the instances from different classes and is the lowest among the instances from the same class. The weight can also be smoothed to e.g. a single precision value or even rounded to an integer in order to improve comprehensibility and at some extent the overall generalization (default: 0.5). After all block threads finished, the results are copied from the GPU device memory back to the CPU main memory and sorted according to the rank. Simplified ranking linear selection is used to select the pair of genes that will constitute the test in the splitting node.

## 4 Experimental Validation

Experimental analysis to evaluate the relative performance of the proposed approach is performed using several cancer-related gene expression datasets. We confront the Evo-REDT with popular RXA extensions as well as outline other algorithm characteristics.

### 4.1 Inducers, Datasets and Settings

To make a proper comparison with the RXA algorithms, we use the same 8 cancer-related benchmark datasets that were tested with the EvoTSP solution [4]. Datasets are deposited in NCBI's Gene Expression Omnibus and summarized in Table 1. A typical 10-fold cross-validation is applied and following RXA algorithms are confronted:

- TSP, TST, and k-TSP were calculated with the AUERA software [8];
- EvoTSP results were taken from the publication [4];
- original TSPDT and Evo-REDT implementations are used.

**Table 1.** Details of gene expression datasets: abbreviation with name, number of genes and number of instances.

Datasets	Genes	Instances	Datasets	Genes	Instances
(a) GDS2771	22215	192	(e) GSE10072	22284	107
(b) GSE17920	54676	130	(f) GSE19804	54613	120
(c) GSE25837	18631	93	(g) GSE27272	24526	183
(d) GSE3365	22284	127	(h) GSE6613	22284	105

In all experiments, a default set of parameters for all algorithms is used in all tested datasets and the presented results correspond to averages of several runs. Evo-REDT uses recommended GDT settings that were experimentally evaluated and given in details in GDT framework description [15], e.g.: population size: 50, mutation rate 80%, crossover rate 20%.

Due to the performance reasons concerning other approaches, the Relief-F feature selection was applied and the number of selected genes was arbitrarily limited to the top 1000. Experiments run on the workstation equipped with Intel Core i5-8400 CPU, 32 GB RAM, and NVIDIA GeForce GTX 1080 GPU card (8 GB memory, 2 560 CUDA cores). The sequential algorithm was implemented in C++ and the GPU-based parallelization part was implemented in CUDA-C (compiled by nvcc CUDA 10; single-precision arithmetic was applied).

**Table 2.** Inducers accuracy and size comparison, best for each dataset is bolded

Dataset	TSP	TST	k-TSP		EvoTSP		TSPDT		Evo-REDT	
	Acc.	Acc.	Acc.	Size	Acc.	Size	Acc.	Size	Acc.	Size
(a)	57.2	61.9	62.9	10	65.6	4.0	60.1	15.4	<b>72.9</b> $\pm$ 8.0	8.2 $\pm$ 1.1
(b)	88.7	89.4	90.1	6.0	96.5	2.1	<b>98.2</b>	1.0	<b>98.2</b> $\pm$ 5.7	2.2 $\pm$ 0.4
(c)	64.9	63.7	67.2	10	<b>78.1</b>	2.8	72.3	5.8	76.2 $\pm$ 9.9	7.3 $\pm$ 1.4
(d)	93.5	92.8	94.1	10	<b>96.2</b>	2.1	88.3	2.0	94.2 $\pm$ 8.8	2.8 $\pm$ 0.9
(e)	56.0	60.5	58.4	14	66.9	3.1	68.1	4.7	<b>73.0</b> $\pm$ 10.9	6.0 $\pm$ 0.8
(f)	47.3	50.1	56.2	18	66.2	2.7	67.2	10.9	<b>74.3</b> $\pm$ 6.2	7.9 $\pm$ 1.0
(g)	81.9	84.2	87.2	14	86.1	4.1	88.6	3.3	<b>91.5</b> $\pm$ 8.5	3.9 $\pm$ 0.7
(h)	49.5	51.7	55.8	10	53.6	6.1	59.6	7.0	<b>70.5</b> $\pm$ 16.9	8.4 $\pm$ 1.0
Average	67.4	69.3	71.5	11.5	76.2	2.7	75.3	6.2	<b>81.3</b> $\pm$ 9.4	5.8 $\pm$ 0.9

## 4.2 Accuracy Comparison of Evo-REDT to Popular RXA Counterparts

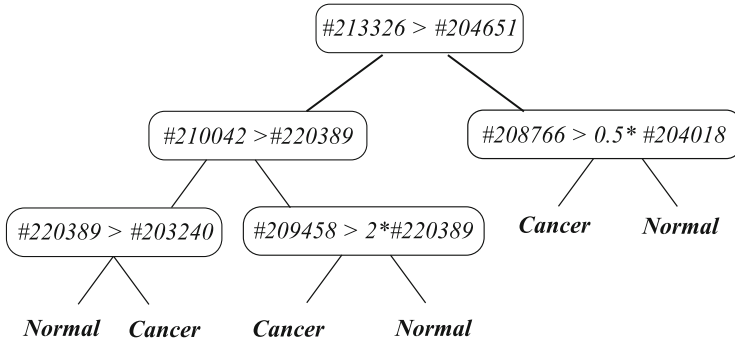
Table 2 summarizes classification performance for the proposed solution and its competitors. The model size of TSP and TST is not shown as it is fixed and equals correspondingly 2 and 3. Both, the evolutionary TSP approach called EvoTSP, as well as a top-down induced RXA decision tree TSPDT, are outperformed by the proposed Evo-REDT solution. The statistical analysis of the obtained results using the Friedman test and the corresponding Dunn’s multiple comparison test (significance level/p-value equals 0.05), as recommended by Demsar [7] showed that the differences in accuracy are significant. We have also performed an additional comparison between the datasets with the corrected paired t-test with the significance level equals 0.05 and 9 degrees of freedom (n-1 degrees of freedom where n = 10 folds). It showed that Evo-REDT significantly outperforms all algorithms on more than half datasets. What is important, the trees induced by the Evo-REDT are not only accurate but also relatively small and simple. This indicates that the model managed to find more deep interaction and sub-interaction between the genes.

## 4.3 Evo-REDT Characteristics

To improve the overall generalization of Evo-REDT as well as the model comprehensibility, we have checked how rounding the weight relation between the genes impacts the results. Experimental results showed that there were no statistical differences between algorithms with 0.1, 0.5 respectively, and without rounding weights. Therefore, in Evo-REDT we used a default 0.5 rounding for the weight relation. An example of tree induced for the first dataset (GDS2771) is illustrated in Fig. 3. We can observe, that Evo-REDT found splitting pairs with various weights and the induced tree is small and easily interpretable.

In this section, we would also like to share some of the preliminary results to verify if the trees induced by the Evo-REDT are somehow useful. By using





**Fig. 3.** An example decision tree induced by Evo-REDT with rounded to 0.5 weights for lung cancer data (GDS2771)

the GDS2771 dataset description available on GenBank NCBI [2] we performed a brief examination of our predictor (see Fig. 3). To check if genes found in the splitting nodes have some biological meaning we have decoded gene names from GDS2771 with GPL96 platform provided by NCBI (in the Figure genes are encoded as Affymetrix Probe Set ID). We found out that 2 out of 9 genes are directly related to lung cancer, another 2 were discussed in several papers while the remaining 5 were also visible in the medical literature. This is only an example of a fraction of knowledge discovered by Evo-REDT but even the presented model is at some point supported by biological evidence in the literature.

Much effort in this paper was put into improving the speed of the proposed solution. Table 3 shows the average calculation time for a single dataset without any parallel calculations and with OpenMP and/or GPU enabled. We also include the approximate induction time of other algorithms (if provided) for illustration purposes only. We cannot compare the results as the machines, software, etc. may be significantly different. However, with additional embedded feature ranking we managed to improve the EA convergence and reduce the number of required iterations which equals 1000 whereas for EvoTSP it is 10 times higher.

As expected, the sequential version of the algorithm is much slower than the rest of the Evo-REDT variants from Table 3. It should be noted that GPU-accelerated Evo-REDT may be applied to much larger gene expression datasets without any feature selection. The potential of the GPU parallelization was not fully utilized within performed experiments due to the limited number of features.

**Table 3.** Average time in seconds for the algorithm to train a model

Algorithm	Evo-REDT			TSP	TST	TSPDT	EvoTSP
	Seq.	OpenMP	OpenMP+GPU				
Time	637	171	110	2.1	712	152	2700

## 5 Conclusions

Finding simple decision rules with relatively high prediction power is still a major problem in biomedical data mining. Our new approach called Evo-REDT tackles this problem with a more generic approach of finding fractional relative relations between the genes. The proposed solution is composed of evolutionary DT inducer and extended concept of RXA. Our implementation covers multiple optimizations including OpenMP and GPU parallelizations as well as incorporates knowledge about the discriminative power of genes into the evolutionary search. Performed experiments show that the knowledge discovered by Evo-REDT is accurate, comprehensible and the model training time is relatively short.

We see many promising directions for future research. In particular, we are currently working with biologists and bioinformaticians to better understand the gene relations generated by Evo-REDT. Next, there is still a lot of ways to extend the tree representation e.g. by using more than one pair of genes in the splitting nodes. Optimization of the approach can also be improved e.g. load-balancing of tasks based on the number of instances in each node, simultaneous analysis of two branches, better GPU hierarchical memory exploitation. Finally, we want to validate our approach using proteomic and metabolomic data as well as integrated multi-omics datasets.

**Acknowledgments.** This project was funded by the Polish National Science Center and allocated on the basis of decision 2019/33/B/ST6/02386 (first author). The second and third author were supported by the grant WZ/WI-IIT/3/2020 from BUT founded by Polish Ministry of Science and Higher Education.

## References

1. Bacardit, J., et al.: Hard data analytics problems make for better data analysis algorithms: bioinformatics as an example. *Big Data* **2**(3), 164–176 (2014)
2. Benson, D.A., et al.: GenBank. *Nucleic Acids Res.* **46**(D1), D41–D47 (2018)
3. Czajkowski, M., Kretowski, M.: Top scoring pair decision tree for gene expression data analysis. *Adv. Exp. Med. Biol.* **696**, 27–35 (2011)
4. Czajkowski, M., Kretowski, M.: Evolutionary approach for relative gene expression algorithms. *Sci. World J.* 593503 (2014). Hindawi
5. Czajkowski, M., Jurczuk, K., Kretowski, M.: A parallel approach for evolutionary induced decision trees. MPI+OpenMP implementation. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2015*. LNCS (LNAI), vol. 9119, pp. 340–349. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-19324-3\\_31](https://doi.org/10.1007/978-3-319-19324-3_31)
6. Czajkowski, M., Kretowski, M.: Decision tree underfitting in mining of gene expression data. An evolutionary multi-test tree approach. *Expert Syst. Appl.* **137**, 392–404 (2019)
7. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**, 1–30 (2006)
8. Earls, J.C., et al.: AUREA: an open-source software system for accurate and user-friendly identification of relative expression molecular signatures. *BMC Bioinform.* **14**, 78 (2013)

9. Eddy, J.A., Sung, J., Geman, D., Price, N.D.: Relative expression analysis for molecular cancer diagnosis and prognosis. *Technol. Cancer Res. Treat.* **9**(2), 149–159 (2010)
10. Geman, D., et al.: Classifying gene expression profiles from pairwise mRNA comparisons. *Stat. Appl. Genet. Mol. Biol.* **3**(19) (2004)
11. Huang, X., et al.: Analyzing omics data by pair-wise feature evaluation with horizontal and vertical comparisons. *J. Pharm. Biomed. Anal.* **157**, 20–26 (2018)
12. Jurczuk, K., Czajkowski, M., Kretowski, M.: Evolutionary induction of a decision tree for large scale data. A GPU-based approach. *Soft Comput.* **21**, 7363–7379 (2017)
13. Kagaris, D., Khamesipour, A.: AUCTSP: an improved biomarker gene pair class predictor. *BMC Bioinform.* **19**(244) (2018)
14. Kotsiantis, S.B.: Decision trees: a recent overview. *Artif. Intell. Rev.* **39**(4), 261–283 (2013)
15. Kretowski, M.: Evolutionary Decision Trees in Large-Scale Data Mining. *Studies in Big Data*, vol. 59. Springer, Heidelberg (2019). <https://doi.org/10.1007/978-3-030-21851-5>
16. Magis, A.T., Price, N.D.: The top-scoring ‘N’ algorithm: a generalized relative expression classification method from small numbers of biomolecules. *BMC Bioinform.* **13**(1), 227 (2012)
17. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996). <https://doi.org/10.1007/978-3-662-03315-9>
18. Robnik-Šikonja, M., Kononenko, I.: Theoretical and empirical analysis of ReliefF and RReliefF. *Mach. Learn.* **53**(1–2), 23–69 (2003)
19. Tan, A.C., Naiman, D.Q.: Simple decision rules for classifying human cancers from gene expression profiles. *Bioinformatics* **21**, 3896–3904 (2005)



# A Variable Neighborhood Search for the Job Sequencing with One Common and Multiple Secondary Resources Problem

Thomas Kaufmann, Matthias Horn<sup>(✉)</sup>, and Günther R. Raidl

Institute of Logic and Computation, TU Wien, Vienna, Austria  
thomas@tkaufmann.at, {horn,raidl}@ac.tuwien.ac.at

**Abstract.** In this work we consider a scheduling problem where a set of non-preemptive jobs needs to be scheduled such that the makespan is minimized. Each job requires two resources: (1) a common resource, shared by all jobs and (2) a secondary resource, shared with only a subset of the other jobs. The secondary resource is required during the job's entire processing time whereas the common resource is only required during a part of a job's execution. The problem models, for instance, the scheduling of patients during one day in a particle therapy facility for cancer treatment. We heuristically tackle the problem by a general variable neighborhood search (GVNS) based on move and exchange neighborhoods and an efficient evaluation scheme to scan the neighborhoods of the current incumbent solution. An experimental evaluation on two benchmark instance sets, including instances with up to 2000 jobs, shows the effectiveness of the GVNS. In particular for larger instances our GVNS outperforms an anytime A\* algorithm that was the so far leading method in heuristic terms as well as a constrained programming model solved by ILOG CP optimizer.

**Keywords:** Sequencing · Scheduling · Variable neighborhood search · Particle therapy patient scheduling

## 1 Introduction

In this work we apply a *general variable neighborhood search* (GVNS) approach to the *job sequencing with one common and multiple secondary resources* (JSOCMSR) problem. The JSOCMSR has been introduced in [8] and considers a scenario where a finite set of jobs must be scheduled without preemption. Each job requires two resources: (1) a common resource, which is shared by all jobs and (2) a secondary resource which is shared by only a subset of the jobs. The secondary resource is required for the entire processing time of a job whereas

---

We gratefully acknowledge the financial support of the Doctoral Program “Vienna Graduate School on Computational Optimization” funded by Austrian Science Foundation under Project No W1260-N35.

the common resource is needed only after some pre-processing time for a part of the job's whole processing time. The objective is to minimize the makespan.

The JSOCMSR problem has applications, for example, in the context of the production of certain goods where on a single machine (the common resource, for example an oven used for heat treatment) some fixtures or molds (the secondary resource) filled with some raw material are sequentially processed. Before the fixtures/molds can be processed on the machine there is a setup time during which the secondary resource is already needed (e.g., preparations within the mold) as well as a post-processing time also still requiring the secondary resource (e.g., cooling before the product can be removed from the mold).

Another more specific application is the scheduling of treatments for cancer patients who are to receive a particle therapy [1, 9, 13]. In this rather novel treatment technique, carbon or proton particles are accelerated in a particle accelerator to almost the speed of light, and this particle beam is directed into one of a few treatment rooms where a patient gets radiated. There are typically two to four treatment rooms that are differently equipped for specific kinds of radiations. In this scenario the JSOCMSR appears as a simplified daily subproblem where the treatment rooms correspond to the secondary resources and the single particle beam, which can only be directed into one of these rooms at a time, corresponds to the common resource. The treatment room for each patient is known in advance and depends on the patients specific needs. Each patient treatment requires a specific preparation time (positioning, fixation, sedation, etc.) in the room before the radiation can be performed and occupies the room after the treatment for some further medical examinations until the patient can eventually leave the room. The JSOCMSR we consider here only represents the "hard core" of the real practical scheduling problem, in which several different objectives, further resources, time windows, and other soft- and hard-constraints need to be taken care of. Maschler et al. [13] tackled this real-world problem with a greedy construction method, which is extended to an iterative greedy metaheuristic and a greedy randomized adaptive search (GRASP).

For the JSOCMSR, Horn et al. [5] proposed an exact anytime  $A^*$  search. For instances, where the workload over all secondary resources is rather balanced this  $A^*$  search works extremely well by solving even large instances with up to 2000 jobs to proven optimality. However, on instances where the workload over the secondary resources is skewed, i.e., one resource is more frequently required than the others, the  $A^*$  algorithm's performance degrades and it is in many cases only able to provide heuristic solutions.

*Contribution of This Work.* For such hard-to-solve JSOCMSR instances we propose a GVNS heuristic with range-limited neighborhood structures. First, we discuss the related work in Sect. 2 and give a formal problem definition in Sect. 3. The GVNS is described in Sect. 4, where we also introduce the so-called *synchronization mechanism* that allows us to quickly determine the changed makespan of the incumbent solution when a neighborhood move is applied. In this way our GVNS algorithm is able to quickly scan through the used neighborhoods. In Sect. 5 experimental results are provided, which indicate that this mechanism

is rather independent of the number of jobs and therefore also applicable for larger problem instances. Ultimately, the proposed GVNS is able to provide new state-of-the-art results for many hard-to-solve instance classes of the JSOCMSR.

## 2 Related Work

As mentioned the JSOCMSR was already approached by Horn et al. [5, 8], who also proved the NP-hardness of the problem. The authors suggested methods for calculating lower bounds for the makespan, given a partial solution with still open jobs. Those lower bounds are then utilized in both a heuristic construction algorithm as well as a novel exact anytime A\* search. The latter performs after a certain number of classical A\* node expansions a beam search, starting from the currently selected node. In this way the A\* search is able to provide besides a proven optimal solution at the end also promising intermediate heuristic solutions. The latter are especially valuable for hard instances where runtimes would be too excessive and the search must be terminated prematurely. This A\* search was compared, among others, to a compact position based *mixed integer linear programming* (MIP) model solved with CPLEX as well as a *constraint programming* (CP) model solved with ILOG CP Optimizer. The experimental evaluation shows that the A\* search clearly dominates the considered competitors.

A problem strongly related to the JSOCMSR is considered by Veen et al. [16] with the important difference that post-processing times are negligible compared to the total processing times of the jobs. This property allows to treat the problem as a traveling salesman problem with a special cost structure, which can be solved efficiently in time  $O(n \log n)$ , where  $n$  is the number of jobs. For other related problems we refer to [5].

A prize-collecting variant of the JSOCMSR (PC-JSOCMSR) is considered by Horn et al. [7] as well as by Maschler and Raidl [12]. In both works, each job is further equipped with a prize and a set of time windows such that the job can only be scheduled within one of its time windows. The objective is to find a subset of jobs together with a feasible schedule such that the overall prize of the scheduled jobs is maximized. In Horn et al. [7] an exact A\* algorithm is proposed for the PC-JSOCMSR, where corresponding upper bound calculations are based on Lagrangian and linear programming relaxations. The A\* algorithm is able to solve small instances with up to 30 jobs to optimality; see [6] for an extended version of the original conference paper. Experiments showed that A\* search outperforms a compact MIP model solved by CPLEX as well as a MiniZinc CP model solved by different back-end solvers. Maschler and Raidl [12] investigated different heuristic methods to solve larger instances with up to 300 jobs. These methods are based on *multivalued decision diagrams* (MDDs) and general variable neighborhood search. Both works, [7, 12], were then extended by Horn et al. [4] by utilizing a novel construction algorithm for relaxed MDDs. On the basis of these, new state-of-the-art results could be obtained for PC-JSOCMSR instances with up to 500 jobs.

### 3 Problem Formalization

The JSOCMSR consists of a finite set  $J = \{1, \dots, n\}$  of  $n$  jobs, the common resource 0, and a set  $R = \{1, \dots, m\}$  of  $m$  secondary resources. Let  $R_0 = \{0\} \cup R$  be the set of all resources. Each job  $j \in J$  requires one specific secondary resource  $q_j \in R$  for its whole processing time  $p_j > 0$ . Let  $J_r = \{j \in J \mid q_j = r\}$  be the subset of jobs requiring resource  $r \in R$  as secondary resource. Moreover, each job  $j$  needs after some pre-processing time  $p_j^{\text{pre}} \geq 0$ , counted from the job's start time, also the common resource 0 for a time  $0 < p_j^0 \leq p_j - p_j^{\text{pre}}$ . For convenience we define the post-processing time, where the secondary resource is still needed but not the common resource anymore, by  $p_j^{\text{post}} = p_j - p_j^{\text{pre}} - p_j^0$ . A solution to the problem is described by the jobs' starting times  $s = (s_j)_{j \in J}$  with  $s_j \geq 0$ . A solution  $s$  is feasible if no two jobs require the same resource at the same time. The objective is to find a feasible solution  $s$  that minimizes the makespan  $\text{MS}(s) = \max\{s_j + p_j \mid j \in J\}$ , i.e., the time the last job finishes its execution.

Since jobs acquire the common resource 0 excursively, a solution implies a total ordering of the jobs. Vice versa, any permutation  $\pi = (\pi_i)_{i=1, \dots, n}$  of jobs in  $J$  can be decoded into a feasible solution in a greedy way by scheduling each job in the given order at the earliest feasible time. We refer to a schedule obtained in this way as a *normalized schedule*. By the notation  $\text{MS}(\pi)$  we refer to the makespan of a normalized schedule induced by the job permutation  $\pi$ . Since any optimal solution is either a normalized schedule or there exists a corresponding normalized schedule with the same objective value, we can restrict our search to job permutations and their corresponding normalized schedules. Job permutations are therefore the primary solution representation in the suggested GVNS.

### 4 Variable Neighborhood Search

The well known *variable neighborhood search* (VNS) metaheuristic, introduced by Mladenović and Hansen [14], has been successfully applied on many combinatorial optimization problems; for a comprehensive review see [2]. To heuristically solve the JSOCMSR we use a GVNS, where two different sets of neighborhood structures  $N_{i=1 \dots k_{\max}}^I$  and  $N_{i=1 \dots l_{\max}}^S$  are alternately applied in intensification and diversification phases. In the intensification phase, a deterministic *variable neighborhood descent* (VND) uses a set of  $k_{\max} = 4$  intensification neighborhood structures, which are searched, depending on their computational cost, in either a first-improvement or best-improvement manner. In the diversification phase a set of  $l_{\max} = 23$  increasingly perturbative shaking neighborhood structures are used to perform random moves in order to reach parts of the search space that are farther away from the incumbent solution. Algorithm 1 illustrates this procedure. The initial solution—represented by permutation  $\pi$ —is created uniformly at random. The GVNS terminates if a certain time-limit is exceeded or the incumbent solution's objective value corresponds to the strongest lower bound  $\text{MS}^{\text{LB}}$  obtained from [5]. In the latter case a proven optimal solution has been found.

---

**Algorithm 1.** General Variable Neighborhood Search

---

```

1: Input: initial solution  $\pi$ ,  $N_{i=1,\dots,k_{\max}}^J, N_{j=1,\dots,l_{\max}}^S$ 
2:  $\pi^{\text{best}} \leftarrow \pi; l \leftarrow 1$ 
3: repeat
4:    $\pi' \leftarrow \text{Shake}(N_l^S, \pi^{\text{best}})$  ▷ diversification
5:    $\pi'' \leftarrow \text{VND}(N_l^J, \pi')$  ▷ intensification
6:    $l \leftarrow l + 1$ 
7:   if  $\text{MS}(\pi') < \text{MS}(\pi^{\text{best}})$  then ▷ new incumbent solution found
8:      $\pi^{\text{best}} \leftarrow \pi'; l \leftarrow 1$ 
9:   else if  $l > l_{\max}$  then ▷ continue with next shaking neighborhood structure
10:     $l \leftarrow 1$ 
11:   end if
12: until  $\text{MS}^{\text{LB}} = \text{MS}(\pi^{\text{best}}) \vee$  time-limit reached
13: return  $\pi^{\text{best}}$ 

```

---

### 4.1 Solution Representation and Evaluation

As mentioned in Sect. 3, our VNS interprets solutions to the JSOCMSR as linear permutations that state the order in which the jobs acquire the common resource 0. To obtain the makespan  $\text{MS}(\pi)$  of such a permutation  $\pi$ , the exact starting time  $s_j$  for each job  $j \in J$  must be determined. This is done by a linear time decoder that greedily schedules each job as soon as its resources become available. As it becomes quite inefficient to naively apply this decoder during neighborhood evaluation, we propose an incremental evaluation scheme in which it is not always necessary to (re-)determine the starting time for each job to obtain its makespan.

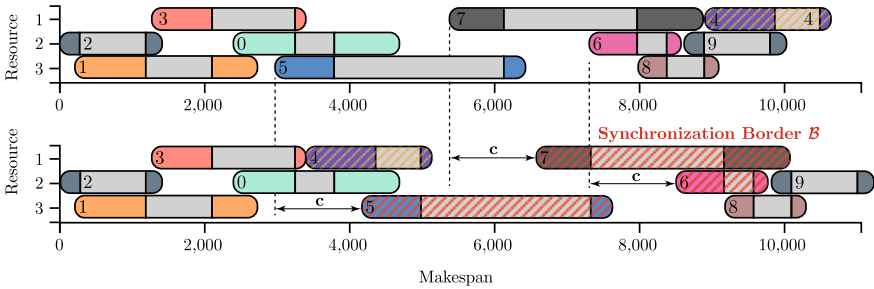
However, due to the incremental nature of the decoding mechanism and a solution’s consequential characteristic, that even small structural changes—like the removal of a job from its current position—potentially propagate to distant sections in the solution, a strictly constant-time incremental evaluation schema is not possible. Instead, we concentrated on an alternative approach, where a certain subsection of a neighboring solution is evaluated until a point of *synchronization* with respect to the incumbent solution is identified. After this point, no structural differences besides a fixed time offset occur. This point of synchronization in the permutation resides at the end of a so-called *synchronization border*, consisting of a minimal set of jobs on different secondary resources which are aligned w.r.t. their starting times in the incumbent solution and the respective neighboring solution in the same way. In the following we define this formally.

**Definition 1 (Synchronization Border).** *Given two solutions  $\pi, \pi'$  and the respective normalized starting times  $s$  and  $s'$ , where  $\pi'$  is a neighbor of  $\pi$  w.r.t. some neighborhood structure  $\mathcal{N}$ . Assume further that the underlying permutation of jobs has only changed up to position  $i$ ,  $0 \leq i < n$ . The synchronization point is then the smallest position  $i'$  with  $i < i' \leq n$ , where a set of jobs  $\mathcal{B} \subseteq \{\pi_k \mid k = i + 1, \dots, i'\}$ , denoted as the synchronization border, satisfies the following conditions:*



1. The set contains exactly one job for each secondary resource that is still claimed by a job in the permutation at or after the synchronization point  $i'$ .
2. The jobs are aligned with respect to their starting times in the same way in  $s$  and  $s'$ , i.e.,  $\exists c \in \mathbb{Z} \forall j \in \mathcal{B} : s_j - s'_j = c$ .

In order to evaluate the makespan of a neighbor  $\pi'$  of the incumbent solution  $\pi$ , our approach starts at the first position in the permutation subject to the structural change induced by the move in the neighborhood and scans through the permutation to identify the synchronization border. As soon as the synchronization border is established we are able to determine the *alignment offset*  $c$ , i.e., the time difference between the solutions concerning the border, and, consequently, can immediately derive the makespan  $MS(\pi')$  of the neighbor solution  $\pi'$ . Figure 1 illustrates this approach, where a neighboring solution  $\pi'$  on the bottom is derived from  $\pi$  by removing job 4 from position 9 and reinserting it at its new position 4. In this example, the synchronization border  $\mathcal{B} = \{5, 7, 6\}$  can be determined already after three steps, allowing to derive the makespan of  $\pi'$  already at position 8.



**Fig. 1.** Illustration of an incumbent solution (top) and a neighboring solution obtained after moving job 4 (bottom) and their synchronization border  $\{5, 7, 6\}$ .

As identifying the synchronization border in a naive iterative way requires time  $\mathcal{O}(nm)$  in the worst case, we use additional auxiliary data structures for each incumbent solution that frequently allow to skip certain parts of the scan through the permutation. In this way the synchronization border can typically be identified much quicker and as a consequence the exploration of the neighborhoods is more efficient. Besides simple lookup tables to detect, for instance, the last job on a particular resource, most importantly, our approach relies on a data structure  $\alpha(\pi) = (\alpha_{i,r}(\pi))_{i=1,\dots,n, r \in R_0}$  indicating for each position  $i$  in permutation  $\pi$  the time from which on each resource  $r$  is available for scheduling a job at this position  $i$ . Thus,  $\alpha_{i,r}(\pi)$  can be used to quickly determine the starting time of a job which should be inserted in  $\pi$  at position  $i$ . As all our neighborhood structures are essentially defined by removing and re-inserting jobs in the permutation representation in certain ways, this data structure allows to immediately

determine the starting time of an inserted job at any position, subsequently requiring only the identification of the synchronization border to determine the implied change in the makespan. Although the preparation of these data structures comes with an additional computational cost of  $\mathcal{O}(nm)$  per incumbent solution for which the VND is started, our experiments in Sect. 5 indicate that in practice the whole approach requires only constant amortized runtime with respect to the number of jobs for identifying the synchronization border and thus the makespan of a neighboring solution.

### 4.2 Intensification

The VND, which is responsible for intensification within the GVNS, makes use of a set of neighborhood structures for linear permutations, as formally defined by Schiavinotto and Stützle [15].

The *insertion neighborhood*  $\mathcal{N}_I(\pi)$  of an incumbent solution  $\pi$  consists of any solution  $\pi'$  obtained by removing any job  $j$  from its current position in  $\pi$  and reinserting it at any other position. We efficiently evaluate the whole neighborhood by considering the removal of each job  $j \in J$  in an outer loop, yielding a partial solution  $\pi \ominus j$  for which the corresponding auxiliary data structure  $\alpha(\pi \ominus j)$  is derived and the partial neighborhood  $\mathcal{N}'_I(\pi \ominus j, j)$  corresponding to the re-insertion of  $j$  at any position except the original one is evaluated in an inner loop. Algorithm 2 shows in more detail how the neighbor solution in which job  $j$  is re-inserted at a position  $i$  in the partial solution  $\pi \ominus j$  is evaluated by determining the synchronization border and the respective alignment offset.

Based on this evaluation scheme, it turned out to be advantageous in the implementation to further divide the insertion neighborhood  $\mathcal{N}_I(\pi)$  into forward and backward insertion neighborhoods such that jobs are only allowed to move forward or backward in the permutation, respectively. This allows to reuse some part of the auxiliary data structures for the entire neighborhood evaluation.

---

**Algorithm 2.** Evaluation of the neighbor in which job  $j$  is reinserted at position  $i$

---

```

1: Input: partial solution  $\pi \ominus j$ , insertion position  $i$ , resource availability times  $\alpha(\pi \ominus j)$ 
2:  $t_r \leftarrow \alpha_{i,r}(\pi \ominus j), \forall r \in R_0$ 
3: synchronization border  $B = \emptyset$ , aligned offset  $c \leftarrow 0$ 
4: for  $k = i, \dots, |\pi \ominus j|$  do                                 $\triangleright$  evaluate  $\pi \ominus j$  from insert position onwards
5:    $j' \leftarrow (\pi \ominus j)_k$ 
6:    $s_{j'} \leftarrow \max\{t_0 - p_{j'}^{\text{pre}}, t_{q_{j'}}\}$                  $\triangleright$  evaluate new starting time for  $j'$ 
7:    $t_0 \leftarrow s_{j'} + p_{j'}^{\text{pre}} + p_{j'}^0; t_{q_{j'}} \leftarrow s_{j'} + p_{j'}^{\text{pre}}$ 
8:   update  $B$  with job  $j'$ 
9:   if  $B$  satisfies conditions from Definition 1 then
10:      $c \leftarrow$  derive alignment offset from  $B$  and  $\pi$ 
11:     break
12:   end if
13: end for
14: return  $\text{MS}(\pi \ominus j) + c$ 

```

---

The exchange neighborhood  $\mathcal{N}_X(\pi)$ , contains any solution derived from the incumbent  $\pi$  by exchanging any pair of jobs in the permutation. Again, the neighborhood evaluation is based on determining synchronization borders, but instead of using intermediate partial solutions, a dual synchronization approach has been devised, where the neighborhood operation is essentially reduced to two insertion operations, where both the offset between the respective exchanged jobs as well as the offset of the latter job to the makespan are obtained with the synchronization technique.

In addition to efficient evaluation schemes for the considered neighborhood structures, we further studied different approaches to reduce neighborhood sizes in order to avoid the evaluation of unpromising neighbors at all. Besides neighborhood reduction based on critical jobs as proposed already by Horn et al. [5], we also considered heuristic approaches like avoiding to schedule two jobs of the same secondary resource consecutively or reducing the size of neighborhoods by limiting the maximum distance of move operations. While these pruning techniques bring the danger of quickly approaching local optima of rather poor quality, concentrating on critical jobs is particularly advantageous in the very beginning of the search. Limiting the maximum distance of move operations particularly showed its effectiveness for exchange neighborhoods, where instead of the dual synchronization evaluation scheme, it becomes more the better option to partially evaluate the entire range between the positions of the two exchanged jobs and perform a single synchronization step at the end of this range. Experimentally, we determined a move distance limitation of  $k = 50$  to provide a good trade-off between the size of the neighborhood and its evaluation's efficiency in the context of our benchmark instances. Nevertheless note that these restricted neighborhoods are primarily used in early VND phases, while more comprehensive neighborhoods become important in latter phases to compensate the limitations. More details on the pruning techniques and their impacts can be found in the first author's master thesis [10]. Here, we will only look more closely on the limitation of move distances.

We used findings of a landscape analysis, where the average quality and depth of local optima were studied to prepare a meaningful parameter tuning configuration, and then applied `irace` [11] to select concrete neighborhood structures and parameters like the step function by which the neighborhoods are searched in the VND. For details regarding the parameter tuning setup we refer to [10]. Finally, we investigated the temporal behavior of our algorithm in a set of experiments to decide the neighborhood change function in the VND [3]. Again, more details on this preliminary investigations can be found in [10].

The finally resulting VND configuration uses four neighborhood structures, subject to a piped neighborhood change function [3]. First, an exchange neighborhood structure with a move distance limitation of 50 is used in conjunction with a first-improvement step function to quickly identify local optima of already relatively high quality. This is followed by the backward insertion neighborhood structure searched in a best improvement manner. Next, the unconstrained exchange neighborhood structure is used and finally the unconstrained

insertion neighborhood structure, again searched in first and best improvement manners, respectively.

### 4.3 Diversification

For diversification, the GVNS applies moves from a total of  $l_{\max} = 23$  shaking neighborhood structures to the incumbent solution, where each shaking neighborhood  $N_i^S$  is parametrized by  $\kappa_i$  describing the number of subsequent applications of the underlying neighborhood move. In order to enable our shaking procedure to introduce fine-grained structural changes into the incumbent solution, we use the exponentially growing function  $\kappa_i = \lceil \exp(\frac{i \cdot \log(n)}{\kappa_{\max} - 1}) \rceil$ , with a maximum number of applied moves per shaking neighborhood of  $\kappa_{\max} = 32$ , to generate two sets of 10 insertion and exchange shaking neighborhood structures respectively. Starting with insertions, those sets are then interleaved and at positions four, ten and twenty extended by a subsequence inversion shaking neighborhood applying one, two and four inversions of five jobs respectively.

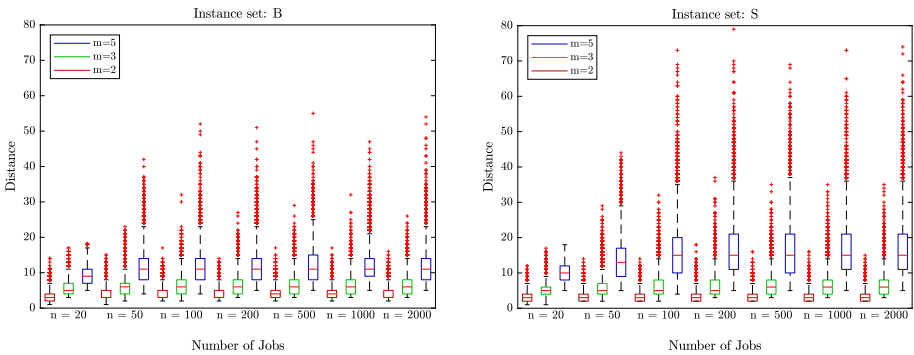
This configuration was mainly hand-crafted based on characteristics of the ruggedness of the respective neighborhood structures and the general structure of the search space. We used the autocorrelation function on random walks of length  $10^6$  to estimate the ruggedness of neighborhood structures and analyzed a large set of globally optimal solutions obtained from  $3.75 \times 10^6$  runs on a diverse set of 300 instances with  $n = 30$  jobs, to gain insight on the distribution of globally optimal solutions in the search space. We found that the studied instances contain a relatively high number of distinct globally or at least nearly optimal solutions, being widely distributed in the search space. A primary reason for this is likely the dependency structure inherent to the problem and the induced symmetries, caused by resource imbalance or utilization gaps on secondary resources, frequently allowing to exchange of jobs on secondary resources without affecting the makespan. For more details, see [10].

## 5 Computational Results

In our computational study we analyzed the practical applicability and impact of the proposed incremental evaluation technique and compared the GVNS to the baselines provided by Horn et al. [5]. These are the anytime A\* algorithm and a CP model. The experiments were conducted on two sets of instances with different characteristics with respect to the workload distribution among the available resources [5]. Balanced instances in set B have the workload uniformly distributed among the secondary resources and obtained durations  $p_j^{\text{pre}}$  and  $p_j^{\text{post}}$  for the pre-processing and post-processing of jobs by sampling the discrete uniform distribution  $\mathcal{U}\{0, 1000\}$  and durations  $p_j^0$  of the main processing phases by sampling  $\mathcal{U}\{1, 1000\}$ . Instances in set S, on the other hand, show a skewed workload distribution, both with respect to the assignment to secondary resources and the utilization of the common resource 0. In skewed instances, a job is assigned to the secondary resource 1 with probability 0.5, while the probability

for the remaining secondary resources  $m > 1$  is  $1/(2m - 2)$ . Both sets consist of instances with  $n \in \{50, 100, 200, 500, 1000, 2000\}$  jobs and  $m \in \{2, 3, 5\}$  secondary resources with 50 randomly sampled instances for each  $(n, m)$  pair. The instance sets are available at <https://www.ac.tuwien.ac.at/research/problem-instances/>. The proposed GVNS was implemented in C++ using G++ 7.4.0 with *-Ofast* optimization level. The experiments were conducted on a computing cluster of 16 machines, each with two *Intel Xeon E5-2640 v4* CPUs with 2.40 GHz in single threaded mode and 15 GB RAM. All considered approaches were executed with a maximum CPU time limit of 900s. The baseline CP model was solved with ILOG CP Optimizer 12.7.1.

In order to study the practical efficiency of our incremental evaluation approach, an experiment was conducted where  $10^4$  randomly selected neighborhood moves in exchange and insertion neighborhoods were applied and the distance from the structural change to the last job in the synchronization border—that is the number of steps until the synchronization border could be determined—was traced. Figure 2 shows the synchronization distance of balanced and skewed instances of different sizes. For the considered instances, it can be observed that our approach exhibits an average amortized runtime behavior that is constant in the number of jobs, but increases with the number of secondary resources due to the nature of the synchronization border. Moreover, Fig. 2 illustrates the sensitivity of the approach to significant resource imbalance, indicated by a higher number of outliers observed in the skewed instance set, likely due to large sections in the schedules where secondary resources are not utilized.



**Fig. 2.** Synchronization distance: number of steps required to identify the synchronization border in balanced and skewed instances, starting from the position of structural change due to a neighborhood move.

Finally, Table 1 compares average results of our GVNS on different instance classes to the baselines of Horn et al. [5]. Columns %-gap state the final optimality gaps in percent, which is calculated by  $100\% \cdot (\text{MS}(\pi) - \text{MS}^{\text{LB}}) / \text{MS}^{\text{LB}}$ , whereas columns %-opt lists the percentage of proven optimal solutions. Both columns use the best lower bound  $\text{MS}^{\text{LB}}$  obtained from Horn et al. [5].

Columns  $\sigma_{\%-\text{gap}}$  show the standard deviations of the corresponding average optimality gaps. Column t provides the median time the GVNS required to obtain its best solution in a run. To obtain statistically more stable results, we executed the GVNS ten times for each of the 50 instances per instance class. For the anytime A\* algorithm and for the CP solver, column t shows the median time when the algorithms terminated either because the optimal solution has been found or the time- or memory limit was exceeded.

Generally, Table 1 shows that the GVNS manages to obtain heuristic solutions comparable to those of the A\* search, while both approaches show their specific advantages on particular subsets of instances. For balanced instances, on the one hand, A\* search already showed its effectiveness, where even large instances up to 2000 jobs could be solved to proven optimality. For instances with  $m = 2$  and  $m = 5$ , the GVNS obtains similar results with respect to solution quality, although the temporal performance decreases with increasing instance size in comparison. For instances with  $m = 3$  the GVNS's solutions are clearly worse than those of the A\* search, although the average optimality-gap of  $\leq 0.288\%$  is still small and much better than the one of the CP approach. In Kaufmann [10] we show that providing an initial solution obtained with the least lower-bound construction heuristic of Horn et al. [5] can further improve the solution quality for this particular instance set, however, A\* is still superior both with respect to quality as well as temporal behavior.

For the harder skewed instances, on the other hand, our GVNS shows a significant improvement compared to both baseline methods with an average optimality gap below 0.214%. Instances with two secondary resources tend to be among the more difficult ones, where even for small instances with 50 jobs, optimality could be proven with the lower bound only in 42% of the runs. This, however, could as well be an indicator for the lower bounds being off the optimum. Interestingly, the GVNS still shows an improvement with respect to the number of obtained proven optimal solutions, where particularly for small to moderately large instances up to 88% could be solved to proven optimality, despite the inherent incompleteness of the GVNS.

## 6 Conclusions

In this work, we presented a GVNS to heuristically tackle the JSOCMSR, a combinatorial optimization problem encountered for example in novel cancer treatment facilities. We devised a generally applicable approach to efficiently evaluate solutions in the course of a neighborhood search in incremental ways and applied it to variants of insertion and exchange neighborhood structures. Insertion and exchange moves were utilized in the intensification phase, a piped VND, as well as in the diversification phase as for randomized shaking.

Our experimental analysis first dealt with the practical efficiency of the incremental evaluation scheme, which still has a linear runtime in the number of jobs in the worst-case but exhibits a essentially a constant average runtime on all our benchmark instances. When comparing the GVNS to the state-of-the-art A\*

**Table 1.** Average results of GVNS, A\* search, and the CP approach.

Type	n	m	GVNS				Anytime A*				CP/ILOG			
			%-gap	$\sigma_{\text{-gap}}$	%-opt	t[s]	%-gap	$\sigma_{\text{-gap}}$	%-opt	t[s]	%-gap	$\sigma_{\text{-gap}}$	%-opt	t[s]
B	50	2	<b>0.000</b>	0.00	100.0	<0.1	<b>0.000</b>	0.00	100.0	1.1	<b>0.000</b>	0.00	100.0	<0.1
B	100	2	<b>0.000</b>	0.00	100.0	<0.1	<b>0.000</b>	0.00	100.0	2.0	<b>0.000</b>	0.00	100.0	<0.1
B	200	2	<b>0.000</b>	0.00	100.0	0.2	<b>0.000</b>	0.00	100.0	5.4	<b>0.000</b>	0.00	100.0	<0.1
B	500	2	<b>0.000</b>	0.00	100.0	2.4	<b>0.000</b>	0.00	100.0	35.3	<b>0.000</b>	0.00	100.0	1.3
B	1000	2	<b>0.000</b>	0.00	100.0	13.0	<b>0.000</b>	0.00	100.0	8.9	<b>0.000</b>	0.00	100.0	9.2
B	2000	2	<b>0.000</b>	0.00	100.0	83.5	<b>0.000</b>	0.00	100.0	46.3	<0.001	0.01	98.0	63.5
B	50	3	0.050	0.22	91.2	<0.1	<b>0.017</b>	0.08	96.0	1.1	0.068	0.30	92.0	<0.1
B	100	3	0.112	0.29	79.6	0.1	<b>0.021</b>	0.09	92.0	2.0	0.226	0.55	78.0	4.2
B	200	3	0.176	0.45	74.0	2.1	<b>0.016</b>	0.06	92.0	5.9	0.556	1.12	56.0	319.4
B	500	3	0.260	0.42	47.0	422.3	< <b>0.001</b>	<0.01	98.0	35.9	2.212	1.83	20.0	900.0
B	1000	3	0.216	0.33	31.0	385.0	<b>0.001</b>	<0.01	98.0	6.1	3.094	1.46	2.0	899.9
B	2000	3	0.288	0.34	15.0	843.2	<b>0.005</b>	0.04	98.0	23.8	4.220	1.20	0.0	900.0
B	50	5	<0.001	<0.01	99.4	0.1	<b>0.000</b>	0.00	100.0	1.2	<b>0.000</b>	0.00	100.0	0.7
B	100	5	<b>0.000</b>	0.00	100.0	0.4	<b>0.000</b>	0.00	100.0	2.2	<b>0.000</b>	0.00	100.0	9.5
B	200	5	<b>0.000</b>	0.00	100.0	2.3	<0.001	0.00	98.0	6.5	<b>0.000</b>	0.00	100.0	91.3
B	500	5	<b>0.000</b>	0.00	100.0	14.3	<b>0.000</b>	0.00	100.0	42.3	<0.001	<0.01	86.0	499.7
B	1000	5	<0.001	<0.01	96.0	49.2	<b>0.000</b>	0.00	100.0	7.9	0.359	0.12	0.0	900.0
B	2000	5	<0.001	<0.01	86.6	128.8	<b>0.000</b>	0.00	100.0	30.4	0.478	0.14	0.0	900.0
S	50	2	<b>0.163</b>	0.23	42.0	4.8	0.268	0.38	40.0	11.4	0.210	0.28	42.0	899.9
S	100	2	<b>0.172</b>	0.32	33.8	115.5	0.367	0.49	26.0	44.8	0.323	0.47	12.0	900.0
S	200	2	<b>0.111</b>	0.18	14.8	606.0	0.440	0.33	2.0	65.2	0.642	0.51	0.0	900.0
S	500	2	<b>0.095</b>	0.08	0.0	831.7	0.532	0.18	0.0	88.7	2.736	0.51	0.0	900.0
S	1000	2	<b>0.105</b>	0.06	0.0	813.3	0.725	0.20	0.0	176.8	4.636	0.43	0.0	900.0
S	2000	2	<b>0.214</b>	0.11	0.0	892.6	0.786	0.18	0.0	252.7	4.784	0.39	0.0	900.0
S	50	3	<b>0.035</b>	0.15	82.0	0.2	0.053	0.21	82.0	1.3	<b>0.035</b>	0.15	80.0	27.7
S	100	3	<b>0.030</b>	0.10	82.8	3.5	0.153	0.37	50.0	16.5	0.060	0.15	52.0	899.7
S	200	3	<b>0.025</b>	0.11	78.8	21.5	0.117	0.26	34.0	26.4	0.135	0.21	36.0	899.8
S	500	3	<b>0.006</b>	0.02	42.4	370.5	0.177	0.24	14.0	121.6	1.360	0.76	4.0	900.0
S	1000	3	<b>0.009</b>	0.02	19.2	584.5	0.621	0.47	2.0	48.0	2.872	0.93	0.0	900.0
S	2000	3	<b>0.041</b>	0.05	5.8	863.3	0.701	0.41	0.0	80.2	4.296	0.98	0.0	900.0
S	50	5	0.046	0.14	83.7	<0.1	0.077	0.19	80.0	1.4	<b>0.045</b>	0.14	84.0	15.0
S	100	5	<b>0.006</b>	0.02	88.4	1.2	0.064	0.18	66.0	6.1	0.019	0.04	70.0	899.6
S	200	5	<b>0.034</b>	0.14	77.2	18.3	0.281	0.49	34.0	38.8	0.161	0.25	28.0	900.0
S	500	5	<b>0.009</b>	0.02	46.8	351.7	0.347	0.34	16.0	188.6	1.229	0.95	8.0	899.9
S	1000	5	<b>0.012</b>	0.02	22.2	625.3	0.702	0.50	0.0	387.3	2.478	1.11	0.0	900.0
S	2000	5	<b>0.105</b>	0.10	2.0	893.6	0.915	0.54	0.0	789.3	4.229	1.22	0.0	900.0

search and the CP model, we observed the GVNS's ability to obtain high-quality solutions for a diverse set of large instances with an average optimality-gap of  $\leq 0.288\%$ . Although for balanced instances, the anytime A\* algorithm of Horn et al. [5] was out of reach for particularly hard instances, our approach showed its effectiveness on harder instances with skewed workloads, where the state of the art could be improved significantly. In future work it would be interesting to investigate the runtime of the incremental evaluation scheme also from a theoretical point-of-view, in the hope that the constant amortized time observed here in practice can even be proven for a larger class of instances. Moreover, it appears promising to apply the underlying ideas of the proposed incremental

evaluation scheme also in the context of related scheduling/sequencing problems and local search based metaheuristics.

## References

1. Conforti, D., Guerriero, F., Guido, R.: Optimization models for radiotherapy patient scheduling. *4OR* **6**(3), 263–278 (2008)
2. Hansen, P., Mladenović, N., Pérez, J.A.M.: Variable neighbourhood search: methods and applications. *Ann. Oper. Res.* **175**(1), 367–407 (2010)
3. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. *EURO J. Comput. Optim.* **5**(3), 423–454 (2016). <https://doi.org/10.1007/s13675-016-0075-x>
4. Horn, M., Maschler, J., Raidl, G., Rönnberg, E.: A\*-based construction of decision diagrams for a prize-collecting scheduling problem. Technical report AC-TR-18-011, Algorithms and Complexity Group, TU Wien (2018)
5. Horn, M., Raidl, G., Blum, C.: Job sequencing with one common and multiple secondary resources: an A\*/Beam Search based anytime algorithm. *Artif. Intell.* **277**(103173) (2019)
6. Horn, M., Raidl, G., Rönnberg, E.: A\* search for prize-collecting job sequencing with one common and multiple secondary resources. *Ann. Oper. Res.* (2020)
7. Horn, M., Raidl, G.R., Rönnberg, E.: An A\* algorithm for solving a prize-collecting sequencing problem with one common and multiple secondary resources and time windows. In: Proceedings of the 12th International Conference of the Practice and Theory of Automated Timetabling, PATAT 2018, pp. 235–256 (2018)
8. Horn, M., Raidl, G., Blum, C.: Job sequencing with one common and multiple secondary resources: a problem motivated from particle therapy for cancer treatment. In: Nicosia, G., Pardalos, P., Giuffrida, G., Umeton, R. (eds.) MOD 2017. LNCS, vol. 10710, pp. 506–518. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-72926-8\\_42](https://doi.org/10.1007/978-3-319-72926-8_42)
9. Kapamara, T., Sheibani, K., Haas, O., Petrovic, D., Reeves, C.: A review of scheduling problems in radiotherapy. In: Proceedings of the International Control Systems Engineering Conference, pp. 207–211. Coventry University Publishing (2006)
10. Kaufmann, T.: A variable neighborhood search for the job sequencing with one common and multiple secondary resources problem. Master's thesis, TU Wien, Vienna, Austria (2019)
11. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M.: The irace package: iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **3**, 43–58 (2016)
12. Maschler, J., Raidl, G.R.: Multivalued decision diagrams for a prize-collecting sequencing problem. In: Proceedings of the 12th International Conference of the Practice and Theory of Automated Timetabling, PATAT 2018, pp. 375–397 (2018)
13. Maschler, J., Riedler, M., Stock, M., Raidl, G.R.: Particle therapy patient scheduling: first heuristic approaches. In: Proceedings of the 11th International Conference of the Practice and Theory of Automated Timetabling, PATAT 2016, pp. 223–244 (2016)
14. Mladenović, N., Hansen, P.: Variable neighborhood search. *Comput. Oper. Res.* **24**(11), 1097–1100 (1997)



15. Schiavinotto, T., Stützle, T.: A review of metrics on permutations for search landscape analysis. *Comput. Oper. Res.* **34**(10), 3143–3153 (2007). <https://doi.org/10.1016/j.cor.2005.11.022>
16. Van der Veen, J.A.A., Wöginger, G.J., Zhang, S.: Sequencing jobs that require common resources on a single machine: a solvable case of the TSP. *Math. Program.* **82**(1–2), 235–254 (1998)



# Evolutionary Graph-Based V+E Optimization for Protection Against Epidemics

Krzysztof Michalak (✉) 

Department of Information Technologies, Faculty of Management,  
Wrocław University of Economics, Wrocław, Poland  
krzysztof.michalak@ue.wroc.pl

**Abstract.** Protection against spreading threats in networks gives rise to a variety of interesting optimization problems. Among others, vertex protection problems such as the Firefighter Problem and vaccination optimization problem can be tackled. Interestingly, in some cases a networked system can be made more resilient to threats, by changing its connectivity, which motivates the study of another type of optimization problems focused on adapting graph connectivity.

In this paper the above-mentioned approaches are combined, that is both vertex and edge protection is applied in order to stop the threat from spreading. Solutions to the proposed problem are evaluated using different cost functions for protected vertices and edges, motivated by real-life observations regarding the costs of epidemics control.

Instead of making decisions for each of the vertices and edges a decision model is used (based on rules or a neural network) with parameters optimized using an evolutionary algorithm. In the experiments the model using rules was found to perform better than the one based on a neural network.

**Keywords:** Disease prevention · Epidemics control · DPEC · Combinatorial optimization · Graph-based problems

## 1 Introduction

A wide variety of phenomena can be described as a spreading of a threat in a certain network. Epidemics, wildfires, floods and even bankruptcies behave in a similar way: a number of entities are affected by a threat which subsequently spreads to other entities in the system. Numerous approaches have been proposed to analyse epidemic processes in complex networks. A review of recent advancements in this area is presented in [26]. When there is a dangerous phenomenon spreading a question naturally arises how to stop this threat in a possibly effective way. This question gives rise to a number of optimization problems in which the goal is to utilize the available countermeasures to stop the threat, taking into account the costs and possible constraints.

### 1.1 Vertex Protection

One of the abstractions that is often studied in the field of research on optimization methods is the Firefighter Problem (FFP) [14] in which spreading of fire is simulated on a graph in discrete time steps. Vertices of the graph can be in one of the states ‘B’, ‘D’, ‘U’ which are interpreted, respectively, as the vertex being on fire, being defended by firefighters or being in an untouched state (neither burning nor defended). In the initial state some vertices are on fire (in the ‘B’ state) and the remaining ones are most often left untouched (in the ‘U’ state). Subsequently, the fire spreads from burning vertices to the untouched ones along the edges of the graph, which determine which vertices are adjacent and can therefore catch on fire. The need for optimization in the Firefighter Problem is motivated by the limitation of resources, which is represented by a constraint stating that in any given time step at most  $N_f$  vertices can become protected against fire (i.e. be set to the ‘D’ state).

A very similar problem formulation can be used for tackling epidemics, bankruptcies, etc. In each of these applications neighbourhood of the vertices can be described in a different way. In the FFP and many other problems the ways through which the threat spreads are represented by edges of a graph, but other representations are also possible, for example based on geographical locations and distances such as in the Foot-and-Mouth Disease (FMD) spreading model used in the paper [2]. Also, the contagion dynamics can be different in different problems. In the classical version of the FFP the spreading of fire is deterministic: an untouched vertex is guaranteed to catch on fire in a given time step if one of its neighbours is burning. A version of the FFP with non-deterministic spreading of fire has also been studied [25]. Epidemics are often modelled using compartmental models, such as SIR (Susceptible  $\rightarrow$  Infected  $\rightarrow$  Recovered), SIS (Susceptible  $\rightarrow$  Infected  $\rightarrow$  Susceptible) or SIRV model, which, apart from the susceptible, infected and recovered states, allows the entities to be vaccinated [18]. Epidemiological models are most often probabilistic, that is the transitions from one state to another happen with certain probabilities as opposed to deterministic changes in the classical version of the FFP. Yet another threat spreading mechanism was proposed by Burkholz [5] for an economic setting in which companies on the market may go bankrupt, and because of unpaid dues incur stresses on other companies. In this model failures spread deterministically, but in order for a company to fail the total load incurred by its bankrupt cooperators has to exceed a certain threshold.

In each of the above-mentioned scenarios certain actions can be taken in order to prevent the threat from spreading. In the case of the Firefighter Problem  $N_f$  vertices can become protected against fire in each time step and the protection is 100% effective (protected vertices remain in the ‘D’ state until the end of simulation and fire cannot get to them). Because the order in which vertices are protected is essential in the FFP, the most common solution representation uses permutations to represent the order in which vertices should be protected. In the case of epidemiological models the most common protection mechanism is vaccination. If the individuals are vaccinated before the pathogen

starts spreading (and so the time of vaccinations does not have to be taken into account) the decisions to vaccinate or not can be represented as a binary vector. Imposing a constraint on the total number of vaccinated vertices the K-Node Immunization Problem is formulated and solved using, among others, heuristic [7] and evolutionary methods [20]. For the Burkholz economic model an optimization problem was formulated [23] in which solutions are vectors of real numbers representing thresholds which can be adjusted by allowing companies to store reserves.

## 1.2 Edge Protection and Network Connectivity

Another approach to network protection is to consider edges instead of vertices [8]. This approach is particularly important in computer network protection [6]. The notion of edge-failure resilience is an actively researched topic in the literature focused on networks protection [19].

It is worth noticing that, as opposed to the computer network protection problem where it is profitable to increase network connectivity, there are optimization problems where the optimum does not coincide with the highest network connectivity. Notably, financial systems can show different level of resilience to shocks of varying magnitude [16] depending on connectivity, a phenomenon that also affects optimization problems for this kind of systems [24]. In the case of counter-epidemic optimization a *lower* connectivity can be expected to produce better outcomes. This effect is the basis of epidemic control strategies based on social distancing [13].

## 1.3 Overview of This Paper

In this paper a combination of the network protection approaches discussed above is studied in a scenario which concerns stopping an epidemic from spreading on a graph. The optimization problem studied here is the problem of optimizing a decision model that determines which action should be taken: protecting a vertex (vaccination) or limiting the number of contacts it makes in the network (isolation). The costs of infections and vaccinations are calculated taking into account the number of affected vertices. The costs of isolation are calculated as the product of the number of the removed edges and the number of time steps the isolation has lasted for. This is motivated by the fact, that contacts in the network represent vital activities undertaken by entities in the system and their removal may, for example, cause income to be lost for businesses with the loss dependent on the time the isolation lasts for.

In the following sections the optimization problem is defined (Sect. 2), the experiments (Sect. 3) and results (Sect. 4) are discussed. Section 5 concludes the paper.

## 2 Optimization Problem

The optimization problem tackled in this paper is a problem of finding a counter-epidemic strategy minimizing several criteria. The epidemic is simulated on

a graph  $G = \langle V, E \rangle$  in which the vertices represent entities that may become infected and the edges represent contacts. The states of the vertices and the transitions between them are based on the SIVR model [27]. There are four states: ‘S’ - susceptible, a vertex that is not infected, but may become so; ‘I’ - infected; ‘V’ - vaccinated, and thus immune to the disease; and ‘R’ - a vertex recovered from the disease, which in the SIVR model cannot be infected again.

Changes in the graph occur in discrete time steps  $t = 0, \dots$  and we will denote the state of the graph at time  $t$  by  $S_t \in \{ \text{‘S’}, \text{‘I’}, \text{‘V’}, \text{‘R’} \}^{|V|}$  and the state of an individual vertex in the graph  $v \in V$  at time  $t$  by  $S_t[v] \in \{ \text{‘S’}, \text{‘I’}, \text{‘V’}, \text{‘R’} \}$ . The initial state is  $S_0$  in which a fraction  $\alpha_{inf}$  of the vertices is infected, so  $\alpha_{inf}|V|$  randomly selected vertices are in the state ‘I’ and the remaining ones are in the state ‘S’. A susceptible vertex may become infected if it is adjacent to at least one infected vertex, with the probability of transmitting the disease from each infected neighbour equal to  $\beta$  per a time step. Recovery occurs with the probability  $\gamma$  per a time step. In each time step protective actions can be taken for each susceptible vertex. If the protective action  $\mathcal{P}^{(vac)}(v)$  is applied, the vertex  $v$  is vaccinated and changes its state to ‘V’ in which it remains until the end of the simulation. If the protective action  $\mathcal{P}_q^{(isol)}(v)$  is applied, the isolation level of the vertex  $v$  is changed, by activating or deactivating edges adjacent to  $v$  so that a fraction  $q$  of the edges adjacent to  $v$  is inactive. The number of deactivated edges for the vertex  $v$  is calculated as  $\text{Round}(q \cdot k(v))$ , where  $k(v)$  is the degree of the vertex  $v$  and the  $\text{Round}()$  function rounds the number to the nearest integer. When more, or fewer, edges need to be activated the edges that change the state are selected at random with uniform probability. An activation state of the edge  $e$  is denoted by  $\mathcal{A}[e]$  with the value of *true* representing an active edge and the value of *false* representing an inactive edge. The disease can only spread along active edges. Therefore, applying the protective action  $\mathcal{P}_q^{(isol)}(v)$  with  $q > 0$  represents a situation when contacts are broken by the vertex  $v$  in order to reduce the risk of being infected. In each time step protective actions are applied before the spreading of the disease takes place (cf. Algorithm 1).

Instead of deciding which protective action to apply for each vertex separately, the optimizer adjusts parameters of a decision model  $\Psi$  which takes the information about nearby cases of the disease as inputs and returns the decision which action to perform (if any) in a given time step  $t$  for the vertex  $v$ . A solution to the optimization problem is the vector of parameters  $x \in \mathbb{R}^k$  of the decision model  $\Psi$ , where the length  $k$  of the vector of parameters  $x$  depends on the type of the decision model used. The vector of inputs  $\phi(v) \in \mathbb{R}^h$  representing the information about nearby cases of the disease contains fractions of infected vertices separated from  $v$  by  $1, \dots, h$  edges, where  $h$  is the horizon around the vertex  $v$  within which infected vertices are detected. The vector  $\phi(v)$  is obtained by performing the Breadth-First Search (BFS) [15] around the vertex  $v$ . For example in the situation shown in Fig. 1 the result is  $\phi(v) = [\frac{1}{3}, \frac{2}{3}, \frac{1}{2}]$ . At the distance  $d = 1$  there are three vertices in the states ‘S’, ‘I’, ‘V’, of which only one is infected, hence  $\phi(v)[1] = \frac{1}{3}$ . The fourth vertex (marked (1) in the figure) is connected through an inactive edge  $e$ , so it is not counted. At the distance

$d = 2$  there is one vertex in the state ‘S’ and two vertices in the state ‘I’, hence  $\phi(v)[2] = \frac{2}{3}$ . The two vertices marked (2) in the figure are separated from  $v$  by a vaccinated vertex, so they are not counted. At the distance  $d = 3$  the two pairs of vertices marked (3) and (4) are separated from  $v$  by infected vertices, so they are not counted. Therefore, only two vertices are taken into account, one susceptible and one infected, and  $\phi(v)[3] = \frac{1}{2}$ .

Algorithm 1 presents the simulation of the epidemic with selection of protective actions performed using the decision model  $\Psi$ . Inputs and outputs of this algorithm are listed in Table 1. Using this simulation procedure three objectives are calculated:  $N_{inf}$  - the number of vertices infected during the simulation,  $N_{vacc}$  - the number of vertices vaccinated during the simulation, and  $N_{isol}$  - the number of edges cut because of isolation, multiplied by the number of time steps in which the isolation was applied. Because the spreading of the epidemic is non-deterministic, the simulations are repeated  $N_{sim}$  times, each time starting from a different set of infected vertices, and the results are averaged. In this paper the number of simulations was set to  $N_{sim} = 5$ . Therefore, the optimization problem tackled in this paper can be formalized as follows:

$$\begin{aligned}
 & \text{minimize } (N_{inf}(x), N_{vacc}(x), N_{isol}(x)) = F(x) \in \mathbb{R}^3 \\
 & \text{subject to } x \in \mathbb{R}^k,
 \end{aligned} \tag{1}$$

where:

$k$  - the number of parameters of the decision model  $\Psi$ .

**Table 1.** Inputs and outputs of Algorithm 1.

Inputs:	
$G = \langle V, E \rangle$	- the graph on which the epidemic is simulated
$\Psi$	- the decision model used for selecting protective actions
$h$	- the radius of the horizon in which to count infected vertices
$\alpha_{inf}$	- the fraction of initially infected vertices
$x$	- a solution to evaluate (a vector of parameters for the decision model $\Psi$ ), $x \in \mathbb{R}^k$
Output:	
$F(x)$	- the vector of objectives $F(x) = (N_{inf}, N_{vacc}, N_{isol})$

### 3 Experiments

In the experiments evolutionary multiobjective optimization was carried out on instances of the optimization problem described in Sect. 2. The parameters for the spread of the epidemic were: the infected fraction  $\alpha_{inf} = 0.01$ , the transmission probability  $\beta = 0.5$  and the recovery probability  $\gamma = 0.1$ . A small fraction

---

**Algorithm 1:** Evaluation of a solution to the optimization problem by simulating the epidemic (inputs and outputs: see Table 1).

---

```

// Initialize the simulation
S := RandomlyInfected( $\alpha_{inf}|V|$ )
for  $e \in E$  do
   $\mathcal{A}[e] := \text{true}$ 

// Main simulation loop
finished := false
while finished = false do
  // Isolation costs
  for  $e \in E$  do
    if  $\mathcal{A}[e] = \text{false}$  then
       $N_{isol} := N_{isol} + 1$ 

  // Protective actions
   $S' := S$ 
  for  $v \in V$  s.t.  $S[v] = 'S'$  do
     $\phi(v) = \text{BFS}(G, S, v, h)$ 
     $\mathcal{P} := \Psi(\phi(v), x)$ 

    // Vaccination
    if  $\mathcal{P}$  is  $\mathcal{P}^{(vacc)}$  then
       $S'[v] := 'V'$ 
       $N_{vacc} := N_{vacc} + 1$ 

    // Isolation
    if  $\mathcal{P}$  is  $\mathcal{P}_q^{(isol)}$  then
      SetIsolationLevel( $q$ )

  // Spreading of the epidemic
  finished := true
   $S := S'$ 
  for  $v \in V$  s.t.  $S[v] = 'I'$  do
    for  $w \in V$  s.t.  $\langle v, w \rangle \in E$  and  $\mathcal{A}[\langle v, w \rangle] = \text{true}$  do
      finished := false
      if  $\text{Random}(U[0, 1]) < \beta$  then
         $S'[w] := 'I'$ 

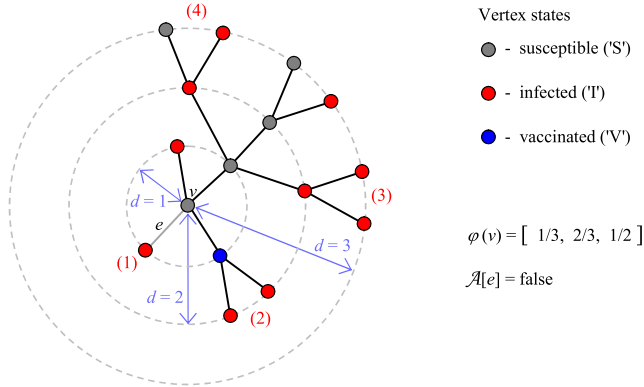
  // Recovery
  for  $v \in V$  s.t.  $S[v] = 'I'$  do
    if  $\text{Random}(U[0, 1]) < \gamma$  then
       $S'[v] := 'R'$ 

   $S := S'$ 

for  $v \in V$  s.t.  $S[v] = 'I'$  or  $S[v] = 'R'$  do
   $N_{inf} := N_{inf} + 1$ 

```

---



**Fig. 1.** An example of the calculation of  $\phi(v)$  for  $h = 3$ . See the description in the text.

of initially infected vertices (1%) was chosen in order to simulate a typical scenario in which an epidemic starts from a small group of infected individuals. The transmission probability  $\beta = 0.5$  is a value based on the literature [4]. From the properties of the geometric distribution [12] (which gives the probability that the first occurrence of a success requires  $t_{inf}$  independent trials, each with success probability  $\gamma$ ) it follows that  $\gamma = 0.1$  translates to the expected duration of the infection of  $t_{inf} = 10$  time steps. Thus, the value  $\gamma = 0.1$  ensures, that an infected individual remains infected long enough to spread the disease.

### 3.1 REDS Graphs

Each optimization problem instance is based on a graph, so it is necessary to decide what type of graphs to use, and select the number of vertices. In this paper REDS graphs were used [1]. In REDS graphs the vertices are placed on the unit square  $[0, 1] \times [0, 1]$  and the generation of edges is controlled by three parameters:  $R$ ,  $E$ , and  $S$ . The radius  $R$  determines the maximum distance at which the addition of a new edge is possible. Social energy  $E$  imposes a limit on how many connections a vertex can make (each edge costs  $D$  which is equal to this edge's length). The cost of an edge between vertices  $v_i$  and  $v_j$  is discounted by the factor of  $\frac{1}{1+S k_{ij}}$ , where  $k$  is the number of common neighbours the vertices  $v_i$  and  $v_j$  have. Because common neighbours cause the cost of creating new edges to be discounted, REDS graphs show a structure similar to a real-life social network with tightly connected groups separated by less crowded spaces. Because of varying density of edges, in REDS graphs communities are formed, even if the vertices are uniformly placed on the unit square. The instances used in the experiments described in this paper were generated using graph parameters shown in Table 2. The last column shows the average vertex degree  $\bar{k}$  which is not adjustable and was calculated from the graphs that were generated using the remaining parameters.



**Table 2.** Parameters of graphs on which test instances were based.

$N_v$	$R$	$E$	$S$	$\bar{k}$
1000	0.1000	0.15	0.5000	7.35
1250	0.0890	0.15	0.4470	7.97
1500	0.0820	0.15	0.4080	8.11
1750	0.0760	0.15	0.3780	8.57
2000	0.0700	0.15	0.3500	9.16
2250	0.0670	0.15	0.3330	9.43
2500	0.0630	0.15	0.3160	10.09

### 3.2 Decision Models

As described in Sect. 2 the optimization algorithm searches for Pareto-optimal vectors of parameters  $x \in \mathbb{R}^k$  which are subsequently used by a decision model  $\Psi$  to make decisions about which protective action to apply. In this paper two different models were tested: a *rule-based model* and a *neural model*. Both models take as input the vector  $\phi(v) \in \mathbb{R}^h$  which contains fractions of infected vertices around a vertex  $v$ , thereby representing the information about nearby cases of the disease. The set of protective actions used in this paper is  $\mathbb{P} = \{\mathcal{P}^{(none)}, \mathcal{P}^{(vacc)}, \mathcal{P}_{0.25}^{(isol)}, \mathcal{P}_{0.50}^{(isol)}, \mathcal{P}_{0.75}^{(isol)}, \mathcal{P}_{1.00}^{(isol)}\}$ . Therefore, the model can decide to take no action, vaccinate the vertex  $v$ , or to apply one of four levels of isolation, ranging from breaking  $\frac{1}{4}$  of contacts, up to a total isolation. From the machine learning perspective, the model  $\Psi$  is a classifier  $\Psi : \mathbb{R}^k \rightarrow \mathbb{P}$ .

#### Rule-based model

The rule-based model consists of five rules, one for each of the actions  $\mathcal{P}^{(vacc)}, \mathcal{P}_{0.25}^{(isol)}, \mathcal{P}_{0.50}^{(isol)}, \mathcal{P}_{0.75}^{(isol)}, \mathcal{P}_{1.00}^{(isol)}\}$ . The conditional part of the  $r$ -th rule ( $r = 1, \dots, 5$ ) is:

$$w_{r,1}\phi(v)[1] + w_{r,2}\phi(v)[2] + \dots + w_{r,h}\phi(v)[h] > \Theta_r, \tag{2}$$

where:

$r$  - the number of the rule,

$w_{r,d}$ , for  $d = 1, \dots, h$  - the weight assigned to the value of  $\phi(v)[d]$ , which contains the fraction of infected vertices at the distance of  $d$  edges from the vertex  $v$ ,

$h$  - the maximum distance (horizon radius) from  $v$  at which the fraction of infected vertices is calculated,

$\Theta_r$  - the threshold at which the rule activates.

For the rule-based model the number of parameters for one rule is  $h + 1$  ( $h$  weights and one threshold) and for all the rules it is  $k = 5(h + 1)$ . In the experiments the horizon was set to  $h = 3$  which resulted in the number of parameters for the rule-based model equal to  $k = 20$ . The rule-based model is

applied by calculating the left-hand sides of the rules (LHS) and comparing to the thresholds on the right-hand sides (RHS). The selected protective action is the first one for which  $LHS > RHS$  in the order of precedence:  $\mathcal{P}^{(vacc)}$ ,  $\mathcal{P}_{1.00}^{(isol)}$ ,  $\mathcal{P}_{0.75}^{(isol)}$ ,  $\mathcal{P}_{0.50}^{(isol)}$ ,  $\mathcal{P}_{0.25}^{(isol)}$ . If no rule activates the  $\mathcal{P}^{(none)}$  action is selected.

### Neural model

The neural model is a three-layer neural network [3]. The number of input neurons is  $N_{in} = h$ , the number of hidden neurons  $N_{hid}$  determines the size of the network, and the number of output neurons  $N_{out}$  has to be equal to the number of protective actions. The number of parameters is equal to the number of elements in weight matrices and bias vectors of the neural network  $k = N_{in} \cdot N_{hid} + N_{hid} + N_{hid} \cdot N_{out} + N_{out}$ . In the experiments the number of hidden neurons was set to  $N_{hid} = 5$ , so the number of parameters for the neural model was  $k = 3 \cdot 5 + 5 + 5 \cdot 6 + 6 = 56$ .

### 3.3 Evolutionary Algorithm

For optimization of parameters of the models the MOEA/D algorithm [17] with the Tchebycheff decomposition was used in which the objectives were normalized by dividing the value of the  $i$ -th objective by the difference between the worst and the best value of this objective in the population. The population consisted of real vectors in  $\mathbb{R}^k$  with  $k = 20$  for the rule-based model and  $k = 56$  for the neural model. The algorithm used four crossover operators: uniform, single-point, two-point and Simulated Binary Crossover (SBX) [9], and seven mutation operators: uniform, displacement, insertion, inversion, scramble, transpose and polynomial mutation [10]. The distribution index for the SBX and for the polynomial mutation was set to  $\eta = 20$ . For deciding which operator to use a mechanism for autoadaptation of operator probability based on success rates of the operators [22] was used. This approach was chosen following previous works on the Firefighter Problem [21, 22], which is also a graph-based problem in which vertices have to be protected from a spreading threat. The population size (which has to be a triangular number for the MOEA/D working on a three-objective problem) was set to  $N_{pop} = 300$  and the stopping criterion was the number of solutions evaluations  $max_{FE} = 10000$ . The neighbourhood size  $T$  and the probabilities of applying the operators  $P_{cross}$  and  $P_{mut}$  were tuned using the grid-search approach separately for rule-based and neural decision models, with the candidate values  $T \in \{20, 30, 40, 50\}$ ,  $P_{cross} \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ , and  $P_{mut} \in \{0.02, 0.04, 0.06, 0.08, 0.10\}$ . For the rule-based model the values  $T = 40$ ,  $P_{cross} = 1.0$ , and  $P_{mut} = 0.06$  were selected, and for the neural model the values  $T = 30$ ,  $P_{cross} = 0.6$ , and  $P_{mut} = 0.02$  were selected. The parameters were tuned on 30 optimization problem instances with  $|V| = 1000$  vertices, which were separate from the ones used in the rest of the experiments to avoid overfitting.

## 4 Results

In the experiments 30 runs of the evolutionary algorithm with each of the decision models described in Sect. 3.2 were performed for each graph size  $|V|$  ranging from 1000 to 2500. For each Pareto front produced by the optimization algorithm the value of the hypervolume indicator [28] was calculated. The hypervolume is often used in the literature to evaluate Pareto fronts, because, as shown by Fleischer [11], maximizing the hypervolume is equivalent to achieving Pareto optimality. From the 30 runs for each algorithm and each graph size  $|V|$  the median value was calculated. These median results are presented in Table 3.

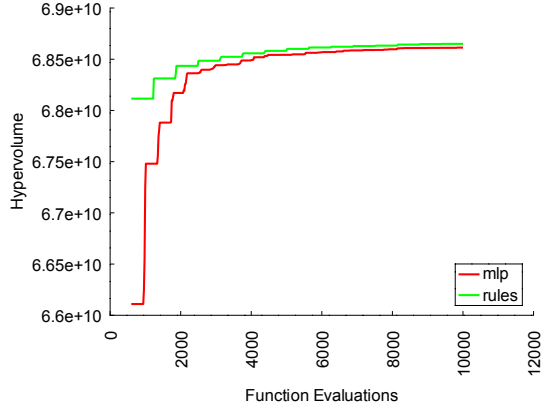
**Table 3.** Median hypervolume for the Pareto fronts produced by the tested methods obtained for  $max_{FE} = 10000$ . Better (larger) of the two values for a given  $|V|$  is underlined.

$ V $	HV: MLP	HV: Rules	p-value
1000	$6.861 \cdot 10^{10}$	<u><math>6.865 \cdot 10^{10}</math></u>	$6.32 \cdot 10^{-5}$
1250	$1.447 \cdot 10^{11}$	<u><math>1.448 \cdot 10^{11}</math></u>	$4.86 \cdot 10^{-5}$
1500	$2.482 \cdot 10^{11}$	<u><math>2.485 \cdot 10^{11}</math></u>	$5.79 \cdot 10^{-5}$
1750	$4.689 \cdot 10^{11}$	<u><math>4.693 \cdot 10^{11}</math></u>	$7.51 \cdot 10^{-5}$
2000	$7.279 \cdot 10^{11}$	<u><math>7.296 \cdot 10^{11}</math></u>	$7.69 \cdot 10^{-6}$
2250	$1.113 \cdot 10^{12}$	<u><math>1.115 \cdot 10^{12}</math></u>	$6.34 \cdot 10^{-6}$
2500	$1.652 \cdot 10^{12}$	<u><math>1.655 \cdot 10^{12}</math></u>	$8.92 \cdot 10^{-5}$

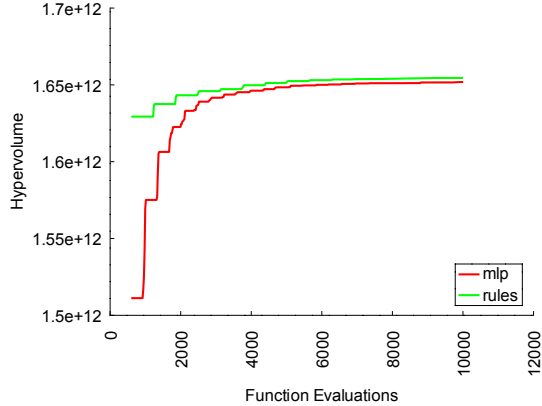
For each value of  $|V|$  a Wilcoxon statistical test was performed in order to verify statistical significance of the results. The null hypothesis of the Wilcoxon test states the equality of medians and thus low p-values (here, all below  $10^{-4}$ ) signify that the median hypervolume values are significantly different. The Family-Wise Error Rate (FWER) calculated as  $1 - \prod_1^7 (1 - p_i)$  for the tests performed for all the graph sizes  $|V|$  from the p-values shown in Table 3 is  $FWER = 0.00034798$ . Presented results show that the rule-based model outperformed the neural model for all the tested graph sizes  $|V|$ .

Another comparison was performed by plotting the median hypervolume calculated from the 30 runs with respect to the number of solution evaluations. Values obtained for  $|V| = 1000$  are presented in Fig. 2 and for  $|V| = 2500$  in Fig. 3. Plots presented in Figs. 2 and 3 show that the rule-based model performs better than the neural model even for small number of solution evaluations.

Below, an example of the rules is given, which produced the best results with respect to the  $N_{inf}$  criterion (minimizing the number of infected vertices) for  $|V| = 1000$ .



**Fig. 2.** The median hypervolume calculated from the 30 runs with respect to the number of solution evaluations for  $|V| = 1000$ .



**Fig. 3.** The median hypervolume calculated from the 30 runs with respect to the number of solution evaluations for  $|V| = 2500$ .

**if**  $0.966\phi(v)[1] + 0.252\phi(v)[2] + 0.242\phi(v)[3] > 0.045$  **then**  $\mathcal{P}^{(vacc)}(v)$   
**if**  $0.489\phi(v)[1] + 0.829\phi(v)[2] + 0.046\phi(v)[3] > 0.202$  **then**  $\mathcal{P}_{0.25}^{(isol)}(v)$   
**if**  $0.180\phi(v)[1] + 0.969\phi(v)[2] + 0.317\phi(v)[3] > 0.603$  **then**  $\mathcal{P}_{0.50}^{(isol)}(v)$   
**if**  $0.671\phi(v)[1] + 0.148\phi(v)[2] + 0.313\phi(v)[3] > 0.804$  **then**  $\mathcal{P}_{0.75}^{(isol)}(v)$   
**if**  $0.987\phi(v)[1] + 0.150\phi(v)[2] + 0.136\phi(v)[3] > 0.044$  **then**  $\mathcal{P}_{1.00}^{(isol)}(v)$

It can be observed that for decisions concerning vaccination of vertex  $v$  the most important is the number of infected vertices adjacent to  $v$  ( $d = 1$ ). This can be explained by the fact, that in the studied epidemic model vaccinations are immediately effective. Similarly, the closest contacts are the most important when decisions concern the introduction of the quarantine (protective action  $\mathcal{P}_{1.00}^{(isol)}(v)$  which cuts off all the edges adjacent to  $v$ ).

## 5 Conclusion

In this paper evolutionary optimization of counter-epidemic strategies was studied. The optimization problem tackled in this paper involves vaccinating vertices and/or inactivating edges in the graph, thereby putting vertices in isolation. Instead of directly working on a set (V+E) of both vertices and edges and making individual decisions the counter-epidemic strategy is based on a decision model that selects protective actions. The evolutionary algorithm optimizes the parameters of this decision model which is subsequently used for deciding whether to vaccinate a vertex or to use isolation instead (and if so, what fraction of the edges adjacent to this vertex to inactivate). In the paper two decision models were tested: a rule-based one and a neural one. The rule-based model outperformed the neural one in tests on optimization problem instances based on REDS graphs with the number of vertices  $|V|$  ranging from 1000 to 2500.

The optimization of decision models was studied with the models optimized for each particular problem instance. The models used in the paper are machine learning models and can be expected to show the generalization ability, that is to solve new problem instances when trained on other problem instances. Therefore, further work can be directed towards utilizing this generalization ability, for example by training the models on some problem instances and reusing them on other, possibly larger, problem instances. Another possibility could be to study the influence of the graph parameters (the  $R$ ,  $E$ , and  $S$  parameters for the REDS graphs) on the quality of the results attained by the models.

**Acknowledgment.** This work was supported by the Polish National Science Centre under grant no. 2015/19/D/HS4/02574. Calculations have been carried out using resources provided by Wroclaw Centre for Networking and Supercomputing (<http://wcss.pl>), grant No. 407.

## References





1. Antonioni, A., Bullock, S., Tomassini, M.: REDS: an energy-constrained spatial social network model. In: Lipson, H., Sayama, H., Rieffel, J., Risi, S., Doursat, R. (eds.) ALIFE 14: The Fourteenth International Conference on the Synthesis and Simulation of Living Systems. MIT Press (2014)
2. Backer, J., Hagens, T., Nodelijk, G., van Roermund, H.: Vaccination against foot-and-mouth disease I: epidemiological consequences. *Prev. Vet. Med.* **107**(1), 27–40 (2012)
3. Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford (1995)
4. Britton, T., Janson, S., Martin-Löf, A.: Graphs with specified degree distributions, simple epidemics, and local vaccination strategies. *Adv. Appl. Probab.* **39**(4), 922–948 (2007)
5. Burkholz, R., Leduc, M., Garas, A., Schweitzer, F.: Systemic risk in multiplex networks with asymmetric coupling and threshold feedback. *Physica D* **323–324**, 64–72 (2016)

6. Chekuri, C., Gupta, A., Kumar, A., Naor, J., Raz, D.: Building edge-failure resilient networks. *Algorithmica* **43**(1), 17–41 (2005)
7. Chen, C., et al.: Node immunization on large graphs: theory and algorithms. *IEEE Trans. Knowl. Data Eng.* **28**(1), 113–126 (2016)
8. Chen, R.L.Y., Phillips, C.A.: k-edge failure resilient network design. *Electron. Notes Discrete Math.* **41**, 375–382 (2013)
9. Deb, K., Agarwal, R.: Simulated binary crossover for continuous search space. *Complex Syst.* **9**(2), 115–148 (1995)
10. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. *Comput. Sci. Inf.* **26**, 30–45 (1996)
11. Fleischer, M.: The measure of pareto optima. Applications to multi-objective metaheuristics. In: *Second International Conference on Evolutionary Multi-Criterion Optimization, EMO 2003*, pp. 519–533. Springer, Heidelberg (2003)
12. Forbes, C., Evans, M., Hastings, N., Peacock, B.: Geometric distribution. In: *Statistical Distributions*, Chap. 23, pp. 114–116. Wiley (2010)
13. Glass, R.J., Glass, L.M., Beyeler, W.E., Min, H.J.: Targeted social distancing design for pandemic influenza. *Emerg. Infect. Dis.* **12**(11), 1671–1681 (2006)
14. Hartnell, B.: Firefighter! An application of domination. In: *20th Conference on Numerical Mathematics and Computing* (1995)
15. Kozen, D.C.: Depth-first and breadth-first search. In: *The Design and Analysis of Algorithms*, pp. 19–24. Springer, New York (1992)
16. Ladley, D.: Contagion and risk-sharing on the inter-bank market. *J. Econ. Dyn. Control* **37**(7), 1384–1400 (2013)
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. Martcheva, M.: Introduction to epidemic modeling. In: *An Introduction to Mathematical Epidemiology*, Texts in Applied Mathematics, vol. 61, pp. 9–31. Springer Science+Business Media, New York (2015)
19. Matthews, L.R., Gounaris, C.E., Kevrekidis, I.G.: Designing networks with resiliency to edge failures using two-stage robust optimization. *Eur. J. Oper. Res.* **279**(3), 704–720 (2019)
20. Maulana, A., Kefalas, M., Emmerich, M.T.M.: Immunization of networks using genetic algorithms and multiobjective metaheuristics. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8. IEEE (2017)
21. Michalak, K.: Auto-adaptation of genetic operators for multi-objective optimization in the firefighter problem. In: Corchado, E., Lozano, J.A., Quintián, H., Yin, H. (eds.) *IDEAL 2014*. LNCS, vol. 8669, pp. 484–491. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10840-7\\_58](https://doi.org/10.1007/978-3-319-10840-7_58)
22. Michalak, K.: The Sim-EA algorithm with operator autoadaptation for the multiobjective firefighter problem. In: Ochoa, G., Chicano, F. (eds.) *EvoCOP 2015*. LNCS, vol. 9026, pp. 184–196. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-16468-7\\_16](https://doi.org/10.1007/978-3-319-16468-7_16)
23. Michalak, K.: Reducing systemic risk in multiplex networks using evolutionary optimization. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2017*, pp. 289–290. ACM, New York (2017)
24. Michalak, K.: Surrogate-based optimization for reduction of contagion susceptibility in financial systems. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019*, pp. 1266–1274. ACM, New York (2019)

25. Michalak, K., Knowles, J.D.: Simheuristics for the multiobjective nondeterministic firefighter problem in a time-constrained setting. In: Squillero, G., Burelli, P. (eds.) *EvoApplications 2016*. LNCS, vol. 9598, pp. 248–265. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-31153-1\\_17](https://doi.org/10.1007/978-3-319-31153-1_17)
26. Pastor-Satorras, R., Castellano, C., Van Mieghem, P., Vespignani, A.: Epidemic processes in complex networks. *Rev. Mod. Phys.* **87**, 925–979 (2015)
27. Tornatore, E., Vetro, P., Buccellato, S.M.: SIVR epidemic model with stochastic perturbation. *Neural Comput. Appl.* **24**(2), 309–315 (2012). <https://doi.org/10.1007/s00521-012-1225-6>
28. 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**, 117–132 (2002)



# Human-Derived Heuristic Enhancement of an Evolutionary Algorithm for the 2D Bin-Packing Problem

Nicholas Ross<sup>1</sup>  , Ed Keedwell<sup>1</sup> , and Dragan Savic<sup>1,2</sup> 

<sup>1</sup> University of Exeter, Exeter EX4 4QF, UK  
nr339@exeter.ac.uk

<sup>2</sup> KWR, Nieuwegein, Netherlands

**Abstract.** The 2D Bin-Packing Problem (2DBPP) is an NP-Hard combinatorial optimisation problem with many real-world analogues. Fully deterministic methods such as the well-known Best Fit and First Fit heuristics, stochastic methods such as Evolutionary Algorithms (EAs), and hybrid EAs that combine the deterministic and stochastic approaches have all been applied to the problem. Combining derived human expertise with a hybrid EA offers another potential approach. In this work, the moves of humans playing a gamified version of the 2DBPP were recorded and four different Human-Derived Heuristics (HDHs) were created by learning the underlying heuristics employed by those players. Each HDH used a decision tree in place of the mutation operator in the EA. To test their effectiveness, these were compared against hybrid EAs utilising Best Fit or First Fit heuristics as well as a standard EA using a random swap mutation modified with a Next Fit heuristic if the mutation was infeasible. The HDHs were shown to outperform the standard EA and were faster to converge than – but ultimately outperformed by – the First Fit and Best Fit heuristics. This shows that humans can create competitive heuristics through gameplay and helps to understand the role that heuristics can play in stochastic search.

**Keywords:** Genetic Algorithms · Heuristics · Hybridization

## 1 Introduction

### 1.1 Background

There are many real-world cutting and packing problems that have been translated into operational research problems in order to find better solutions. One such problem is the two-dimensional finite bin-packing problem (2DBPP) [1]. This problem requires that a selection of boxes of assorted size are fit into the least number of identically finite-sized bins. Boxes and bins are sized in two dimensions, the boxes may not be cut or overlapped, and the bin's fixed capacity may not be exceeded. Most versions of the problem start with empty bins and the ability to add additional bins as needed. The simplest solution (but least efficient) would be to place every box in a new bin. More effective heuristics



have been developed from other deterministic approaches, such as First Fit, Next Fit, and Best Fit amongst many others [2].

First Fit is perhaps the simplest of these heuristics, in which the selected box is simply placed into the first bin in which it fits. Both First Fit and First Fit Decreasing (in which the boxes are sorted by size before placement) have been found to be competitive approaches to solving the problem [3]. Next Fit functions the same way as First Fit, except that the heuristic starts where the previous iteration finished i.e. if the heuristic places a box in the sixth bin, then the heuristic would start by looking in the seventh bin for a place to put the next box.

The Best Fit heuristic is another competitive approach to the problem [4]. This heuristic searches through all bins to place the box in the bin where the space remaining most closely matches the dimensions of the selected box without violating the bin capacity in either dimension. Other competitive heuristics have been developed by researchers such as the adaptive sequence-based heuristic of Oliveira & Gamboa [5] and the two-dimensional version of the Djang and Finch heuristic developed by López-Camacho et al. [6].

Stochastic methods such as Genetic Algorithms (GAs) have also been applied to the 2DBPP and other bin-packing problems. However, as general-purpose algorithms they struggle to be as competitive as the simpler deterministic heuristic techniques [7, 8]. Grouping Genetic Algorithms [9] applied to the simpler one-dimensional bin-packing problem outperformed the regular GA, while other researchers have used Multi-Objective techniques in a generalised framework to more easily compare against other heuristics and allow the possibility of combining techniques [10].

Combining the deterministic and stochastic approaches into a hybrid EA, hyper-heuristic, or other hybrid heuristic has proven to be a very effective approach. The many different hybrid approaches taken to tackle the 2DBPP include combining the GRASP and VND algorithms [11], combining iterative simulated annealing with binary search [12], combining an improved heuristic with the Variable Neighbourhood Search algorithm [13], combining chaos search with a firefly algorithm [14], and using adversarial self-play for reinforcement learning making use of neural nets and Markov Decision Processes [15].

Hyper-heuristics offer another way of easily combining search-based methods, heuristics, algorithms, and metaheuristics [16]. The work of López-Camacho et al. [17] directly combines a selection of deterministic methods such as Best Fit and First Fit with a GA, while other researchers make use of multi-objective EAs [18] or use an automated approach to design hybrid metaheuristics with a GA [19].

A hybrid EA was formed in Blum & Schmid [20] by combining an EA with a randomised one-pass heuristic, while hybrid GAs have been created by combining a GA with the Best Fit Decreasing heuristic [21], multiple local search heuristics [22], the Crow Search Algorithm [23], or Human-Derived Heuristics (HDHs) [24, 25], all with promising results.

In Ross et al. [25], participants played a gamified version of the 2DBPP and their moves were recorded. Machine learning was then applied to this dataset to obtain decision tree regressor models which provided the HDH. While heuristics are normally either “rules of thumb” employed by those with domain-specific experience or algorithms

built from theory-backed research, HDH are created by learning from how a human interacts with a problem.

## 1.2 Proposed Approach

Several different HDHs were derived from experimental data and each of them combined in turn to form a hybrid GA. These would be compared with a standard GA and hybrid GAs making use of either the Best Fit or First Fit heuristics. Each HDH would take the form of a decision tree obtained by machine learning on subsets of the data. Four different subsets of data were selected based on the moves made by the humans solving the problem during bin-packing gameplay.

The first and most obvious move set to learn from consisted of All Moves in the data set ( $\text{HDH}_{\text{ALL}}$ ). This heuristic learns from moves that improved, worsened, or left the solution unchanged, and it could reasonably be expected to help find good solutions but might take more iterations to do so.

Taking a greedy approach, the second move set to learn from consists of only moves that improved the solution. The Improving Moves heuristic ( $\text{HDH}_{\text{IMP}}$ ) should converge faster than the  $\text{HDH}_{\text{ALL}}$  heuristic, though there is a greater danger of getting stuck in a local optimum.

To test the hypothesis that the selection of learning moves has an influence on the performance of the different hybrid GAs, the third set of moves that were learned from consists of only moves that make the solution worse. The Deteriorating Moves heuristic ( $\text{HDH}_{\text{DET}}$ ) is expected to be outperformed by the other HDHs, and possibly by the standard GA as well.

The last approach is based on Composite Moves ( $\text{HDH}_{\text{COM}}$ ), which are moves that make the solution worse followed by moves that improve the solution. This heuristic should perform similarly to the  $\text{HDH}_{\text{IMP}}$  heuristic, but with a reduced chance of getting stuck in a local optimum. Each of these heuristics will take the place of the mutation operator in the hybrid GA.

These will be compared against hybrid GAs making use of First Fit heuristics or Best Fit heuristics in place of their mutation operators, and a standard GA that uses a feasible-only random swap mutation based on Next Fit. This latter mutation is necessary to give the standard GA a fair chance when competing with the other heuristics, as all the others will only create feasible mutations. This mutation operates by initially attempting a random swap mutation, but if that would make the solution infeasible then it will try to fit the box into the next bin and repeat until it succeeds.

In Ross et al. [25] it was found that better results were obtained by combining the standard mutation with the HDH mutation in the same algorithm. For this experiment several different proportions of the standard mutation are tested with the HDH, First Fit, and Best Fit heuristics. Each hybrid GA will be tested with proportions of 100%, 99%, 40%, 10%, and 1% heuristic mutation with the balance made up of the standard mutation.

## 2 Experimental and Computational Details

### 2.1 Gamification

The 2DBPP problem used in this paper has a fixed number of finite bins and begins with the boxes already randomly distributed between them. The problem contains 10 bins and 20 boxes. The 20 boxes were created by splitting 5 bins, meaning the global optimum is reached by fitting all 20 boxes into just 5 bins. No new bins can be added, and empty bins are not removed.

To achieve a finer detail of scoring than counting the number of used bins, the problem was instead scored by giving a maximum score for each completely empty or exactly full bin, and adding a score for each other bin based on its closeness to being full or empty. The score for each bin was calculated as follows:

$$\begin{aligned} \text{If Bin} = \text{Empty or Bin} = \text{Full, } \text{Score} &= \text{Dim1Max} + \text{Dim2Max} \\ \text{Else, } \text{Score} &= \left| \frac{\text{Dim1Max}}{2} - \text{BinDim1} \right| + \left| \frac{\text{Dim2Max}}{2} - \text{BinDim2} \right| \end{aligned} \quad (1)$$

Where *Dim1Max* and *Dim2Max* are the maximum size respectively of the first and second dimensions of the bins, and *BinDim1* and *BinDim2* represent the current filled proportion of the bin in the first and second dimensions respectively. In the problem used both *Dim1Max* and *Dim2Max* were set to the same value of 500, meaning that each empty or full bin scored 1000 and the maximum score was equal to the number of bins in the problem multiplied by 1000. With 10 bins this gives a global optimum score of 10,000. Infeasible solutions were not scored, but their total capacity violation was recorded for the sake of the fitness function.

The data used in this study was obtained through the gamification of a bin-packing problem. Gamification is the process of turning something into a game or making use of game-like features such as scoring and victory conditions. It has been successfully used to keep player attention and focus on mundane or repetitive tasks [26–28].

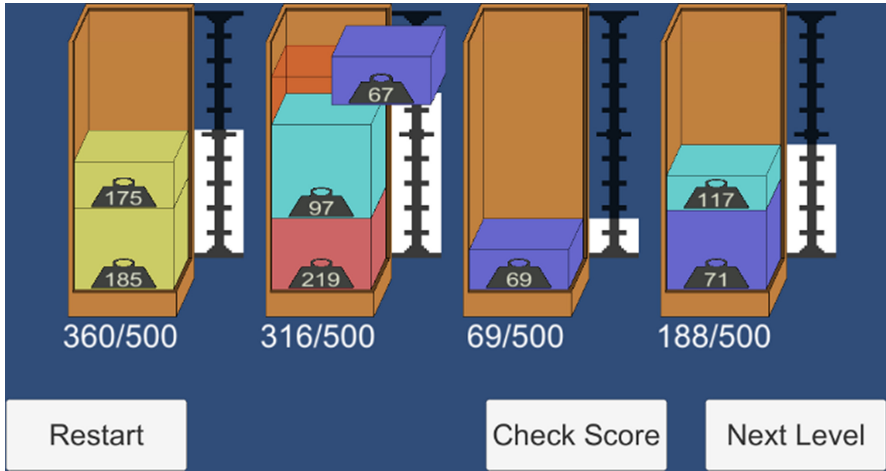
In this case a game was created from a 2DBPP which participants played while their moves were recorded. The game is described in more detail in [25], but essentially consisted of a simple problem with 4 bins and 8 boxes (Fig. 1).

The players were told the objective and shown how the game worked and were then encouraged to compete against each other in solving the problem in the least number of moves. Each move would see the player select a single box in the problem and move it to whichever bin they chose, after which their score would be updated.

Data was gathered from every player that finished the game and solved the problem, regardless of individual performance or the number of moves taken to do so. A total of 10 players completed the simple 4-bin problem. This game data was passed through a method that selected moves for each data set for the machine learning, based on which heuristic was being implemented. This created a data set for each of the four HDHs.

### 2.2 Deriving Human Heuristics

Every move that a human player made could be split into two parts; the selection of the box to move and the selection of the bin to place it in. As the First Fit and Best Fit



**Fig. 1.** The bin-packing game being played

heuristics have no set box selection method but do have a deterministic bin selection, the fairest comparison was to leave the box selection as random for all heuristics and just investigate the moves with regards to target bin selection. This fixed the machine learning output as a quality of the target bin, which was best represented by the amount of space remaining in that bin.

A key step in the process is the determination of the input variables for use in the machine learning approach. The size of the selected box was an obvious variable, and then a number of more general inputs that described the problem space could be included. These would allow the heuristics to be somewhat generalisable across different bin-packing and related problems. The preliminary experimentation, a total of four inputs were selected, each as a sum of the two dimensions.

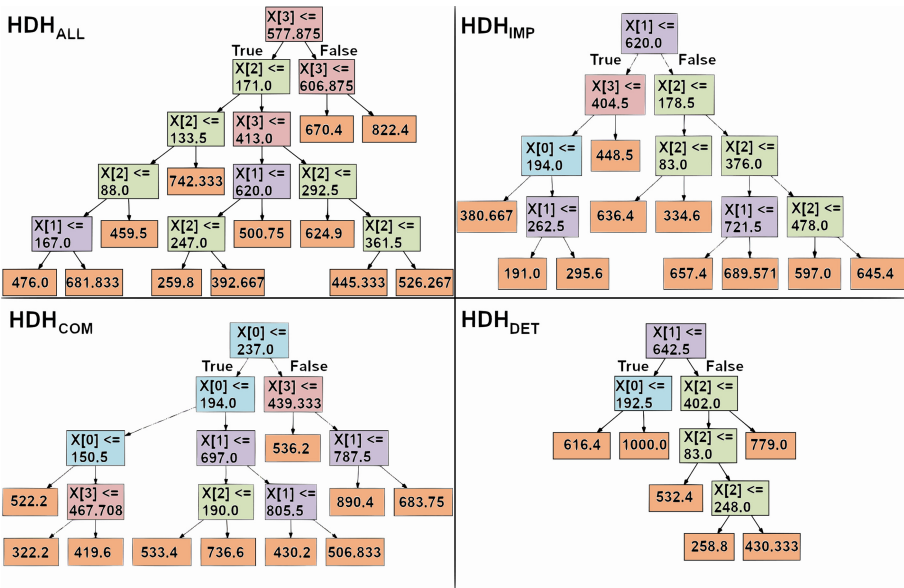
The first input was the size of the box that the player had selected ( $X[0]$ ). The second input was the maximum available space remaining in any non-empty bin ( $X[1]$ ). The third input was the minimum available space in any non-full bin ( $X[2]$ ), and the last input was the mean space remaining across all partially full bins ( $X[3]$ ). The output was the available space remaining in the target bin (Table 1).

**Table 1.** Inputs and output for the machine learning. The colour of each input and output matches the colours of their respective nodes in the decision trees.

Inputs				Output
$X[0]$ : Size of Selected Box	$X[1]$ : Maximum Remaining Bin Space	$X[2]$ : Minimum Remaining Bin Space	$X[3]$ : Mean Remaining Bin Space	Target Remaining Bin Space

Machine learning was carried out in Python using *Scikit-learn* [29]. Decision trees were chosen as they give a human-readable insight into the workings of the heuristic. For each HDH a data set with the selected moves was loaded in and an *sklearn* decision tree regressor was trained. To aid readability, each tree was constrained to a minimum leaf node size of 5, a maximum of 12 leaf nodes, and a maximum depth of 6.

The resultant decision trees are shown in Fig. 2. The HDH<sub>ALL</sub> decision tree that learned from all the moves in the data set is unique in not using the box size (X[0]) input in any of its calculations. The HDH<sub>IMP</sub> decision tree makes less use of the mean space remaining input (X[3]) than HDH<sub>ALL</sub>, while the HDH<sub>DET</sub> decision tree doesn't use it at all. The HDH<sub>DET</sub> tree is also the only tree that will seek out completely empty bins.



**Fig. 2.** The decision trees generated for the four different Human-Derived Heuristics. The box colours correspond to the inputs in Table 1.

The last tree generated was that for the HDH<sub>COM</sub> heuristic, which makes more use of the box size input (X[0]) and less use of the minimum bin space remaining input (X[2]) than the others. All four decision trees were exported into a program to be used as mutation operators in a GA.

### 2.3 Experimental Setup

The GA used for this experiment was a steady-state GA. The problem was represented using *k*-ary encoding with a *k* of 10 and a length equal to 20, the number of boxes in the problem. Each integer value in the encoding represented which bin the box at that index position was in.

The parameters and crossover type were tuned for the standard GA, without using the heuristics. From this initial testing a population size of 100 and a tournament selection method with a tournament size of 2 was chosen. Uniform crossover performed the best, so it was selected, along with a mutation rate of 0.1. As the GA was steady-state, two children would be created and added to the population each generation, and the two least fit members of the population would then be removed.

The fitness function scored both *feasible* solutions and *infeasible* solutions. This worked by first checking if the selected solution exceeded bin capacity and was therefore infeasible; if it was infeasible it would be scored 0 for fitness and then each infeasible bin would be scored based on how much it exceeded the bin capacity limit and added to a violation score. If the solution was feasible it would be scored as mentioned in the introduction in Eq. 1 (with an optimum score of 10,000).

The mutation selected boxes from the child problems of the crossover with a probability of 0.1. For the standard GA the mutation moved the selected box to a bin at random, but if that made that bin infeasible it would then attempt to place the box in the next bin along instead. This would be repeated as needed, looping back round to the start of the bins until the mutation found an appropriately sized bin. This approach was adopted to provide a fairer benchmark for standard mutation. The human players were not permitted to make moves that resulted in infeasible solutions and this mutation operator performs the same function for the standard GA.

For the HDH mutation the decision tree determines the bin into which the randomly selected box should be placed, based on the closest match to the determined space remaining. For First Fit and Best Fit mutations their respective heuristics were applied, with First Fit searching from the start until it found the first bin that could fit the selected box, and Best Fit searched the entire problem space for the bin that had the closest space remaining to match the box size. The hybrid GA mutation had a chance of implementing either the heuristic mutation or the standard mutation.

The GA for each condition was run for 200,000 iterations on 30 different instances of the 10-bin problem, with each different problem instance being repeated 30 times for a total of 900 runs per condition.

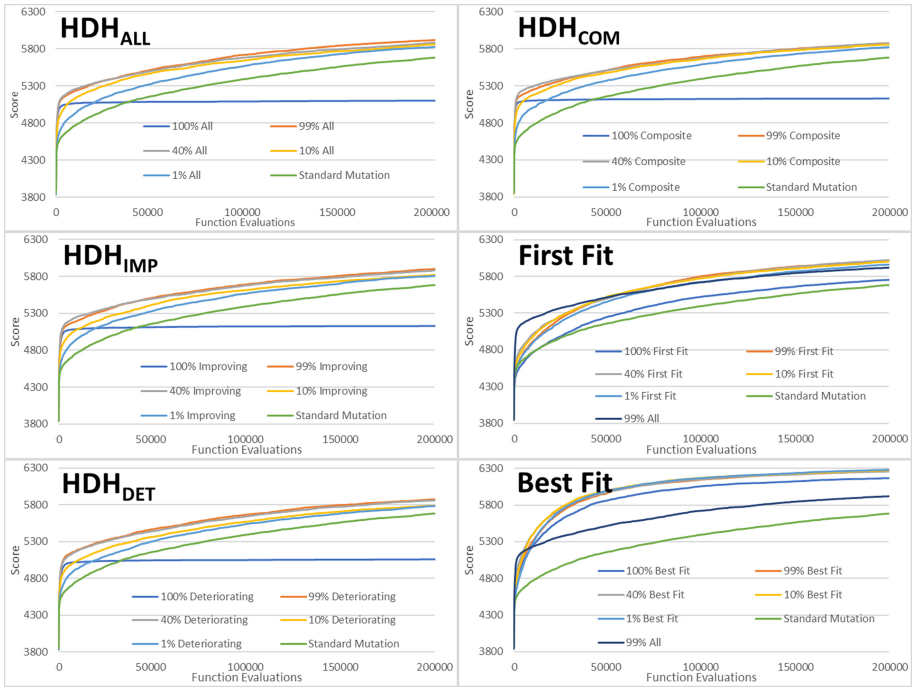
### 3 Results and Discussion

The fittest result in the population at each iteration of the GA was recorded and then averaged across all 900 runs. These mean fittest values were then plotted for each HDH against the values from the standard GA (Fig. 3).

#### 3.1 Human-Derived Heuristics vs Standard GA

##### 3.1.1 HDH<sub>ALL</sub>

100% HDH<sub>ALL</sub> converged early to a local optimum and failed to progress further, being quickly overtaken by the other percentages and the standard GA. 99% HDH<sub>ALL</sub> achieved the highest fitness by the end of the run and showed faster convergence than the standard GA, with 40% HDH<sub>ALL</sub> following a similar pattern. 10% HDH<sub>ALL</sub> was slower to converge but ended close behind 40% HDH<sub>ALL</sub> and still ahead of the standard GA.



**Fig. 3.** Mean fittest solution per generation for each condition.

1% HDH<sub>ALL</sub> outperformed the standard GA and ended at a similar level of fitness as the other percentages, though it converged slower.

### 3.1.2 HDH<sub>IMP</sub>

The 100% HDH<sub>IMP</sub> condition quickly converged to a suboptimal solution and was outperformed by the other conditions including the standard GA while the 99% HDH<sub>IMP</sub> and 40% HDH<sub>IMP</sub> conditions converged fastest and ended with the fittest solutions. 10% HDH<sub>IMP</sub> and 1% HDH<sub>IMP</sub> performed similarly, though 10% HDH<sub>IMP</sub> converged the faster of the two. All HDH<sub>IMP</sub> heuristics except for 100% HDH<sub>IMP</sub> outperformed the standard GA.

### 3.1.3 HDH<sub>DET</sub>

Though the 100% HDH<sub>DET</sub> converged quickly it was still outperformed by the standard GA. All other HDH<sub>DET</sub> conditions managed to outperform the standard GA, with the 99% HDH<sub>DET</sub> heuristic performing best with 40% HDH<sub>DET</sub> a close second. The 10% HDH<sub>DET</sub> and 1% HDH<sub>DET</sub> conditions performed at an intermediate level between the standard GA and the 40% and 99% HDH<sub>DET</sub> heuristics, with the 10% condition performing slightly better.

### 3.1.4 HDH<sub>COM</sub>

The 100% HDH<sub>COM</sub> heuristic also converged very early to a local optimum and then failed to progress further. The 99% HDH<sub>COM</sub>, 40% HDH<sub>COM</sub>, and 10% HDH<sub>COM</sub> performed almost identically, except that the 40% HDH<sub>COM</sub> heuristic converged faster and the 10% HDH<sub>COM</sub> heuristic converged slightly slower. The 1% HDH<sub>COM</sub> heuristic lagged slightly behind the others, but all HDH<sub>COM</sub> percentages except for 100% outperformed the standard GA.

The final fittest score results for all 900 runs of each HDH condition were then compared statistically against the standard GA. For every HDH percentage condition an F-Test was performed comparing the variance with the standard GA results, followed by a two-factor t-Test. These results can be found in Table 2.

**Table 2.** Mean and maximum fittest scores attained over the 900 runs. The mean values that differ significantly from the standard GA are denoted by asterisks (\*) and the highest value for mean and maximum fittest score are highlighted in bold and underlined.

		Mean fittest score	Statistical comparison vs Standard GA	Maximum fittest score
Standard GA		5683	N/A	7374
HDH <sub>ALL</sub>	100%	5102*	t(1762) = 24.0 p < .001	6516
	99%	<b><u>5920*</u></b>	t(1774) = -11.2 p < .001	7362
	40%	5886*	t(1789) = -9.4 p < .001	7718
	10%	5860*	t(1788) = -8.2 p < .001	7724
	1%	5827*	t(1792) = -6.6 p < .001	7310
HDH <sub>IMP</sub>	100%	5129*	t(1774) = 23.2 p < .001	6640
	99%	5901*	t(1792) = -10.0 p < .001	7574
	40%	5882*	t(1788) = -9.2 p < .001	7392
	10%	5822*	t(1792) = -6.4 p < .001	7186
	1%	5807*	t(1798) = -5.7 p < .001	7124
HDH <sub>COM</sub>	100%	5130*	t(1778) = 23.3 p < .001	6572
	99%	5876*	t(1798) = -8.8 p < .001	<b><u>8208</u></b>
	40%	5875*	t(1788) = -8.9 p < .001	7824
	10%	5859*	t(1785) = -8.2 p < .001	7238
	1%	5824*	t(1798) = -6.5 p < .001	7504
HDH <sub>DET</sub>	100%	5060*	t(1775) = 26.2 p < .001	6690
	99%	5876*	t(1780) = -9.0 p < .001	7706
	40%	5864*	t(1787) = -8.4 p < .001	7626
	10%	5788*	t(1798) = -4.8 p < .001	7734
	1%	5786*	t(1798) = -4.6 p < .001	7042



Every condition differed significantly from the standard GA, with the 100% HDH<sub>ALL</sub>, 100% HDH<sub>IMP</sub>, 100% HDH<sub>DET</sub>, and 100% HDH<sub>COM</sub> all performing significantly worse, and all other HDH percentages performing significantly better. 99% HDH<sub>ALL</sub> achieved the highest mean fittest score and 99% HDH<sub>COM</sub> found the highest maximum fittest score across all runs. Within each HDH condition the 99% heuristic reached the highest mean fittest score while the 100% heuristic performed the worst.

The 99% HDH<sub>ALL</sub> result did not perform significantly better than 40% HDH<sub>ALL</sub> ( $t(1798) = -1.6, p = .10$ ), but did perform significantly better than 10% HDH<sub>ALL</sub> ( $t(1798) = -3.0, p = .003$ ) and 1% HDH<sub>ALL</sub> ( $t(1792) = -4.5, p < .001$ ).

The 99% HDH<sub>IMP</sub> heuristic saw no significant difference compared against 40% HDH<sub>IMP</sub> ( $t(1798) = -0.9, p = .38$ ), but significantly outperformed the 10% HDH<sub>IMP</sub> ( $t(1798) = -3.7, p < .001$ ) and 1% HDH<sub>IMP</sub> heuristics ( $t(1798) = -4.4, p < .001$ ).

The 99% HDH<sub>DET</sub> heuristic saw no significant difference in performance against 40% HDH<sub>DET</sub> ( $t(1791) = -0.6, p = .55$ ) but performed significantly better than 10% HDH<sub>DET</sub> ( $t(1791) = -4.2, p < .001$ ) and 1% HDH<sub>DET</sub> ( $t(1785) = -4.2, p < .001$ ).

The 99% HDH<sub>COM</sub> heuristic did not significantly outperform either the 40% HDH<sub>COM</sub> ( $t(1798) = 0.04, p = .96$ ) or 10% HDH<sub>COM</sub> ( $t(1798) = 0.83, p = 0.41$ ), but performed significantly better than 1% HDH<sub>COM</sub> ( $t(1798) = 2.45, p = .01$ ).

Comparing the mean highest scoring heuristics from each condition against each other found the 99% HDH<sub>ALL</sub> to perform significantly better than the 99% HDH<sub>DET</sub> ( $t(1798) = -2.2, p = 0.03$ ) and 99% HDH<sub>COM</sub> ( $t(1787) = -2.1, p = 0.04$ ) heuristics but not the 99% HDH<sub>IMP</sub> ( $t(1792) = -0.9, p = .35$ ) heuristic.

### 3.2 Human-Derived Heuristics vs First Fit

The results for the First Fit heuristic can be seen in Fig. 3, plotted against the standard GA and 99% HDH<sub>ALL</sub> heuristic. All percentage conditions of First Fit performed significantly better than the standard GA (see Table 3), though the 100% First Fit heuristic did not perform as well as the others and was outperformed by 99% HDH<sub>ALL</sub>. The First Fit heuristic converged slower than 99% HDH<sub>ALL</sub>, but all except 100% First Fit eventually reached a higher fitness. 100% First Fit scored significantly less than 99% HDH<sub>ALL</sub>, but the others all scored significantly higher than 99% HDH<sub>ALL</sub> (Table 3).

### 3.3 Human-Derived Heuristics vs Best Fit

The results for the Best Fit heuristic can be seen in Fig. 3, with the standard GA and 99% HDH<sub>ALL</sub> heuristic for comparison. 100% Best Fit was the worst performing of them, though the Best Fit heuristics were faster to converge than the First Fit heuristics, and highest scoring of all the heuristics tested. Every Best Fit heuristic significantly outperformed both the standard GA and the 99% HDH<sub>ALL</sub> condition as well (see Table 3), though the Human-Derived Heuristic was faster to converge than the others.

### 3.4 Discussion

Although there were minor differences in performance between them, each HDH, regardless of the dataset source appeared to perform in a similar manner. In terms of application

**Table 3.** Mean and maximum fittest scores attained over the 900 runs. The mean values that differ significantly from the standard GA (\*) and 99% HDH<sub>ALL</sub> (†) are marked. Highest values for mean and maximum fittest score are highlighted in bold and underlined.

Mean fittest score			Statistical comparison		Maximum fittest score
			vs Standard GA	vs HDH <sub>ALL</sub>	
Standard GA		5683	N/A		7374
HDH <sub>ALL</sub>	99%	5920*	t(1774) = -11.2 p < .001	N/A	7362
First fit	100%	5754*†	t(1760) = -2.9 p = .003	t(1684) = -7.1, p < .001	8456
	99%	6011*†	t(1729) = -16.1 p < .001	t(1785) = 4.8, p < .001	7718
	40%	6022*†	t(1782) = -15.8 p < .001	t(1798) = 5.1, p < .001	8678
	10%	6004*†	t(1784) = -15.0 p < .001	t(1798) = 4.2, p < .001	8214
	1%	5961*†	t(1789) = -12.8 p < .001	t(1798) = 2.0, p = .04	7644
Best fit	100%	6165*†	t(1798) = -22.1 p < .001	t(1791) = 11.9, p < .001	<b><u>10000</u></b>
	99%	6268*†	t(1701) = -29.1 p < .001	t(1770) = 18.5, p < .001	8218
	40%	6256*†	t(1685) = -28.7 p < .001	t(1760) = 18.1, p < .001	8208
	10%	6271*†	t(1744) = -28.5 p < .001	t(1792) = 18.1, p < .001	<b><u>10000</u></b>
	1%	<b><u>6281</u></b> *†	t(1733) = -29.1 p < .001	t(1787) = 18.8, p < .001	<b><u>10000</u></b>

level, 100% HDH rapidly converged to a local optimum and was overtaken by the others as expected, with the 99% and 40% conditions performing the best followed by the 10% condition and finally the 1% condition.

The 99% condition showed that employing even a small amount of random mutation was enough to prevent the HDH getting stuck in a local optimum. Conversely, and surprisingly, the 1% condition showed that employing only a small amount of deterministic mutation was enough to significantly improve the standard GA.

The Deteriorating Moves (HDH<sub>DET</sub>) heuristic performed better than expected but still achieved the lowest mean fittest scores of the four HDHs. It is likely that amongst the moves that the decision tree learned were useful moves that made the solution temporarily worse but created an opening for better moves.

The Composite Moves ( $\text{HDH}_{\text{COM}}$ ) heuristic was the fastest to converge, but both the  $\text{HDH}_{\text{ALL}}$  and the  $\text{HDH}_{\text{IMP}}$  heuristics achieved better results (although the Composite Moves heuristic achieved the highest maximum score from any of the HDHs). The Composite Moves heuristic used in this paper was a single decision tree, but a true Composite Moves heuristic might be better represented by two trees; one tree to make the moves that make the solution temporarily worse and a second to improve it.

The Improving Moves heuristic performed almost as well as the All Moves heuristic, with very little difference between them except that the All Moves heuristic reaching a higher mean fittest fitness score by the end of the run.

When compared to the First Fit heuristic the HDHs performed competitively at the start of the run but were eventually outperformed by all but 100% First Fit. The 100% First Fit heuristic performed only a little better than the standard GA, and it was surprising to see the other First Fit percentages performing significantly better than it.

The Best Fit heuristics performed very well against the HDHs, though again the 100% Best Fit condition performed poorest. The Best Fit heuristics were the only ones to reach the global optimum, though on isolated runs and not for all conditions.

The 1% Best Fit heuristic had both the highest mean fittest score of any of the heuristics and found the global optimum. This result shows that just a small amount of a competitive deterministic heuristic can have a strong effect on a GA.

The only advantage enjoyed by the HDHs when compared to the Best Fit heuristic is a slightly faster convergence rate. This fast convergence could be useful if the HDH was combined with another heuristic or was incorporated into a hyper-heuristic that could take advantage of the different capabilities at its disposal.

Although HDHs were outperformed by the established heuristics, an area where HDHs would be useful is on problems that don't have existing established heuristics such as Best Fit. Learning a heuristic from human interactions with a previously unseen problem is easier than attempting to create new rules of thumb. Furthermore, many real-world problems will not have accompanying heuristics and so the HDH methodology might be used to create them.

Future work could see these heuristics tested on other problems, and other heuristics developed from similar problems compared against these. Combining several Human-Derived Heuristics and more traditional heuristics into a hyper-heuristic might yield even more promising results.

## 4 Conclusions

In this study machine learning was used on four different data sets to create four different Human-Derived Heuristics (HDHs). Each of them was developed from human players solving a gamified version of a small 2D bin-packing problem.

The four HDHs were then in turn combined with a hybrid GA as the mutation operator, with the GA utilising the HDH either 100%, 99%, 40%, 10%, or 1% of the time during the run and the remainder of the time using a random swap mutation modified with a Next Fit heuristic. The First Fit and Best Fit heuristics were then executed in the same process and the results compared.

For the HDHs the 100% heuristics performed poorly, but the other conditions all performed significantly better than the standard GA. Several of the HDHs also outperformed the 100% First Fit heuristic, but the other First Fit and all the Best Fit conditions were able to outperform the HDHs.

Surprisingly, using either 1% of a stochastic mutation or 1% of a deterministic mutation with 99% of the other resulted in better results than 100% of either alone.

**Acknowledgments.** This work was supported by Skipworth Engelhardt Asset Management Strategists Limited (SEAMS) and the Human-Computer Optimisation for Water Systems Planning and Management (HOWS) project funded by the Engineering and Physical Sciences Research Council (EPSRC) – grant EP/P009441/1.

## References

1. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. *Eur. J. Oper. Res.* **183**(3), 1109–1130 (2007)
2. Berkey, J.O., Wang, P.Y.: Two-dimensional finite bin-packing algorithms. *J. Oper. Res. Soc.* **38**(5), 423–429 (1987). <https://doi.org/10.1057/jors.1987.70>
3. Dósa, G., Sgall, J.: First fit bin packing: a tight analysis. In: 30th International Symposium on Theoretical Aspects of Computer Science (STACS 2013), Dagstuhl, Germany, 2013, vol. 20, pp. 538–549. <https://doi.org/10.4230/LIPIcs.STACS.2013.538>
4. Dósa, G., Sgall, J.: Optimal analysis of best fit bin packing. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) *ICALP 2014*. LNCS, vol. 8572, pp. 429–441. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-43948-7\\_36](https://doi.org/10.1007/978-3-662-43948-7_36)
5. Oliveira, Ó., Gamboa, D.: Adaptive sequence-based heuristic for the two-dimensional non-guillotine bin packing problem. In: Madureira, A.M., Abraham, A., Gandhi, N., Varela, M.L. (eds.) *HIS 2018*. AISC, vol. 923, pp. 370–375. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-14347-3\\_36](https://doi.org/10.1007/978-3-030-14347-3_36)
6. López-Camacho, E., Ochoa, G., Terashima-Marín, H., Burke, E.K.: An effective heuristic for the two-dimensional irregular bin packing problem. *Ann. Oper. Res.* **206**(1), 241–264 (2013). <https://doi.org/10.1007/s10479-013-1341-4>
7. Falkenauer, E., Delchambre, A.: A genetic algorithm for bin packing and line balancing. In: *Proceedings of 1992 IEEE International Conference on Robotics and Automation*, 1992, vol. 2, pp. 1186–1192. <https://doi.org/10.1109/robot.1992.220088>
8. Lam, G.T., Ho, V.A., Logofatu, D., Badica, C.: Considerations on using genetic algorithms for the 2D bin packing problem: a general model and detected difficulties. In: *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 303–308 (2017). <https://doi.org/10.1109/icstcc.2017.8107051>
9. Kucukyilmaz, T., Kiziloz, H.E.: Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem. *Comput. Ind. Eng.* **125**, 157–170 (2018). <https://doi.org/10.1016/j.cie.2018.08.021>
10. Luo, F., Scherson, I.D., Fuentes, J.: A novel genetic algorithm for bin packing problem in jMetal. In: *2017 IEEE International Conference on Cognitive Computing (ICCC)*, pp. 17–23 (2017). <https://doi.org/10.1109/iecc.2017.10>
11. Parreño, F., Alvarez-Valdes, R., Oliveira, J.F., Tamarit, J.M.: A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing. *Ann. Oper. Res.* **179**(1), 203–220 (2010). <https://doi.org/10.1007/s10479-008-0449-4>

12. Hong, S., Zhang, D., Lau, H.C., Zeng, X., Si, Y.-W.: A hybrid heuristic algorithm for the 2D variable-sized bin packing problem. *Eur. J. Oper. Res.* **238**(1), 95–103 (2014). <https://doi.org/10.1016/j.ejor.2014.03.049>
13. Zhang, D., Che, Y., Ye, F., Si, Y.-W., Leung, S.C.H.: A hybrid algorithm based on variable neighbourhood for the strip packing problem. *J. Comb. Optim.* **32**(2), 513–530 (2016). <https://doi.org/10.1007/s10878-016-0036-6>
14. Zhao, C., Jiang, L., Teo, K.L.: A hybrid chaos firefly algorithm for three-dimensional irregular packing problem. *J. Ind. Manag. Optim.* **16**(1), 409 (2020). <https://doi.org/10.3934/jimo.2018160>
15. Laterre, A., Fu, Y., Jabri, M.K., Cohen, A.-S., Kas, D., Hajjar, K.: Ranked reward: enabling self-play reinforcement learning for bin packing, p. 10 (2019)
16. Pillay, N., Qu, R.: Packing Problems. In: *Hyper-Heuristics: Theory and Applications*, pp. 67–73. Springer International Publishing, Cham (2018)
17. López-Camacho, E., Terashima-Marín, H., Ross, P.: A hyper-heuristic for solving one and two-dimensional bin packing problems. In: *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, Dublin, Ireland, pp. 257–258 (2011). <https://doi.org/10.1145/2001858.2002003>
18. Gomez, J.C., Terashima-Marín, H.: Evolutionary hyper-heuristics for tackling bi-objective 2D bin packing problems. *Genet. Program. Evolvable Mach.* **19**(1), 151–181 (2018). <https://doi.org/10.1007/s10710-017-9301-4>
19. Hassan, A., Pillay, N.: Hybrid metaheuristics: an automated approach. *Expert Syst. Appl.* **130**, 132–144 (2019). <https://doi.org/10.1016/j.eswa.2019.04.027>
20. Blum, C., Schmid, V.: Solving the 2D bin packing problem by means of a hybrid evolutionary algorithm. *Procedia Comput. Sci.* **18**, 899–908 (2013). <https://doi.org/10.1016/j.procs.2013.05.255>
21. Kaaouache, M.A., Bouamama, S.: Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud. *Procedia Comput. Sci.* **60**, 1061–1069 (2015). <https://doi.org/10.1016/j.procs.2015.08.151>
22. Beyaz, M., Dokeroglu, T., Cosar, A.: Hybrid heuristic algorithms for the multiobjective load balancing of 2D bin packing problems. In: Abdelrahman, O.H., Gelenbe, E., Gorbil, G., Lent, R. (eds.) *Information Sciences and Systems 2015. LNEE*, vol. 363, pp. 209–220. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-22635-4\\_19](https://doi.org/10.1007/978-3-319-22635-4_19)
23. Laabadi, S., Naimi, M., El Amri, H., Achchab, B.: A crow search-based genetic algorithm for solving two-dimensional bin packing problem. In: Benzmüller, C., Stuckenschmidt, H. (eds.) *KI 2019. LNCS (LNAI)*, vol. 11793, pp. 203–215. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-30179-8\\_17](https://doi.org/10.1007/978-3-030-30179-8_17)
24. Johns, M.B., Mahmoud, H.A., Walker, D.J., Ross, N.D.F., Keedwell, E.C., Savic, D.A.: Augmented evolutionary intelligence: combining human and evolutionary design for water distribution network optimisation. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, Prague, Czech Republic, pp. 1214–1222 (2019). <https://doi.org/10.1145/3321707.3321814>
25. Ross, N.D.F., Johns, M.B., Keedwell, E.C., Savic, D.A.: Human-evolutionary problem solving through gamification of a bin-packing problem. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Prague, Czech Republic, pp. 1465–1473 (2019). <https://doi.org/10.1145/3319619.3326871>
26. Darejeh, A., Salim, S.S.: Gamification solutions to enhance software user engagement—a systematic review. *Int. J. Hum.-Comput. Interact.* **32**(8), 613–642 (2016). <https://doi.org/10.1080/10447318.2016.1183330>
27. Morschheuser, B., Hamari, J., Koivisto, J.: Gamification in crowdsourcing: a review. In: *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pp. 4375–4384 (2016). <https://doi.org/10.1109/hicss.2016.543>

28. Suh, A., Wagner, C., Liu, L.: Enhancing user engagement through gamification. *J. Comput. Inf. Syst.* **58**(3), 204–213 (2018). <https://doi.org/10.1080/08874417.2016.1229143>
29. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)



# Towards Novel Meta-heuristic Algorithms for Dynamic Capacitated Arc Routing Problems

Hao Tong<sup>1(✉)</sup>, Leandro L. Minku<sup>1</sup>, Stefan Menzel<sup>2</sup>, Bernhard Sendhoff<sup>2</sup>,  
and Xin Yao<sup>1(✉)</sup>

<sup>1</sup> School of Computer Science, University of Birmingham, Edgbaston,  
Birmingham B15 2TT, UK  
{hxt922,L.L.Minku,x.yao}@cs.bham.ac.uk

<sup>2</sup> Honda Research Institute Europe GmbH, 63073 Offenbach, Germany  
{stefan.menzel,bs}@honda-ri.de

**Abstract.** The Capacitated Arc Routing Problem (CARP) is an abstraction for typical real world applications, like waste collection, winter gritting and mail delivery, to allow the development of efficient optimization algorithms. Most research work focuses on the static CARP, where all information in the problem remains unchanged over time. However, in the real world, dynamic changes may happen when the vehicles are in service, requiring routes to be rescheduled. In this paper, we mainly focus on this kind of Dynamic CARP (DCARP). Some meta-heuristics solve (D)CARP by generating individuals that are sequences of tasks to be served as the individual representation. The split of this sequence into sub-sequences to be served by different vehicles needs to be decided to generate an executable solution, which is necessary for calculating individual's fitness. However, the existing split schemes for static CARP and DCARP are not capable of getting high quality feasible solutions for DCARP. Therefore, we propose two different split schemes in this paper – an optimal and a greedy split scheme. The optimal split scheme, assisted by A-star algorithm, can obtain the best vehicle routes from an ordered list. The greedy split scheme is not guaranteed to obtain optimal splits, but it is much more efficient. More importantly, it can keep the rank information between different individuals. Our experiments show that the greedy split scheme has good relative accuracy with respect to the optimal split scheme and that the two proposed split schemes are better than the existing DCARP split scheme in terms of the obtained solutions' quality.

**Keywords:** Dynamic CARP · Split scheme · A-star algorithm · Greedy search

## 1 Introduction

The Capacitated Arc Routing Problem (CARP) is a classical and important combinatorial optimization problem, dealing with a set of edges in a graph served by a number of vehicles with limited capacity [4]. Consider a pre-defined graph

containing a set of vertices and edges, where every edge has a travel cost and some edges, called tasks, have demands required to be served once by vehicles. CARP's aim is to find an optimal routing schedule that assigns vehicles to serve all tasks once and only once, minimizing the total travel cost. There is a wide range of real world applications related to CARP, such as waste collection [7], winter gritting [6], mail delivery [2] and others.

Plenty of approaches have been proposed to handle the various CARP applications, including constructive heuristic methods [3, 18], efficient algorithms for different kinds of CARP [1, 7, 16], algorithms for large scale CARPs [11, 17], and algorithms for uncertain CARPs [12, 16]. Most existing work focuses on static scenarios, in which the condition of a CARP does not change once it is given. However, in the real world, CARP is likely to dynamically change during the service process of vehicles. For example, new tasks may be added or some edges may not be available any more. This is referred to as Dynamic CARP (DCARP), and was, to the best of our knowledge, firstly investigated by [5, 15]. Liu et al. [10] proposed a memetic algorithm with a new split scheme to solve DCARP, which included six factors, i.e. vehicle availability, road accessibility, added/cancelled tasks, road congestion and change in tasks' demands. After Liu et al., to the best of our knowledge, only one work related to DCARP was published [13], in which failure of vehicles is the only dynamics considered.

Many algorithms for CARP usually produce a sequence of tasks, i.e. the individual represented in the meta-heuristic and evolutionary algorithms is a sequence of tasks [7, 16]. A split scheme is applied to obtain the executable sub-routes (i.e., sub-sequences of tasks) to be served by different vehicles. Such executable sub-routes are necessary for evaluating the fitness of the individuals. Fitness evaluation, and in particular the split scheme used during fitness evaluation, is thus a key point when using sequence of tasks as the individual representation in meta-heuristics to solve DCARP. However, split schemes for static CARP are unsuitable for DCARP, because DCARP has to assign outside vehicles to serve the remaining tasks and return to the depot, instead of using only vehicles that are in the depot. The only existing split scheme proposed for DCARP uses a random-based operator. Among other problems, it (1) makes the fitness evaluation noisy and (2) leads to fitness values that are unlikely to correspond to the actual fitness of the solution to be adopted in practice.

In this paper, we propose two new split schemes toward meta-heuristic algorithms for DCARP: (1) an optimal split scheme whose time and space complexity are high, and (2) a greedy split scheme which is not optimal, but has lower time and space complexity, being more suitable for real world applications. We perform experiments showing that the greedy split scheme maintains the rank of individuals of the population much better than the exiting split scheme, and also showing that the greedy split scheme is much more efficient than the proposed optimal split scheme.

The remainder of this paper is organized as follows. Section 2 discusses related work on split schemes for (D)CARP. Section 3 presents the main procedures of



our two proposed split schemes. Section 4 presents experiments to evaluate the proposed and existing DCARP split schemes. Section 5 concludes the paper.

## 2 Related Work

As explained in Sect. 1, the representation of individuals for solving CARP by meta-heuristic algorithms is a sequence of tasks. To form a feasible CARP solution whose fitness can be computed, this sequence needs to be split into different sections, each of them to be served by a different vehicle. Given this work's focus on DCARP split schemes, this section concentrates on split schemes.

### 2.1 Split Scheme for Static CARP

For static CARP, the Ulusoy's scheme [18] is used, which can obtain an optimal split from an individual by building an auxiliary graph. Assume an individual  $\{t_1, t_2, \dots, t_{N_t}\}$  for CARP, Ulusoy's split builds an auxiliary graph,  $G^*$ , to find the optimal split.  $G^*$  contains  $N_t + 1$  nodes, in which the first node represents the depot. Each edge  $(i, j)$  in the auxiliary graph represents a feasible sub-route,  $r_{sub}$ , of the CARP, and its cost is the weight,  $w_{i,j}$ , of the corresponding edge. For example, consider that edge  $(i, j)$  represents the route  $r_{sub} = \{depot \rightarrow t_{i+1} \rightarrow t_{i+2} \rightarrow \dots \rightarrow t_j \rightarrow depot\}$ , and  $w_{ij} = cost_{r_{sub}}$ . Then, Ulusoy's split uses Dijkstra's algorithm to find the shortest path from first node to the last node in  $G^*$ , which indicates the optimal split for the assigned individual. Ulusoy's split cannot be used for DCARP, because different vehicles in DCARP start from different stop points in the intermediate states, whilst all targets are the depot.

Some studies extended Ulusoy's split to multi-depot CARP [19]. However, each edge in the auxiliary graph represents a route that must start and end at the same depot. Therefore, these split schemes are also not suitable for DCARP, where vehicles start from different stop points in the intermediate states.

### 2.2 Split Schemes for DCARP

Liu et al. [10] proposed the first split scheme for DCARP, called Distance Based Split Scheme (DSS). Consider a given individual  $\{t_1, t_2, \dots, t_{N_t}\}$ . DSS randomly splits it into  $N_K$  sub-routes, where  $N_K$  is the number of vehicles used in service and is defined by the solution of the initial CARP instance. For each sub-route  $S_k$ , DSS calculates the sum of cost  $C_{pvj}$  of each vehicle  $j$  from its current position to each task's start node and end node:

$$C_{pvj} = \sum_{t_i}^{S_k} cost(v_j, start_{t_i}) + cost(v_j, end_{t_i}) \quad (1)$$

where  $v_j$  denotes the location of vehicle  $j$ . The vehicle with minimal  $C_{pvj}$  will be assigned to serve the  $S_k$ . After all vehicles are assigned to all tasks, the total

cost can be calculated for the whole schedule. Finally, DSS repeats the above process three times and selects the schedule with minimal total cost.

DSS has some significant weaknesses. Firstly, it makes the fitness evaluation noisy, as the fitness is not deterministic anymore and highly depend on the quality of the best among three random splits. Secondly, the fitness value obtained based on DSS is unlikely to correspond to the actual fitness of the solution to be adopted in practice. Thirdly, if the demand of a sub-route exceeds the remaining capacity of an assigned vehicle, DSS uses a path repair operator, where the vehicle returns back to the depot to get refilled and then goes back to continue serving the next task. This is inefficient to some extent, because other vehicles may still have big remaining capacities and could potentially be assigned to serve the remaining tasks that exceeded the original vehicle's capacity. Furthermore, DSS never considers new sub-routes starting from the depot, resulting in an inflexible schedule. In some cases, vehicles may not have enough capacity to serve the distant tasks after serving the near tasks so that they have to apply the repair operator to serve the distant tasks separately with a high cost. However, if it was allowed to create a new route for near tasks, vehicles would have enough capacity to serve all distant tasks simultaneously, avoiding to serve the distant tasks independently with a very high cost, and thus the total cost is reduced. Finally, DSS cannot guarantee an optimal split.

To overcome DSS' shortcomings, we propose two deterministic split schemes for DCARP. Both of them can handle the situation where vehicles start from different stop points, and they also consider that new vehicles can be assigned to serve tasks. One of them focuses on optimality and the other one on efficiency.

### 3 Proposed Split Schemes for DCARP Fitness Evaluation

For static CARP, Ulusoy's split finds the optimal split by building an auxiliary graph, in which the shortest path represents the optimal split. Inspired by this idea, our two proposed DCARP split schemes also mainly contain two steps: *auxiliary graph construction* and *path finding*. The two split schemes apply different path-finding strategies based on the same auxiliary graph.

#### 3.1 Auxiliary Graph Construction

In the auxiliary graph for static CARP, the edge between any two nodes represents a sub-route, serving a set of tasks. For instance, edge  $e_{ij}$  represents a sub-route serving the task's set  $\{t_{i+1}, t_{i+2}, \dots, t_j\}$ . Similarly, we also use an edge in the auxiliary graph to represent a sub-route for DCARP. Hence, for an ordered list of tasks, i.e.,  $t_1, t_2, \dots, t_{N_t}$ , there are  $N_t + 1$  nodes in the auxiliary graph. However, different from the static CARP, we already have some vehicles outside of the depot, which can be assigned to serve the remaining tasks. They stopped in different positions when the change happened, and they have to start from these positions to serve the remaining tasks. Therefore, an edge  $(i, j)$  between two nodes represents different routes for different vehicles in the auxiliary graph.

As a result, we construct several edges between two nodes in the auxiliary graph to represent sub-routes for all outside vehicles. Besides, considering that new sub-routes starting from the depot can also be created<sup>1</sup>, we add an additional edge between two nodes to represent the route starting from the depot. Algorithm 1 presents the procedure for building an auxiliary graph for DCARP.

---

**Algorithm 1:** Build auxiliary graph for DCARP

---

**Input:** Individual :  $I = \{t_1, t_2, \dots, t_N\}$   
 Stop points for outside vehicles:  $V = \{v_1, v_2, \dots, v_K\}$   
 Remaining capacity for outside vehicles:  $CP = \{cp_1, cp_2, \dots, cp_K\}$

- 1 Generate  $N + 1$  Nodes (Index from 0 to  $N$ ) for the auxiliary graph  $G^*$ ;
- 2  $v_0 = depot$ ,  $cp_0 =$  original capacity;
- 3  $V = \{v_0, v_1, v_2, \dots, v_K\}$ ,  $CP = \{cp_0, cp_1, cp_2, \dots, cp_K\}$ ;
- 4 **for** each vehicle  $k$  in  $V$  **do**
- 5     **for** each node pair:  $(Node_i, Node_j)$  **do**
- 6         Use vehicle  $k$  to serve  $\{t_{i+1}, t_{i+2}, \dots, t_j\}$ ;
- 7         Sub-route:  $r_{ijk} = \{v_k \rightarrow t_{i+1} \rightarrow t_{i+2}, \rightarrow \dots, \rightarrow t_j \rightarrow depot\}$ ;
- 8         Calculate the total demand  $d_{ijk}$  of  $r_{ijk}$ ;
- 9         **if**  $d_{ijk} > cp_k$  **then**
- 10             | *continue*;
- 11         **else**
- 12             | Calculate the cost of  $r_{ijk}$ :  $c_{ijk}$ ;
- 13             | Assign an edge  $e_{ijk}$  with weight  $c_{ijk}$  between  $Node_i$  and  $Node_j$ ;

**Output:** An auxiliary graph  $G^*$

---

Assume that  $K$  vehicles are currently outside the depot. The stop points and remaining capacities for vehicles are  $V = \{v_1, v_2, \dots, v_K\}$  and  $CP = \{cp_1, cp_2, \dots, cp_K\}$ . For the additional edge for new vehicles, we add a stop point  $v_0 = depot$  into  $V$  and  $cp_0$ , equal to the original capacity, into  $CP$ , in Line 2–3. For each pair of nodes  $(Node_i, Node_j)$ , we build an edge  $e_{ijk}$  for each vehicle  $k$ , which represents the sub-route  $r_{ijk} = \{v_k \rightarrow t_{i+1} \rightarrow t_{i+2} \rightarrow \dots \rightarrow t_j \rightarrow depot\}$ , in Lines 6–7. However, if the total demand of  $r_{ijk}$  exceeds the vehicle’s remaining capacity, edge  $e_{ijk}$  will be removed due to the capacity constrain, in Lines 8–10. Otherwise, the weight of  $e_{ijk}$  is assigned with the cost of  $r_{ijk}$ ,  $c_{ijk}$  in Lines 12–13.

### 3.2 A-Star Based Optimal Split Scheme

**Path Finding:** For static CARP, the Dijkstra algorithm is used directly to find the shortest path in the auxiliary graph. However, the number of edges in DCARP is much larger, as explained in Sect. 3.1, because there is a different edge

<sup>1</sup> New routes could potentially be served by a new vehicle (if we extra vehicles are available), or by an outside vehicle (after it finishes serving its currently assigned tasks and returns to the depot).

$(i, j, k)$  for each vehicle  $k$  between the pair of nodes  $(i, j)$ . To increase efficiency, we adopt the A-star algorithm to find the optimal path, instead of Dijkstra.

There are also two important constraints which have to be considered in DCARP and which did not exist in static CARP. First, any two edges in the whole path cannot belong to the same outside vehicle. Otherwise, the outside vehicle would have to serve two sub-routes, starting from the same outside stop point. However, when the vehicle finishes one sub-route, it stops at the depot. Therefore, it is impossible for this vehicle to serve another sub-route starting from an outside stop point. Secondly, all outside vehicles have to return to the depot even if they are not assigned to serve tasks in the new schedule. This means that, if no edge is assigned to a given outside vehicle in the whole path, the cost of this vehicle returning to the depot still needs to be considered in the total final split cost.

To use A-star, we need to determine a suitable admissible heuristic function  $h(n)$ . A heuristic is admissible if the cost it retrieves is smaller than or equal to the actual minimal cost to reach the target node in the tree from  $n$  [14]. In our problem, we use the minimal cost from the current node to the final node in the auxiliary graph without considering the constraints as the heuristic function. The cost function  $g(n)$  is calculated according to the actual path to reach  $n$ . In our split scheme, when different paths arrive at the same node in the auxiliary graph, the path with better cost will not replace the worse one because the choice of vehicles before influences the cost from the current node to the target. Therefore, the tree-based A-star search [14] is used in our split scheme. The procedure of the A-star-based optimal split scheme is presented in Algorithm 2.

The split scheme builds an auxiliary graph in Line 1 for shortest path finding. From the first node in the auxiliary graph, A-star expands each selected node with minimal  $f(n)$  and finds its successors, in Lines 8–9 and Line 20. The estimated costs  $f(n)$  for all successors are calculated in Lines 13–14. During the procedure of expanding the current node, the A-star-based split scheme applies two strategies to handle the two constraints for DCARP. Firstly, in order to avoid the repetition of edges belonging to the same vehicle, it removes all edges belonging to the vehicles which have already been selected in the current part of the path and then explores the rest of path, in Line 12. For the second constraint, when A-star reaches the final node, we repair the cost of the final path if some outside vehicles are never selected, adding the returning cost for these non-selected outside vehicles in Lines 15–16.

**Complexity Analysis:** A-star guarantees the optimality of the path found. However, it has a high space and time complexity since there are  $N$  tasks in total. For static CARP, the number of edges is  $\#E_S = \frac{N(N-1)}{2}$ , and the number of all possible paths is  $\#P_S = 2^{N+1}$  in the auxiliary graph. In the auxiliary graph for DCARP, if there are  $K$  outside vehicles, the number of edges is

$$\#E_D = \frac{(K+1) \cdot N(N-1)}{2}$$

---

**Algorithm 2:** A\* based optimal split scheme

---

**Input:** Individual :  $I = \{t_1, t_2, \dots, t_N\}$

- 1 Build an auxiliary graph  $G^*$  for DCARP;
- 2  $expandNode = Node_0$ ;  $openNodeSet = \{\}$ ;  $pathSet = \{\}$ ;
- 3 **while** *True* **do**
- 4     **if**  $expandNode == target$  **then**
- 5         Shortest path  $P$ : path correspond to  $expandNode$ ;
- 6         Minimal cost  $C$ :  $f_{expandNode}$  correspond to  $expandNode$ ;
- 7         **break**;
- 8     Select  $rootPath$  (i.e. path from  $Node_0$  to  $expandNode$ ) from  $pathSet$ ;
- 9     Find all feasible successors for  $expandNode$  in the graph;
- 10    **for** each successor of  $expandNode$  **do**
- 11         $newPath = rootPath + expandNode \rightarrow successor$ ;
- 12        Remove all edges belong to vehicles used in  $newPath$  for  $successor$ ;
- 13        Calculate the  $h_{succ}$  and  $g_{succ}$ ;
- 14        Set  $f_{succ} = h_{succ} + g_{succ}$ ;
- 15        **if**  $successor == target$  **then**
- 16            Repair  $f_{succ}$ ;
- 17        Add the  $successor$  into  $openNodeSet$ ;
- 18        Add the  $newPath$  into  $pathSet$ ;
- 19     Remove  $expandNode$ ,  $rootPath$  from  $openNodeSet$  and  $pathSet$ ;
- 20     Select the node in  $openNodeSet$  with minimal  $f$  as  $expandNode$ ;
- 21 The shortest path from  $Node_0$  to  $target$  in  $G^*$ :  $P = \{p_1, p_2, \dots, p_M\}$ ;
- 22 Each  $p_m$  represents an edge  $e_{ijk}$ , which denotes a sub-route  $r_{ijk}$ ;
- 23 Obtain the solution  $S$  by splitting the  $I$  by  $P$ .

**Output:** Solution  $S = \{r_1, r_2, \dots, r_M\}$ , Minimal cost:  $C$

---

and the number of all possible paths, i.e. from the first node to the target in the auxiliary graph, is

$$\#P_D = \sum_{n=1}^N C_{N-1}^{n-1} \sum_{i=0}^{\min(n,K)} C_n^{n-i} \cdot P_K^i$$

Therefore, the number of routes in the auxiliary graph for DCARP is much larger than in the static case. The A-star algorithm has to save all expanded nodes during the search process. In the worst case, it will visit and save all  $\#P_D$  possible paths. Even though heuristics can frequently avoid the worst case scenario, the computational time still depends on  $\#P_D$  and is often still unacceptably high, as will be demonstrated by our experiments in Sect. 4.3.

### 3.3 Greedy Split Scheme

**Path Finding:** As discussed above, the A-star based optimal split scheme is computationally expensive when using A-star search to find the optimal path in

the auxiliary graph. Therefore, we propose a greedy strategy to find a path with a good quality in the auxiliary graph.

In the auxiliary graph, each edge  $e_{ijk}$  represents a route  $r_{ijk}$ , serving a list of tasks and having a cost  $c_{ijk}$ . So, we can obtain an efficiency parameter for each route, which is the average cost for each demand of this route. Generally, when there are several possible routes to be selected, we will choose the route with the lowest average cost for each demand, from a greedy perspective. The greedy procedure is presented in Algorithm 3.

---

**Algorithm 3:** Greedy split scheme

---

**Input:** Individual :  $I = \{t_1, t_2, \dots, t_N\}$

- 1 Build an auxiliary graph  $G^*$  for DCARP;
- 2 **for** each edge  $e_{ijk}$  in  $G^*$  **do**
- 3     $\lfloor$  Calculate the ACD:  $ACD_{ijk}$ ;
- 4  $expandNode = Node_0$ ;  $newPath = Node_0$
- 5 **while** *True* **do**
- 6    **if**  $expandNode == target$  **then**
- 7        Greedy path:  $newPath, P = \{p_1, p_2, \dots, p_M\}$ ;
- 8        Calculate the greedy cost of greedy path:  $C$ ;
- 9         $\lfloor$  break;
- 10     $rootPath \leftarrow newPath$ ;
- 11    Find all edges linking to *successors* for  $expandNode$ ;
- 12    Select the edge with the minimal  $ACD$ ;
- 13     $Node_X \leftarrow successor$  that the selected edge belongs to;
- 14     $newPath = rootPath + expandNode \rightarrow Node_X$ ;
- 15    Remove all edges corresponding to vehicles being used in  $newPath$ ;
- 16     $expandNode \leftarrow Node_X$ ;
- 17 Each  $p_m$  represents an edge  $e_{ijk}$ , which denotes a sub-route  $r_{ijk}$ ;
- 18 Obtain the solution  $S$  by splitting the  $I$  by  $P$ .

**Output:** Solution  $S = \{r_1, r_2, \dots, r_m\}$ , Greedy cost:  $C$

---

Assuming the ordered tasks  $\{t_1, t_2, \dots, t_{N_i}\}$ , we build an auxiliary graph according to Algorithm 1 (Line 1). Then, we calculate the average cost for each demand (ACD), for each edge  $e_{ijk}$ , as  $ACD_{ijk}$ , in Lines 2–3. In each step, the greedy split scheme will select the edge with the minimal  $ACD$  from all edges linking to the current node, in Lines 11–12. Similarly, in order to satisfy the constraints, when an edge belonging to one outside vehicle is selected in each step, it will remove all edges corresponding to this vehicle in the later path-finding process, in Line 15. Finally, the process terminates when the target is found, and then the actual cost for the greedy path is calculated, considering the return cost of any unused outside vehicles, in Lines 7–8.

**Complexity Analysis:** The greedy split scheme is much more efficient than A-star-based optimal split scheme. Without considering the auxiliary graph construction, its time complexity is only  $\mathcal{O}(N_t)$ .

## 4 Experiments

The greedy split scheme has much lower time complexity than the optimal split scheme, but may lead to splits of lower quality. Given that the split scheme will be used as part of the fitness evaluation of individuals within a meta-heuristic algorithm, it is desirable that better individuals according to the optimal split scheme are still considered as better individuals according to the greedy split scheme. In particular, if the ranking (relative accuracy) of individuals does not change when adopting the greedy instead of the optimal split scheme, the lower quality of the splits obtained by the greedy split scheme will not negatively affect the meta-heuristic algorithm, depending on the selection mechanisms being used. Therefore, in this section, we compare the relative accuracy of the greedy split scheme (GSS) with that of the existing split scheme, the distance-based split scheme (DSS) [9]. In addition, we also compare the fitness and computational time for different split schemes.

### 4.1 Comparison on Relative Accuracy

At first, we test three difference split schemes in a series of problem instances to show the relative accuracy of GSS and DSS, relative to optimal split schemes (OSS). Two benchmark sets, referred as *gdb* set [3] and *egl* set [8], for static CARP are used as basis map for DCARP in our experiments. Each benchmark generates one scenario for simulating the situation of changes happening, where the changes include broken down vehicles, closed roads, roads congestion, added tasks and increased demands. After the changes, *gdb* and *egl* instances have on average 174 and 37 tasks, representing a high and a low dimensional set of instances, respectively. For each scenario after the changes,  $N_p = 40$  individuals are randomly generated and evaluated by different split schemes.

In order to compare the relative accuracy, we use the Kendall rank correlation coefficient ( $\tau$ ) as the measurement. Assuming that the fitnesses of individual  $i$  are  $f_{OSS}$ ,  $f_{GSS}$  and  $f_{DSS}$ ,  $\tau$  can be calculated as

$$\tau = \frac{2}{N_p(N_p - 1)} \sum_{i < j} \text{sgn}(f_{OSS_i} - f_{OSS_j}) \text{sgn}(f_{SS_i} - f_{SS_j}) \quad (2)$$

where  $f_{SS_i}$  denotes  $f_{GSS_i}$  or  $f_{DSS_i}$ , and  $N_p$  is the size of population.  $\tau \in [-1, 1]$  and a large  $\tau$  indicates the relative accuracy of one split schemes is highly agreed with the optimal split scheme.

The results on relative accuracy are presented in Fig. 1. The left part belongs to the *egl* set of benchmarks, whose dimension, i.e. the number of tasks, is larger than that of the *gdb* set in the right part. From the result, GSS is better than DSS with respect to the relative accuracy, especially in high dimension. It is mainly because DSS splits the individuals by random selection. The fitness obtained highly depends on the random seed, which influences the relative accuracy deeply. For example, for two individuals, the OSS can provide the relative fitness, but the better individual may be splitted into a low quality solution

due to a ill-chosen random seed. By contrast, the GSS split an individual on a deterministic way and there is no randomness in the split procedure, so that its relative accuracy is higher than DSS.

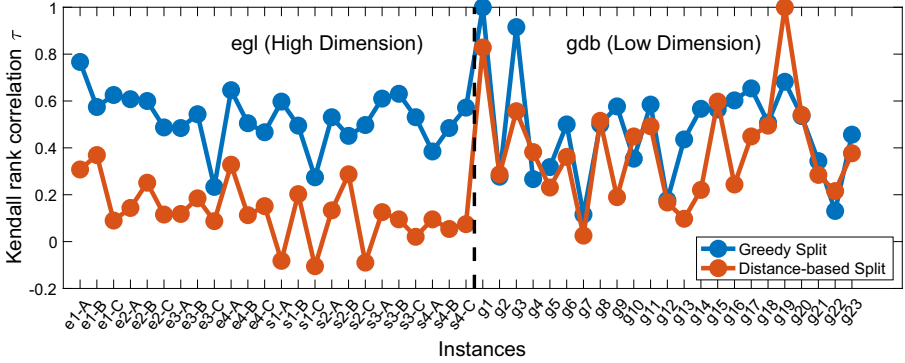


Fig. 1. Relative accuracy achieved by GSS and DSS.

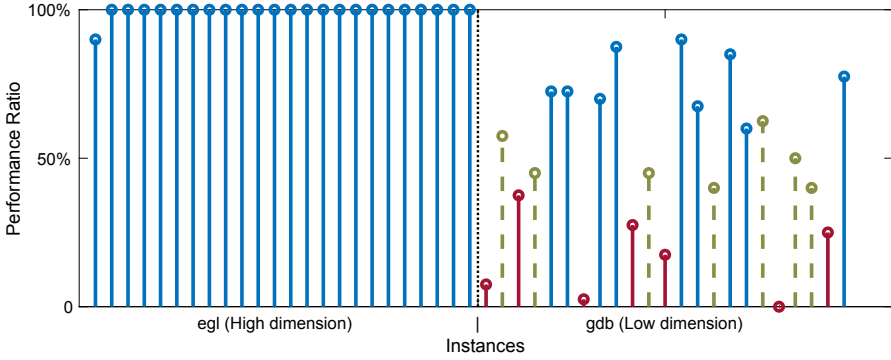
However, when the problem is of low dimension, the difference in relative accuracy is small, because the search space for split becomes small. The result of one random split is close to another random split, so that the influence of randomness decreases.

### 4.2 Comparison on Obtained Fitness

For the same individual, GSS and DSS obtain different solutions, with different fitness values, i.e. total cost. Therefore, in this subsection, we compare the performance of GSS and DSS, with respect to the fitness values. For each population with  $N_p$  individuals, as shown in the previous subsection, we compare every fitness obtained by GDD and DSS. Wilcoxon Sign Rank tests with a 0.05 significance level were carried out to compare the fitness values of the individuals and the ratio of individuals where GSS was better than DSS in the population is shown in Fig. 2, where the blue(red) solid stems represent that GSS (DSS) performs significantly better than DSS (GSS) and the dash stems represents that both split schemes have no significant difference. In conclusion, GSS performs better than DSS significantly in most DCARP cases.

We can see that in all high dimensional cases, the fitness values obtained by GSS are almost all better than those of DSS in each population. However, GSS’s performance decreases compared with that of DSS in low dimensional cases. There are two main reasons to explain the results. First, as discussed in the previous subsection, the solution obtained by DSS highly depends on the random seed. When the problem dimension is very low, the random split is likely to obtain a very good solution thanks to its diversity and the small search space. However, it cannot help a lot in the high dimensional cases. Second, when a





**Fig. 2.** Ratio of individuals where GSS is better than DSS in the population. Solid stems represent DSS and GSS have significant difference and dash stems represent they have no significant difference.

given vehicle’s capacity is exceeded due to a change, DSS uses a repair operator to make the vehicle return to the depot first and then continue to serve the remaining tasks from the depot. This typically results in higher costs than if new sub-routes starting from the depot could be created during the split process as done by our approaches, given that new sub-routes not only directly help to satisfy the capacity constraints, but also are optimized together with the sub-routes starting from outside, which makes our split more flexible and efficient.

### 4.3 Comparison on Computational Time

Finally, in this subsection, we will compare the computational time for three split schemes. We select 8 maps for each dataset as test scenario. As discussed in Sect. 3, OSS might require much time to obtain the solution. Therefore, in the previous experiments, the scenarios we generated were suitable for OSS to obtain the result within 300 seconds, to have enough results for the comparisons. In this section, we randomly generate the test scenarios. If OSS is unable to find the optimal solution within 300 seconds, the experiment is considered as a failure and the computational time is not considered. The computational results are presented in Table 1, in which the time for GSS and OSS contains the time for auxiliary graph construction.

GSS is the most efficient method among the three split schemes, and OSS is the most computationally expensive in all test scenarios. GSS starts from the first node in the auxiliary graph, and selects the edge with minimal  $ACD$ . DSS splits the individuals randomly. Although the random split is very efficient, the repair operator is required to obtain the total cost, which is a relatively computationally expensive process. OSS has a very large search space, and the A-star algorithm will save all explored nodes, which causes its computational time to be very large in some cases. From the results, we can also observe that the number of tasks and outside vehicles have a big impact on the computational

**Table 1.** The computational time (in *seconds*) for OSS, GSS, DSS. For OSS, the success rate is shown in brackets.  $N_t$  is the number of tasks (dimension) and  $K$  is the number of outside vehicles.

Instances	$N_t$	$K$	$t_{GSS}$	$t_{DSS}$	$t_{OSS}$	Instances	$N_t$	$K$	$t_{GSS}$	$t_{DSS}$	$t_{OSS}$
egl-e1-A	35	2	0.03	0.70	0.22 (40/40)	gdb2	3	3	0.04	0.08	0.13 (6/6)
egl-e2-A	39	3	0.03	0.66	0.77 (40/40)	gdb5	2	3	0.02	0.06	0.03 (2/2)
egl-e3-A	29	4	0.03	0.42	9.91 (2/40)	gdb8	20	7	0.03	0.29	11.4 (6/40)
egl-e4-A	38	6	0.04	0.64	27.3 (8/40)	gdb9	16	5	0.02	0.29	0.86 (39/40)
egl-s1-A	70	7	0.07	1.38	N/A (0/40)	gdb16	7	1	0.02	0.15	0.05 (40/40)
egl-s2-A	86	13	0.11	1.49	N/A (0/40)	gdb18	14	3	0.02	0.27	0.07 (40/40)
egl-s3-A	66	9	0.08	1.08	N/A (0/40)	gdb22	15	5	0.02	0.28	4.58 (38/40)
egl-s4-A	64	4	0.04	0.88	6.78 (1/40)	gdb23	21	4	0.02	0.36	26.5 (36/40)

time for OSS. When there are many outside vehicles, the search space becomes very large, so that it is very hard to find the optimal split results for OSS. This is because the tasks and outside vehicles directly determine the search space as shown in Eq. (23).

## 5 Conclusion

The split scheme is essential for fitness evaluation in DCARP. However, the existing split scheme for static CARP is unsuitable for DCARP, and the existing DCARP splitting scheme highly depends on the random seed, being unable to provide stable results. Therefore, in this paper, we propose two new split schemes. The first split scheme is an optimal split scheme based on the A-star search. It is capable to provide an optimal solution for an ordered list of tasks. However, it is computationally expensive in many scenarios due to the huge search space. The second is a greedy split scheme, which is much more efficient than the optimal split scheme and even than the existing random split scheme. Our experiments show that the greedy split scheme is capable of leading to similar individual rankings to the optimal split scheme, and its fitness is much better than that of the existing random split scheme for DCARP, especially in high dimensional test cases.

Future work includes the testing of proposed split schemes in more real instances and the design of meta-heuristic methods to solve DCARP using the proposed split schemes.

**Acknowledgements.** Hao Tong gratefully acknowledges the financial support from Honda Research Institute Europe (HRI-EU).

## References

1. Brandão, J., Eglese, R.: A deterministic tabu search algorithm for the capacitated arc routing problem. *Comput. Oper. Res.* **35**(4), 1112–1126 (2008)
2. Eiselt, H.A., Gendreau, M., Laporte, G.: Arc routing problems, part ii: the rural postman problem. *Oper. Res.* **43**(3), 399–414 (1995)
3. Golden, B.L., DeArmon, J.S., Baker, E.K.: Computational experiments with algorithms for a class of routing problems. *Comput. Oper. Res.* **10**(1), 47–59 (1983)
4. Golden, B.L., Wong, R.T.: Capacitated arc routing problems. *Networks* **11**(3), 305–315 (1981)
5. Handa, H., Chapman, L., Yao, X.: Dynamic salting route optimisation using evolutionary computation. In: 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 158–165. IEEE (2005)
6. Handa, H., Chapman, L., Yao, X.: Robust route optimization for gritting/salting trucks: a CERCIA experience. *IEEE Comput. Intell. Mag.* **1**(1), 6–9 (2006)
7. Lacomme, P., Prins, C., Ramdane-Chérif, W.: Evolutionary algorithms for periodic arc routing problems. *Eur. J. Oper. Res.* **165**(2), 535–553 (2005)
8. Li, L.Y., Eglese, R.W.: An interactive algorithm for vehicle routing for winter-gritting. *J. Oper. Res. Soc.* **47**(2), 217–228 (1996)
9. Liu, M., Singh, H.K., Ray, T.: A benchmark generator for dynamic capacitated arc routing problems. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 579–586. IEEE (2014)
10. Liu, M., Singh, H.K., Ray, T.: A memetic algorithm with a new split scheme for solving dynamic capacitated arc routing problems. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 595–602. IEEE (2014)
11. Mei, Y., Li, X., Yao, X.: Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. *IEEE Trans. Evol. Comput.* **18**(3), 435–449 (2013)
12. Mei, Y., Tang, K., Yao, X.: A global repair operator for capacitated arc routing problem. *IEEE Trans. Syst. Man Cybern. B Cybern.* **39**(3), 723–734 (2009)
13. Monroy-Licht, M., Amaya, C.A., Langevin, A., Rousseau, L.M.: The rescheduling arc routing problem. *Int. Trans. Oper. Res.* **24**(6), 1325–1346 (2017)
14. Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*, 3rd edn. Prentice Hall, Upper Saddle River (2010)
15. Tagmouti, M., Gendreau, M., Potvin, J.Y.: A dynamic capacitated arc routing problem with time-dependent service costs. *Transp. Res. Part C Emerg. Technol.* **19**(1), 20–28 (2011)
16. Tang, K., Mei, Y., Yao, X.: Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Trans. Evol. Comput.* **13**(5), 1151–1166 (2009)
17. Tang, K., Wang, J., Li, X., Yao, X.: A scalable approach to capacitated arc routing problems based on hierarchical decomposition. *IEEE Trans. Cybern.* **47**(11), 3928–3940 (2016)
18. Ulusoy, G., et al.: The fleet size and mix problem for capacitated arc routing. *Eur. J. Oper. Res.* **22**(3), 329–337 (1985)
19. Xing, L., Rohlfshagen, P., Chen, Y., Yao, X.: An evolutionary approach to the multidrop capacitated arc routing problem. *IEEE Trans. Evol. Comput.* **14**(3), 356–374 (2009)



# Robust Evolutionary Bi-objective Optimization for Prostate Cancer Treatment with High-Dose-Rate Brachytherapy

Marjolein C. van der Meer<sup>1</sup>(✉), Arjan Bel<sup>1</sup>, Yury Niatsetski<sup>2</sup>,  
Tanja Alderliesten<sup>3</sup>, Bradley R. Pieters<sup>1</sup>, and Peter A. N. Bosman<sup>4</sup>

<sup>1</sup> Department of Radiation Oncology, Amsterdam UMC, University of Amsterdam, Amsterdam, The Netherlands

`marjolein.vandermeer@amsterdamumc.nl`

<sup>2</sup> Physics and Advanced Development, Elekta, Veenendaal, The Netherlands

<sup>3</sup> Department of Radiation Oncology, Leiden University Medical Center, Leiden, The Netherlands

<sup>4</sup> Life Sciences and Health Research Group, Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

**Abstract.** We address the real-world problem of automating the design of high-quality prostate cancer treatment plans in case of high-dose-rate brachytherapy, a form of internal radiotherapy. For this, recently a bi-objective real-valued problem formulation was introduced. With a GPU parallelization of the Multi-Objective Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm (MO-RV-GOMEA), good treatment plans were found in clinically acceptable running times. However, optimizing a treatment plan and delivering it to the patient in practice is a two-stage decision process and involves a number of uncertainties. Firstly, there is uncertainty in the identified organ boundaries due to the limited resolution of the medical images. Secondly, the treatment involves placing catheters inside the patient, which always end up (slightly) different from what was optimized. An important factor is therefore the robustness of the final treatment plan to these uncertainties. In this work, we show how we can extend the evolutionary optimization approach to find robust plans using multiple scenarios without linearly increasing the amount of required computation effort, as well as how to deal with these uncertainties efficiently when taking into account the sequential decision-making moments. The performance is tested on three real-world patient cases. We find that MO-RV-GOMEA is equally well capable of solving the more complex robust problem formulation, resulting in a more realistic reflection of the treatment plan qualities.

**Keywords:** Evolutionary Algorithms · Robust optimization · Multi-objective optimization · Empirical study · Radiation oncology

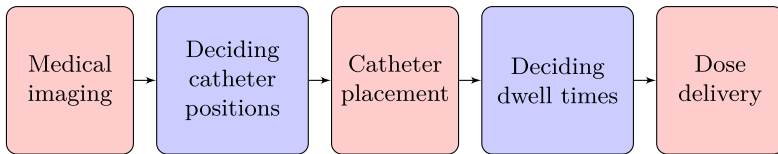
---

Supported by Elekta, Sweden.

© Springer Nature Switzerland AG 2020  
T. Bäck et al. (Eds.): PPSN 2020, LNCS 12270, pp. 441–453, 2020.  
[https://doi.org/10.1007/978-3-030-58115-2\\_31](https://doi.org/10.1007/978-3-030-58115-2_31)

# 1 Introduction

Brachytherapy is a form of internal radiotherapy that can be used for treating prostate cancer. The treatment involves intraoperative placement of a number of very thin needles, called catheters, inside the patient, for a radioactive source to be moved through. The entire procedure is a two-stage sequential decision-making process. The first part is determining how to place the catheters, after which the catheters are actually placed. The second part is determining how to move the radioactive source through the catheters, which is performed after catheter placement. Each catheter has a fixed set of positions, called dwell positions, where the radioactive source can pause for certain amounts of time, called dwell times. The workflow is illustrated in Fig. 1.



**Fig. 1.** The simulation workflow used to study robust optimization. Purple blocks indicate decision-making stages. (Color figure online)

Key quality indicators can be formulated for treatment plans, enabling optimization to support decision making. For the first part, both catheter position variables and dwell time variables play a role. For the second part, only the dwell time variables still play a role. On the one hand, enough catheters should be placed to ensure a good treatment. On the other hand, a larger number of catheters increases the risk of complications for the patient. The number of catheters is therefore an important part of the catheter position optimization. Since catheter placement is performed in the operating room, no changes to the catheters can be made afterwards. The result of catheter position optimization (that includes dwell times) should therefore be representative of what can be achieved in the dwell time optimization.

Optimizing brachytherapy is difficult, for multiple reasons. Firstly, the problem is inherently multi-objective, due to the trade-off between radiation dose to the tumor (which you want to maximize) and to the surrounding tissue (which you want to minimize). Secondly, the objective functions comprising the multi-objective problem are such that there is no gradient information. Thirdly, there are time constraints on the optimization due to the patient waiting for treatment. For solving difficult multi-objective problems, Evolutionary Algorithms (EAs) are the state-of-the-art [3]. Previous work on bi-objective optimization for both dwell times [2] and catheter positions [11] has shown promising results.

Since all problem variables in the second stage are already part of the first stage, arguably there is no need for a two-stage optimization process. After a single optimization, both decisions can be made and executed. However, in practice such a one-shot approach is not sufficient, because brachytherapy involves

a number of uncertainties. Firstly, there is uncertainty in the identified organ boundaries due to the limited resolution of the medical images used for treatment planning. Secondly, catheters always end up (slightly) different in the patient from what was planned. Therefore, to ensure the best possible plans, the dwell times should be re-optimized after actual catheter placement has taken place. To avoid overly optimistic catheter position optimization fronts, that may lead to the wrong conclusion about how many catheters are needed for a particular patient in the first decision phase, these uncertainties should be taken into account in the optimization problem. Since the most time consuming part of the optimization is the calculation of the objective functions of a treatment plan, straightforwardly applying robust optimization would result in clinically infeasible run times.

In this work, we will introduce robust optimization to the full workflow for high-dose-rate prostate brachytherapy, while still keeping the run times low. The aim is for the optimization fronts to be representative of what can be achieved in clinical practice. Specifically, the catheter position optimization fronts should be representative of what can be achieved later in the dwell time optimization. We will evaluate the run time of the optimization, as well as the robustness of the resulting treatment plans.

## 2 Background

### 2.1 Insightful Decision Support via Bi-objective Optimization

The ultimate goal is to obtain the highest quality treatment plan to be used for the dose delivery. There are several key evaluation criteria that can be mathematically formulated, enabling the use of optimization methods. For an in-depth explanation of all details involved, we refer the interested reader to related literature [11]. Here, we briefly summarize the most important concepts.

In clinical practice, the evaluation of a treatment plan is based on a clinical protocol, which describes how much radiation the prostate and seminal vesicles should receive as part of the treatment, as well as how much dose is maximally allowed to the surrounding healthy organs to avoid complications. This radiation dose that is prescribed for the prostate is called the *planning-aim dose*. The clinical protocol of the Amsterdam UMC is formulated in terms of so-called dose-volume indices. There are two types of dose-volume indices; volume indices and dose indices. A volume index  $V_x^o$  is the volume of organ  $o$  that receives at least  $x\%$  of the planning-aim dose. A dose index  $D_x^o$  is the lowest dose to the most irradiated  $x\text{cm}^3$  of organ  $o$ .

Single-objective optimization approaches are often based on a simplified version of the clinical protocol [7, 8]. All objectives following from the simplified protocol are combined into a single optimization function by the weighted-sum approach. As a result, optimized treatment plans often require manual improvements by the medical planners [4], which is a time-consuming and little insightful process. Alternatively, optimizing for all these indices would entail solving a

many-objective optimization problem, for which the results are not straightforward to interpret, visualize, and use for decision making.

For this reason, two grouped objectives were defined that have proven effective and insightful for clinical practice [9]. The resulting bi-objective optimization model is based directly on the clinical protocol. Dose-volume index criteria related to the dose coverage of the prostate and the seminal vesicles are combined into the Least Coverage Index (LCI). Criteria related to the sparing of organs at risk, namely rectum, bladder, and urethra, are combined into the Least Sparing Index (LSI). Upper bounds to the amount of radiation the prostate can receive also fall under the LSI. Since it is unknown a priori how to weight the different dose-volume index criteria, the objectives are constructed by combining the criteria in a worst-case manner, which was observed to be much related to how plans are manually improved in clinical practice. This results in the following optimization objectives:

$$\begin{aligned} \text{LCI} &= \min \left\{ V_{100\%}^{\text{prostate}} - 95, V_{80\%}^{\text{vesicles}} - 95 \right\}, \\ \text{LSI} &= \min \left\{ 86 - D_{1\text{cm}^3}^{\text{bladder}}, 74 - D_{2\text{cm}^3}^{\text{bladder}}, 78 - D_{1\text{cm}^3}^{\text{rectum}}, 74 - D_{2\text{cm}^3}^{\text{rectum}}, \right. \\ &\quad \left. 110 - D_{0.1\text{cm}^3}^{\text{urethra}}, 50 - V_{150\%}^{\text{prostate}}, 20 - V_{200\%}^{\text{prostate}} \right\}. \end{aligned} \quad (1)$$

For catheter position optimization, an additional constraint on the healthy tissue immediately surrounding the prostate is necessary [11]. This constraint is based on the number of catheters  $N$ :

$$C = \left\{ \begin{array}{ll} V_{200\%}^{\text{healthy tissue}} - 0.125N, & \text{for LSI} \geq -25 \\ V_{200\%}^{\text{healthy tissue}} - 0.125N \left( 1 + \frac{-25 - \text{LSI}}{100} \right), & \text{for LSI} < -25 \end{array} \right\} \leq 0. \quad (2)$$

The result of solving this problem is a trade-off curve of treatment plans that, when satisfying  $\text{LCI} > 0$  and  $\text{LSI} > 0$  (and  $C \leq 0$ ) adhere to the clinical protocol. Visualizing this makes the most important trade-offs immediately insightful, as well as whether the clinical protocol can be achieved.

## 2.2 Problem Variables

As mentioned in the introduction, there are two decision phases. In the first phase, the catheter positions need to be optimized. To use the model of Sect. 2.1, we also need to set the dwell times per catheter. Hence, in the first phase, all catheter positions are optimized at the same time as the dwell times pertaining to these catheters. In the second phase, the catheter positions are fixed, and only the dwell times are to be optimized.

For catheter position optimization, the number of catheters is given as input. Moreover, constraints are added to the optimization model describing which catheter positions are feasible. Catheters have to be inside either the prostate (with a  $-1$  mm margin) or the seminal vesicles. Catheters are not allowed to intersect with either rectum or urethra (both with a  $1$  mm margin). Finally, the distance between the surfaces of each pair of catheters has to be at least  $1$  mm. For an in-depth explanation of all details involved, we refer the interested reader to related literature [11].

## 2.3 Evolutionary Optimization

In a comparison between different EAs, the best performing EA for the problem at hand was the Multi-Objective Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm (MO-RV-GOMEA) [9]. A key reason is that MO-RV-GOMEA is capable of exploiting gray-box settings where problem-specific enhancements can be readily applied. Specifically, MO-RV-GOMEA makes use of so-called *partial evaluations*. Instead of changing all variables of a potential solution and then performing an evaluation, the variables are changed in multiple steps, and after each step an evaluation is performed. If the solution was improved, the change is kept; if not, the change is reverted. For brachytherapy, these many evaluations can be done efficiently, because the impact of changes to certain dwell times can be computed by considering radiation originating from the corresponding dwell positions only [9]. A similar argument holds for catheter positions [11].

A second reason for its enhanced performance is that MO-RV-GOMEA models the dependencies between variables by using a so-called *Linkage Tree* (LT). At the bottom of this tree, each of the variables is in a singleton set. Higher up, sets are merged together based on the strength of the dependencies between their variables. At the top of the LT, all sets have been merged, resulting in a single set containing all variables. Combining all sets in the LT results in the so-called *Family Of Subsets* (FOS). During optimization, all FOS elements are considered. For every FOS element, a joint Gaussian distribution is estimated, based on a selection of best solutions. Such a Gaussian distribution is known to work well when there is no gradient information and the fitness landscape may not be smooth everywhere, e.g., it is adopted by the state-of-the-art real-valued EA known as CMA-ES [6]. The variables that are in one FOS element are then resampled together based on this distribution. This way, dependencies between variables are taken into account. When applied to brachytherapy, the dependencies between variables are modelled based on the distances between the dwell positions [9, 11].

## 3 Accounting for Uncertainties via Robust Optimization

### 3.1 Organ Reconstructions: A Problem-Specific Solution

While the dose-volume indices of a treatment plan are theoretically uniquely defined, computing values for these indices in practice is not. A key reason is that dose-volume indices are computed from 3D (organ) volumes. However, medical scans are usually sets of 2D images. An algorithm is then used to reconstruct the 3D organ shapes from delineations, performed on the 2D images. Due to the limited resolution of the medical images, such a reconstruction is not uniquely defined and differs from one clinical system to another. A solution is to perform robust optimization over different organ reconstructions, to avoid overfitting on one particular reconstruction. Three organ reconstruction settings have previously been studied, for details, see [10]. Combinations of these settings yield 8 possible 3D organ reconstructions per patient. Hence, there are 8 combinations



of (LCI, LSI, C) values per plan. Taking again a worst-case scenario approach to combining objective values (in this case defined for different reconstruction settings), the robust optimization model is defined as

$$\text{LCI} = \min_{i=1,\dots,8} \{\text{LCI}_i\}, \quad \text{LSI} = \min_{i=1,\dots,8} \{\text{LSI}_i\}, \quad \text{C} = \max_{i=1,\dots,8} \{\text{C}_i\}. \quad (3)$$

This model is identical for both dwell time optimization and catheter position optimization. A straightforward implementation would be to compute the LCI, LSI, and C 8 times. This would lead to approximately 8 times more computational effort, as calculating the objective values associated with a treatment plan is the most time-consuming component in the EA. With runtime being important for clinical usability, reducing this additional runtime is important. To do so, advantage is taken of the large volume overlap between different organ reconstructions (i.e., it is at the borders that organ reconstructions differ, not at the interiors). When evaluating the quality of a treatment plan, the dose in each overlapping part of the patient in all scenarios is calculated only once. The parts that do not overlap are small, and evaluated separately, for each reconstruction. After this, the dose-volume indices are calculated 8 times. As a result, performing a fixed number of evaluations is only approximately twice as slow as the original optimization.

### 3.2 Catheter Displacements: An EA Generic Solution

When catheters are placed inside the patient, they always end up (slightly) different from what was planned. Accounting for this uncertainty requires taking into account the fact that the complete workflow is a sequential decision-making process. Between catheter position optimization and dwell time optimization, there is the catheter placement which causes the uncertainty. We will simulate actual catheter placement by randomly displacing all catheters by 1 mm, where the 1 mm is based on discussions with a clinical expert. After the displacements, dwell times are re-optimized, but catheter positions are fixed.

If these displacements are not taken into account, catheter position optimization fronts will be overly optimistic compared to the dwell time optimization fronts, because optimization will overfit on the one scenario in which catheters are not displaced at all. Hence, a lower number of catheters will appear to be sufficient than is really the case. As a result, optimization will be an ineffective decision support tool because likely not enough catheters would be placed in the patient to ensure a good treatment.

To avoid this, the random catheter displacements should thus be taken into account in the optimization. The most straightforward approach to do so correctly would be to consider many catheter displacements (in the order of 100) each time a set of catheter positions is evaluated. For each catheter displacement, dwell times would be separately optimized, to take into account that dwell time optimization is performed again after the catheters are displaced. Unfortunately, this would be prohibitively computationally expensive, because one full dwell-time optimization takes about 30 s [2] and we have only a few minutes to decide catheter positions in clinical practice.

Alternatively, when a set of catheter positions is evaluated, dwell times could be kept fixed when displacing catheters. This is a conservative lower bound on the real evaluation, since this disregards the dwell time optimization performed after catheter placement. This would therefore result in overly pessimistic catheter position optimization fronts, which is also undesirable from the perspective of a clinical decision support tool. As a result, too many catheters would be placed in the patient, which would increase the risk of complications. Moreover, due to the many scenarios, this approach would still be too computationally expensive.

We therefore propose a third approach that is generic to multi-objective EAs in sequential decision-making processes under uncertainty where the uncertainty between stages involves variable realization (i.e., realizing the actual optimized catheter positions in the clinic). When evaluating a treatment plan for multiple catheter displacements, dwell times are still kept fixed when displacing catheters. However, such evaluations are only used to frequently filter the solutions in the elitist archive. Specifically, every generation, Algorithm 1 is used; outside of the elitist archive, no robust evaluations over catheter displacements are performed. This way, some robustness of the treatment plans to catheter displacements is taken into account, without resulting in too optimistic/pessimistic fronts or clinically infeasible running times. It should be said that in MO-RV-GOMEA, the elitist archive plays a role in providing parent solutions, so the impact of only filtering the elitist archive for robustness this way is potentially larger than for other EAs that employ elitist archives.

**Algorithm 1:** Filtering of the elitist archive

```

1 Let  $(X, Y)$  be the representation of a catheter position.
2 Make a backup of the catheter positions in the elitist archive.
3 for  $m=1, \dots, 100$  do
4   for all catheters  $i$  do
5     Sample  $\theta$  uniformly in  $[0, 2\pi]$ .
6     for all solutions  $j$  in the elitist archive do
7        $X(j) += 1\text{mm} \cdot \cos(\theta)$ .
8        $Y(j) += 1\text{mm} \cdot \sin(\theta)$ .
9       Apply boundary repair if necessary.
10    end
11  end
12  Evaluate the elitist archive.
13  Restore the backup of the catheter positions in the elitist archive.
14 end
15 for all solutions  $j$  in the elitist archive do
16   Determine the Nadir point of  $j$  of the 100 evaluations.
17 end
18 Filter the elitist archive based on the Nadir points: solutions for which its Nadir
    point is dominated by the Nadir point of another solution, are removed from
    the elitist archive.

```

## 4 Experiments

In our experiments, we simulate the workflow of clinical practice, including the sequential decision-making steps. The goal is to see whether with our new approach, good plans can still be obtained and, possibly more importantly, whether the predicted quality of plans in the first stage is a realistic representation of plans obtained in the second stage. If so, a properly informed decision can be made about the number of catheters to use for a particular patient.

A simulation of the workflow starts with catheter position optimization. The running time is limited to 15 min. After catheter position optimization, a single treatment plan is selected from the front, with the highest quality in terms of  $\min\{\text{LCI}, \text{LSI}\}$ . This quality is defined as  $L$ :

$$L := \max_{\text{plans } j \text{ in front}} \{\min\{\text{LCI}_j, \text{LSI}_j\}\}. \quad (4)$$

Subsequently, dwell time optimization is performed again separately. The running time is limited to 6 min for the original dwell time optimization, and 15 min for the robust dwell time optimization. We use larger runtimes here than strictly needed in clinical practice because we want to observe also the convergence properties of the EA. Each simulated workflow is applied to the data of 3 patient cases for 16, 10, and 4 catheters. Due to the randomness in the EA and the catheter displacements, 10 runs are performed of each simulated workflow.

Three approaches are compared. Catheter position optimization is always followed by catheter displacements and robust dwell time optimization over organ reconstructions. The first approach uses the original catheter position optimization, where no uncertainties are considered at all. The second approach uses the robust catheter position optimization over only organ reconstructions. The third approach uses the robust catheter position optimization over both organ reconstructions and catheter displacements, using elitist archive filtering as in Algorithm 1.

The difference between the results of catheter position optimization and dwell time optimization is tested with a paired samples t-test on  $L$  for each of the patients, numbers of catheters, and versions of catheter position optimization separately, whereby the difference was considered to be statistically significant if  $p < 0.00185$ . This includes a Bonferroni correction for 27 test, i.e.,  $p < 0.05/27$ .

To study the convergence of MO-RV-GOMEA, we use the well-known hypervolume metric [12], i.e., the area in the bi-objective space that is covered by the front and a so-called *reference point*. Here, we choose the reference point  $(-30, -30)$  and only consider solutions in the front dominating this point.

For all code, a GPU-acceleration was implemented in CUDA (NVIDIA Corporation, Toolkit v8.0.61), based on previous work [2]. Optimization was performed on an NVIDIA Titan Xp, which contained 12 GB of memory.

## 5 Results

The results of the original catheter position optimization are shown in Fig. 2. The part of the objective space where all aims in the clinical protocol are

satisfied, i.e.  $LCI > 0$  and  $LSI > 0$ , is highlighted. The influence magnitude of the uncertainties depends on the patient and the number of catheters. Except for patient 2 with 16 catheters, there is a statistically significant difference between the catheter position and dwell time fronts. This means that the catheter position fronts are not realistic, as they are higher than what is obtained when taking into account the uncertainties. This shows the need for robust optimization.

The results of robust catheter position optimization over only organ reconstruction settings are shown in Fig. 3. For 16 catheters, for all patients, there is no statistically significant difference between the catheter position and dwell time fronts. Hence, in these cases, only robust optimization over organ reconstruction settings is needed. It should be noted that the catheter position fronts have dropped towards the dwell time fronts, but the dwell time fronts themselves did not improve. Hence, taking into account organ reconstruction settings during catheter position optimization results in more realistic fronts, but not necessarily in better catheter positions.

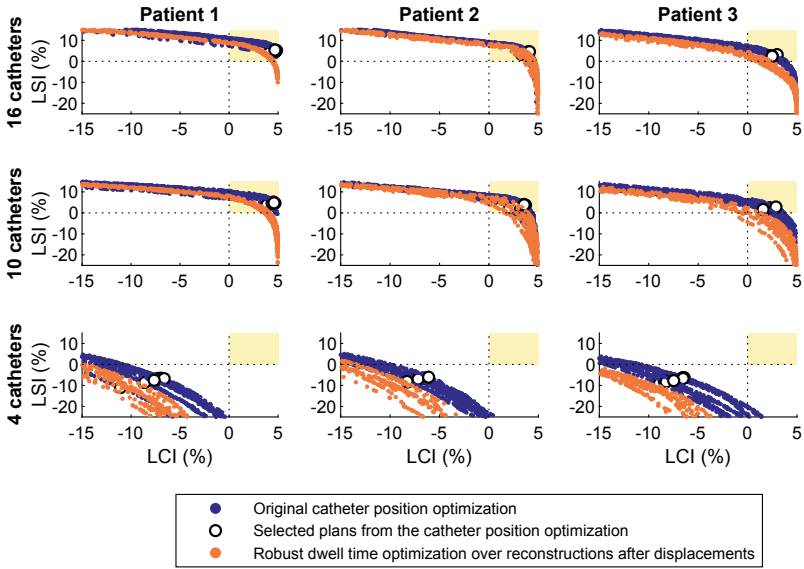
For patient 1 with 4 catheters, and for patient 2 with 10 and 4 catheters, there is still a statistically significant difference between the catheter position and dwell time fronts. Hence, in these cases, taking into account only organ reconstruction settings is not sufficient to also obtain robustness to catheter displacements. This shows the need for robust optimization over both uncertainties.

The results of robust catheter position optimization over both organ reconstruction settings and catheter displacements are shown in Fig. 4. For all patients and numbers of catheters, there is no statistically significant difference between the catheter position and dwell time fronts. Hence, the catheter position fronts are now realistic. It can be seen that with 16 catheters, for all patients, plans exist that satisfy all constraints in the clinical protocol (i.e.,  $LCI > 0$  and  $LSI > 0$ ). This is sometimes the case for 10 catheters, and never for 4 catheters.

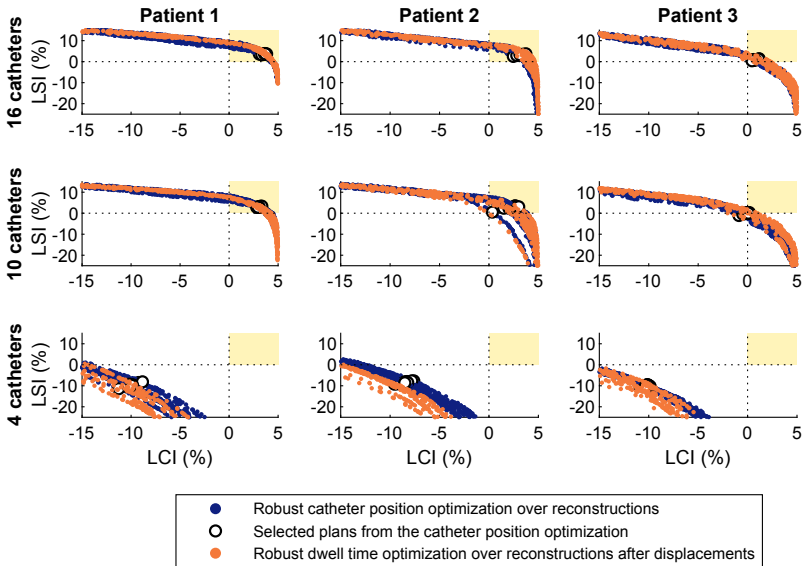
Finally, the hypervolume values of the fronts of the three types of catheter position optimization over time are shown in Fig. 5. For these patients, even with robust optimization, convergence is still achieved quickly, indicating that in clinical practice we may very well use only 5 min instead of 15, which is clinically acceptable.

## 6 Discussion

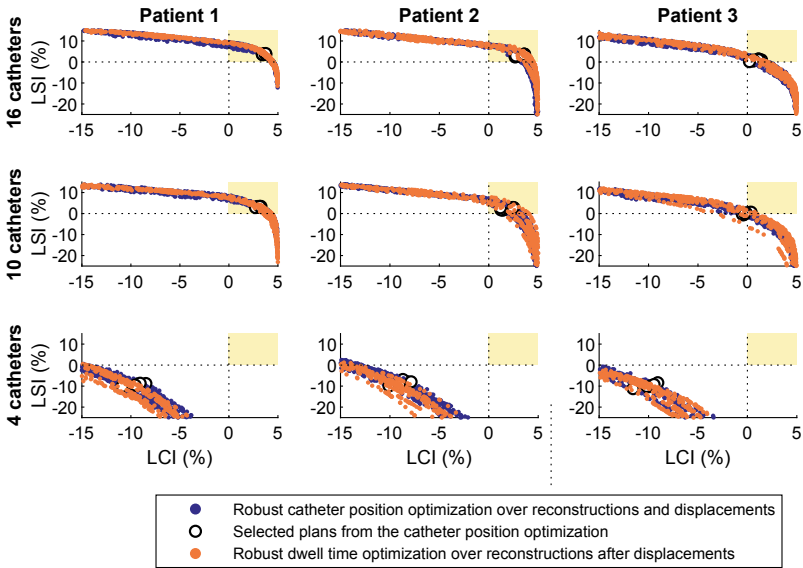
In this paper, we introduced preclinical work on robust optimization for high-dose-rate prostate brachytherapy. By performing robust optimization over both organ reconstruction settings and catheter displacements, the catheter position and dwell time fronts obtained in the first and second stages of the sequential decision-making process become virtually the same. Hence, decisions based on the catheter position fronts are now more representative of the resulting dwell time fronts. The larger part of this robustness appears to be due to the robust optimization over organ reconstructions, rather than over catheter displacements. It should still be studied whether this also holds for different patients and numbers of catheters.



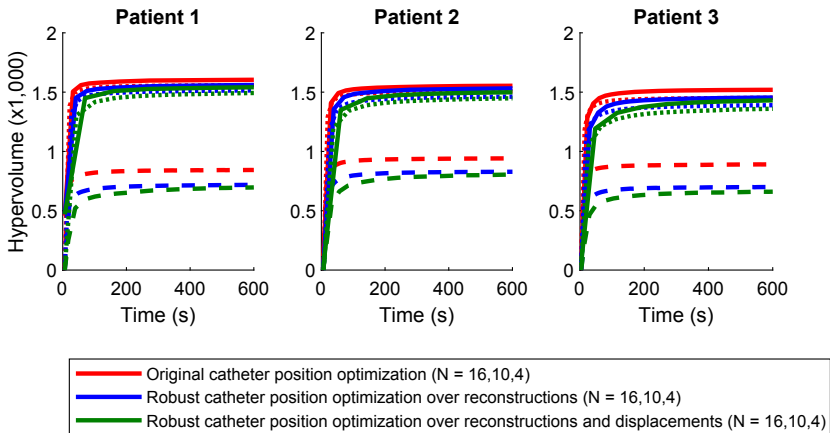
**Fig. 2.** The original (non-robust) catheter position optimization (blue), for selected plans (white circles) followed by catheter displacements and robust dwell time optimization over organ reconstruction settings (orange). Ten runs are shown. (Color figure online)



**Fig. 3.** The robust catheter position optimization over organ reconstruction settings (blue), for selected plans (white circles) followed by catheter displacements and robust dwell time optimization over organ reconstruction settings (orange). Ten runs are shown. (Color figure online)



**Fig. 4.** The robust catheter position optimization over organ reconstruction settings and catheter displacements (blue), for selected plans (white circles) followed by catheter displacements and robust dwell time optimization over organ reconstruction settings (orange). Ten runs are shown. (Color figure online)



**Fig. 5.** The hypervolume values of the fronts of the three types of catheter position optimization (colors) over time. For all patients and numbers of catheters (16 is solid, 10 is dotted, 4 is dashed), the average over ten runs is shown. (Color figure online)

In the simulated workflow, only a single set of catheter positions was selected from each catheter position front. After random catheter displacements, dwell time optimization was sufficient to obtain a front of equally good plans again. Combined with the fact that different catheter position configurations are indeed obtained along the front of the first-stage optimization, this suggests that the problem of robust catheter position optimization itself is highly redundant, e.g. due to many (almost) equally good local optima. Arguably, positioning itself could be considered to be single-objective: the objective of maximizing  $\min\{\text{LCI}, \text{LSI}\}$  would have been sufficient. However, a more in-depth analysis with physicians is needed of the different catheter position configurations that are obtained to see if there are any other reasons to deviate from this.

The proposed techniques for robust optimization are more general than this optimization method (MO-RV-GOMEA) or these uncertainties (organ reconstruction settings and catheter displacements). Besides the generality of the elitist archive filtering for sequential multi-objective decision making under uncertainty, it is for instance likely that the technique for re-using intersections of organs will also work for different uncertainties related to organ shape and catheter positions (such as uncertainties in delineations [1] and catheter angles [5]). This could be explored in future work.

## 7 Conclusion

We showed how a recently introduced state-of-the-art evolutionary bi-objective optimization approach for high-dose-rate prostate brachytherapy can be extended to include robust optimization, without requiring a prohibitively large running time when optimized with MO-RV-GOMEA. Two types of uncertainty were considered: one with a fixed set of scenario's, and one with a stochastic component. Using a different approach for each type of uncertainty, both were included directly in the optimization. The results show that more realistic fronts of catheter position optimization can now be obtained. This way, the optimization can be used more reliably in clinical practice as a basis for making such important clinical decisions as how many catheters to use for a particular patient and where to place them. Moreover, additional insights into the optimization can now be obtained. Specifically, we have learned that a promising approach that may well improve run time further may be to robustly optimize catheter positions single-objectively, by optimizing the minimum of the two objectives in the original optimization model.

**Acknowledgements.** This work is part of the research program IPPSI-TA, which has project number 628.006.003 and is financially supported by the Netherlands Research Council (NWO) and Elekta AB (Stockholm, Sweden). The authors gratefully acknowledge the support of the NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

## References

1. Balvert, M., den Hertog, D., Hoffmann, A.L.: Robust optimization of dose-volume metrics for prostate HDR-brachytherapy incorporating target and OAR volume delineation uncertainties. *INFORMS J. Comput.* **31**(1), 100–114 (2019)
2. Bouter, A., Alderliesten, T., Pieters, B.R., Bel, A., Niatsetski, Y., Bosman, P.A.N.: GPU-accelerated bi-objective treatment planning for prostate high-dose-rate brachytherapy. *Med. Phys.* **46**(9), 3776–3787 (2019)
3. Deb, K.: Multi-objective optimisation using evolutionary algorithms: an introduction. In: Wang, L., Ng, A., Deb, K. (eds.) *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pp. 3–34. Springer, London (2011). [https://doi.org/10.1007/978-0-85729-652-8\\_1](https://doi.org/10.1007/978-0-85729-652-8_1)
4. Dinkla, A.M., et al.: A comparison of inverse optimization algorithms for HDR/PDR prostate brachytherapy treatment planning. *Brachytherapy* **14**(2), 279–288 (2015)
5. Gorissen, B.L.: Practical robust optimization techniques and improved inverse planning of HDR brachytherapy. Tilburg University, School of Economics and Management, Tech. rep. (2014)
6. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* **11**(1), 1–18 (2003)
7. Karabis, A., Belotti, P., Baltas, D.: Optimization of catheter position and dwell time in prostate HDR brachytherapy using HIPO and linear programming. In: Dössel, O., Schlegel, W.C. (eds.) *World Congress on Medical Physics and Biomedical Engineering*, pp. 612–615. Springer, Heidelberg (2009). <https://doi.org/10.1007/978-3-642-03474-9>
8. Lessard, E., Pouliot, J.: Inverse planning anatomy-based dose optimization for HDR-brachytherapy of the prostate using fast simulated annealing algorithm and dedicated objective function. *Med. Phys.* **28**(5), 773–779 (2001)
9. Luong, N.H., Alderliesten, T., Bel, A., Niatsetski, Y., Bosman, P.A.N.: Application and benchmarking of multi-objective evolutionary algorithms on high-dose-rate brachytherapy planning for prostate cancer treatment. *Swarm Evol. Comput.* **40**, 37–52 (2018)
10. van der Meer, M.C., et al.: Sensitivity of dose-volume indices to computation settings in high-dose-rate prostate brachytherapy treatment plan evaluation. *J. Appl. Clin. Med. Phys.* **20**(4), 66–74 (2019)
11. van der Meer, M.C., Pieters, B.R., Niatsetski, Y., Alderliesten, T., Bel, A., Bosman, P.A.N.: Better and faster catheter position optimization in HDR brachytherapy for prostate cancer using multi-objective real-valued GOMEA. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1387–1394 (2018)
12. 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>





# A Hybrid Evolutionary Algorithm for Reliable Facility Location Problem

Han Zhang<sup>✉</sup>, Jialin Liu<sup>✉</sup>, and Xin Yao<sup>✉</sup>

Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation,  
Department of Computer Science and Engineering, Southern University of Science  
and Technology, Shenzhen 518055, China

11849181@mail.sustech.edu.cn, {liujl,xiny}@sustech.edu.cn

**Abstract.** The reliable facility location problem (RFLP) is an important research topic of operational research and plays a vital role in the decision-making and management of modern supply chain and logistics. Through solving RFLP, the decision-maker can obtain reliable location decisions under the risk of facilities' disruptions or failures. In this paper, we propose a novel model for the RFLP. Instead of assuming allocating a fixed number of facilities to each customer as in the existing works, we set the number of allocated facilities as an independent variable in our proposed model, which makes our model more close to the scenarios in real life but more difficult to be solved by traditional methods. To handle it, we propose EAMLS, a hybrid evolutionary algorithm, which combines a memorable local search (MLS) method and an evolutionary algorithm (EA). Additionally, a novel metric called l3-value is proposed to assist the analysis of the algorithm's convergence speed and exam the process of evolution. The experimental results show the effectiveness and superior performance of our EAMLS, compared to a CPLEX solver and a Genetic Algorithm (GA), on large-scale problems.

**Keywords:** Reliable facility location problem · Integer programming · Hybrid algorithm · Evolutionary algorithm · Local search

## 1 Introduction

The facility location problem aims at finding the optimal locations for facilities from a set of candidate location nodes in order to minimize the cost such as the fixed facility cost and the transposition cost, or to maximize the total revenue. In general, there are also some constraints to be considered, such as satisfying all customers' demands, etc. It is an NP-hard optimization problem [1–3] and

---

This work was supported by the National Key R&D Program of China (Grant No. 2017YFC0804003), the National Natural Science Foundation of China (Grant No. 61976111, 61906083), the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. JCYJ20190809121403553), the Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

© The Author(s) 2020

T. Bäck et al. (Eds.): PPSN 2020, LNCS 12270, pp. 454–467, 2020.

[https://doi.org/10.1007/978-3-030-58115-2\\_32](https://doi.org/10.1007/978-3-030-58115-2_32)

has attracted much attention from researchers in both the scientific community and engineering field due to its wide application in real world. The facilities could be hospitals, restaurants, post stations, bus stations, industrial plants, banks, warehouses, and distribution centers, etc. The facility location decision has high precedence in the whole logistics decisions and has a great influence on subsequent operation level decisions [4]. Daskin et al. [1] regards the location decisions as “the most critical and most difficult of the decisions needed to realize an efficient supply chain”.

In RFLP, the facility is not always available all the time [1]. One or more of them may not work from time to time because of disruptions, examples include natural disasters, inclement weather, destruction of facilities by fire or flood, expiration of the contract, and any other force majeure factors. In such a situation, these are facility “failures”. The failures of the facilities will result in excessive transportation costs because the customers that were considered to be served by them must be served by other, usually more distant, facilities [1]. Therefore, by solving RFLP, we can get a location decision which can ensure a certain level of reliability to guarantee customers can get service when facilities’ failures occur.

Many models have been proposed for RFLP, in which all kinds of factors were taken into account and many of them are formulated for specific applications in real life. In addition, large-scale RFLP problems have rarely been considered. The algorithms studied in literature were mainly tested on problems of small size.

This paper focuses on two aspects: the problem formulation and the algorithm. Based on the work of [5,6], we propose a new reliable facility location-allocation problem (RFLP) formulation, which does not fix the number of allocated facilities to each customer as a constant and is more close to reality. The resulted model is a nonlinear 0–1 integer programming model which is more complicated for traditional methods. In this paper, a hybrid evolutionary algorithm called EAMLS is proposed to solve it. EAMLS combines a memorable local search method with an evolutionary algorithm, which has a good performance on both small-scale and large-scale problems considered in this paper. It is worth mentioning that the instances used in our experiments are much larger than the ones used in previous work. Furthermore, a convergence metric  $l3$ -value is proposed for analyzing the algorithm and observing the evolutionary process.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work of RFLP. In Sect. 3, our new RFLP formulation is introduced. We proposed a hybrid evolutionary algorithm EAMLS in Sect. 4. Section 5 presents computational studies, and Sect. 6 concludes.

## 2 Related Work

By solving a specific RFLP, decision-makers expect to get a robust location decision which is still economical when some facilities fail under various disruptions. The research can be divided into two categories according to the method used to handle facility failure or ensure reliability.

Some works [7–9] use a disruptive scenarios approach to describe facility failure. In this approach, scenarios contain facility failure information, e.g., simultaneously disrupted facility sites, modified customer demands, and facility costs, etc. The disruptive scenarios approach can describe the facility failure information well, but it usually requires plenty of scenarios to cover different disruptive situations, which implies large computational cost, especially for large-scale problems.

Another approach to ensure reliability is to allocate two or more facilities to serve each customer [5, 6, 10, 11]. In this approach, the method for reliability is intuitive and easy to understand. Both a location decision (which contains how many facilities needed to build and where to build them) and an allocation decision (which shows how to allocate facilities to serve customers) are determined before the occurrences of facilities’ disruptions/failures.

Some RFLP models have been proposed, e.g., models proposed by Li et al. [5] and Snyder and Darskin [6]. Table 1 summarizes the notations used in the models.

**Table 1.** Description of notations.

Notations	Description	Notations	Description
$I$	the set of customers, index by $i$ ;	$m$	# of facilities allocated for each customer;
$J$	the set of candidate location sites, index by $j$ ;	$p$	the facility failure probability;
$NF$	the set of candidate location sites that will not fail;	$f_i$	the fix cost of $j$ ;
$F$	the set of candidate location sites that may fail;	$\alpha$	weighted parameter;
$c_{ij}$	the cost of per unit demand shipped from $j$ to $i$ ;	$h_i$	the demands of customer $i$ ;

Besides, there are two sets of decision variables: location decision variables ( $\mathbf{X}$ ) and allocation decision variables ( $\mathbf{Y}$ ):

$$X_j = \begin{cases} 1, & \text{if candidate location site } j \text{ is selected;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$Y_{ijr} = \begin{cases} 1, & \text{if } j \text{ is allocated as the level-}r \text{ facility to serve } i; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In Eq. (2), the “level- $r$ ” facility  $j$  for customer  $i$  means the facility  $j$  will provide service only when the front  $r$  allocated facilities (from level-0 to level- $(r-1)$ ) fail.

A classical RFLP model in [6] is as follows.

$$\text{Min } \alpha w_1 + (1 - \alpha)w_2 \tag{3}$$

Subject to:

$$w_1 = \sum_{j \in J} f_j X_j + \sum_{i \in I} \sum_{j \in J} h_i c_{ij} Y_{ij0} \tag{4}$$

$$w_2 = \sum_{i \in I} h_i \left[ \sum_{j \in NF} \sum_{r=0}^{m-1} c_{ij} p^r Y_{ijr} + \sum_{j \in F} \sum_{r=0}^{m-1} c_{ij} p^r (1-p) Y_{ijr} \right] \tag{5}$$

$$\sum_{j \in J} Y_{ijr} + \sum_{j \in NF} \sum_{t=0}^{r-1} Y_{ijt} = 1 \quad \forall i \in I, r = 0, \dots, m-1 \tag{6}$$

$$Y_{ijr} \leq X_j \quad \forall i \in I, j \in J, r = 0, \dots, m-1 \tag{7}$$

$$\sum_{r=0}^{m-1} Y_{ijr} \leq 1 \quad \forall i \in I, \forall j \in J \tag{8}$$

$$m = |J| \tag{9}$$

$$X_u = 1 \tag{10}$$

$$X_j \in \{0, 1\} \quad \forall j \in J \tag{11}$$

$$Y_{ijr} \in \{0, 1\} \quad \forall i \in I; \forall j \in J; r = 0, \dots, m-1 \tag{12}$$

In this model, there are two objectives in the objective function,  $w_1$  is the operating cost and  $w_2$  is the expected failure cost. The objective of the model is to minimize the weighted sum of the two objectives. Besides, there is an emergency facility  $u$  which will always be selected and not fail, and all customers can get service from it.

Several shortcomings are observed in the literature:

(1) The number of facilities allocated to each customer (i.e.,  $m$  in Eq. (9)) is fixed in models of most literature, e.g.,  $m = 2$  (i.e.,  $Y_{ij0}$  and  $Y_{ij1}$ ) in [5] and  $m = |J|$  in [6]. One issue of this allocation setting is the determination of an appropriate value of  $m$ . If  $m$  is bigger than the number of selected candidate location sites, i.e.,  $\sum_{j \in J} X_j$ , it is not in line with the actual situation because we cannot allocate nonexistent facilities to customers. If we set the value of  $m$  smaller than  $\sum_{j \in J} X_j$ , the value of  $\sum_{j \in J} X_j$  is changed during the exploration in solution space, therefore it is hard for us to set a suitable  $m$  value. If we set  $m = 2$  directly, which means allocate just one primary facility and one backup facility to serve each customer, the reliability is a bit weak intuitively.

(2) To our best knowledge, there is a lack of research on the large-scale problem. The largest problem instance in the related research is 150-node and the optimization solver such as CPLEX can find near-optimal or even optimal solutions for the problem.

(3) There is a lack of research on the algorithm which can solve the large-scale problems efficiently as well.

Correspondingly, this paper:

(1) constructs a new formulation in which a non-fixed allocation setting, i.e.,  $m = \sum_{j \in J} X_j$ , is used;

(2) proposes a hybrid evolutionary algorithm EAMLS which combines a local search method with an evolutionary algorithm and performs well on both small-scale and large-scale problems;

(3) performs experimental studies on large-scale problems whose scale is much larger than any related literature;

(4) proposes a convergence metric l3-value to help observe the evolutionary process, adjust parameters and further improve the algorithm.

### 3 Problem Formulation

We propose a new RFLP formulation in which we set the number of allocated facilities to each customer as an variable instead of a fixed constant.

The mathematical formulation of our model is as follows, formulated based on [5,6]. The decision variables are defined by Eqs. (1) and (2).

$$\text{Min} \sum_{j \in J} f_j X_j + \alpha \sum_{i \in I} \sum_{j \in J} \sum_{r=0}^{m-1} h_i c_{ij} p^r (1-p) Y_{ijr} \tag{13}$$

Subject to:

$$m = \sum_{j \in J} X_j \tag{14}$$

$$m \geq 2 \tag{15}$$

$$\sum_{j \in J} Y_{ijr} = 1 \quad \forall i \in I; r = 0, \dots, m-1 \tag{16}$$

$$\sum_{r=0}^{m-1} Y_{ijr} \leq X_j \quad \forall i \in I, \forall j \in J \tag{17}$$

$$X_j \in \{0, 1\} \quad \forall j \in J \tag{18}$$

$$Y_{ijr} \in \{0, 1\} \quad \forall i \in I; \forall j \in J; r = 0, \dots, m-1 \tag{19}$$

The objective function of the model is to minimize the total cost associate with facilities construction (i.e., the term  $\sum_{j \in J} f_j X_j$ ) and transportation between the facilities and customers (i.e., the term  $\sum_{i \in I} \sum_{j \in J} \sum_{r=0}^{m-1} h_i c_{ij} p^r (1-p) Y_{ijr}$ ).

Constraint (14) makes the number of facilities allocated to each customer (i.e.,  $m$ ) a variable and its value is related to location decision variables (i.e.,  $\mathbf{X}$ ). Constraint (15) represents at least two facilities are constructed to ensure reliability. Constraint (16) assures only one facility can be the level- $r$  supplier of customer  $i$ . Constraint (17) means candidate location site  $j$  can be allocated to customer as a supplier only when it is selected. Constraint (18) and (19) are standard integrality constraints.

Compared with classical models shown in Sect. 2, the significant difference in our model is the new non-fixed facility allocation setting, i.e., constraint (14). In our model, the value of  $m$  is not fixed but varies with decision variables  $\mathbf{X}$ , therefore it is more realistic, ensures reliability, but makes our model much more complex and difficult to solve by traditional methods as well.

## 4 A Hybrid Evolutionary Algorithm: EAMLS

This paper develops a new hybrid evolutionary algorithm EAMLS (Evolutionary Algorithm with Memorable Local Search) which combines a memorable local search method and an EA, and a convergence metric l3-value is proposed. In this section, the structure of EAMLS is explained first, then the design of operators of the Genetic Algorithm (GA) and EAMLS is introduced. Finally, the details of l3-value are described.

### 4.1 EAMLS

Algorithm 1 is the pseudo-code of EAMLS. Compared with the GA, the main characters of EAMLS contain: (1) no crossover operation; (2) population size self-adaptation; (3) the combination of a memorable local search (MLS) and EA; and (4) the adoption of convergence metric l3-value.

In Algorithm 1, variable *allNeighborInds* stores all non-repeating neighborhood individuals generated by MLS before current generation and is updated at the end of every generation (Algorithm 1, Line 2 and Line 13). In the evolutionary process, a new population is generated from the current population after mutation, MLS, and survival selection (Algorithm 1, Lines 5–8), and convergence metric l3-value is calculated (Algorithm 1, Line 9). If l3-value is bigger than a pre-set threshold  $\beta$ , population size is increased by a pre-set step size  $p$  (Algorithm 1, Lines 10–12). The description of the l3-value will be shown in Sect. 4.3.

Algorithm 2 is the pseudo-code of the memorable local search (MLS). First, we will introduce the definition of the neighborhood. The neighborhood of an individual is the set of individuals whose Hamming distance is 1 from that

---

**Algorithm 1.** Evolutionary Algorithm with Memorable Local Search.

**Input:**  $G$ : number of generations;  $\mu$ : population size;  $l$ : individual length;  $m$ : mutation rate;  $\beta$ : threshold of l3-value;  $p$ : step size of population self-adaptation;

**Output:** *bestSol*: the best individual in the final population;

```

1: initPop  $\leftarrow$  initializePop( $\mu, l$ );
2: allNeighborInds  $\leftarrow$  an empty set;
3: pop  $\leftarrow$  evaluatePop(initPop);
4: for  $g = 1$  to  $G$  do
5:   popAfterMutation  $\leftarrow$  mutation(pop,  $m$ );
6:   offspring  $\leftarrow$  evaluatePop(popAfterMutation);
7:   offspringLS  $\leftarrow$  memorableLocalSearch(pop, offspring);
8:   pop  $\leftarrow$  survival(pop, offspring, offspringLS,  $\mu$ );
9:   l3-value  $\leftarrow$  getl3Value(pop, allNeighborInds);
10:  if l3-value  $>$   $\beta$  then
11:     $\mu \leftarrow \mu + p$ ;
12:  end if
13:  add offspringLS to allNeighborInds;
14: end for
15: bestSol  $\leftarrow$  selectBestIndividual(pop)
16: Return bestSol

```

---

---

**Algorithm 2.** Memorable Local Search.

---

**Input:** *pop*: the parent population; *offspring*: the child population generated after mutation; *n*: # of individuals which need to check whether to do local search; *indLSed*: the set of individuals which have already done local search before this generation;

**Output:** *offspring<sub>LS</sub>*: the population generated by local search;

- 1: *offspring<sub>LS</sub>*  $\leftarrow$  an empty set;
- 2: *parentPop*  $\leftarrow$  combine *pop* and *offspring*;
- 3: *sortedParentPop*  $\leftarrow$  sort *parentPop* by fitness increasing order;
- 4: *i*  $\leftarrow$  0;
- 5: **for** *j*  $\leftarrow$  1 to *len(sortedParentPop)* **do**
- 6:     **if** *sortedParentPop*[*j*] not in *indLSed* **then**
- 7:         *neighborInds*  $\leftarrow$  *generateNeighbor(sortedParentPop*[*j*]);
- 8:         add *neighborInds* to *offspring<sub>LS</sub>*;
- 9:         *i*  $\leftarrow$  *i* + 1;
- 10:     **if** *i* > *n* **then**
- 11:         break;
- 12:     **end if**
- 13: **end if**
- 14: **end for**
- 15: **Return** *offspring<sub>LS</sub>*

---

individual. In MLS, sort  $(\mu + \lambda)$  population (variable *sortedParentPop* in Algorithm 2) in decreasing order, i.e., good individuals are in the front. Then check individuals one by one in sorted  $(\mu + \lambda)$  population whether it has been local-searched before this generation, and do local-search for those have not been local-searched (Lines 5–7 in Algorithm 2. It looks like that the algorithm remembers all local-searched individuals and that’s why we name it Memorable Local Search). Exit the loop until the number of new individuals which have been local-searched in this generation reaches *n* (Lines 9–12 in Algorithm 2).

## 4.2 Operator Design of GA and EAMLS

In Sect. 5, we use a GA for comparison. Here some operators’ design for GA and EAMLS is as follows<sup>1</sup>:

**Representation.** This paper uses binary representation. Every bit represents a location decision variable  $X_j, j \in J$ .

**Population Initialization.** Stochastic initialization is used in GA and EAMLS. Every gene of an individual takes 0 or 1 with equal probability.

**Fitness Function.** In general, the bigger the fitness value is, the better the individual will be. Therefore, the reciprocal of the objective value of the individual is used as the fitness function.

---

<sup>1</sup> If there is no special statement, that operator is adopted in both GA and EAMLS.

**Selection Operator.** In GA, roulette wheel selection is used to select parents to do crossover operation.

**Crossover Operator.** In GA, a one-point crossover operator is used. For two parent individuals selected by the selection operator, do crossover operation according to a pre-set crossover rate.

**Mutation Operator.** The bit-flipping mutation is used in GA and EAMLS. During mutation, every gene/bit of one individual mutates with a pre-set mutation rate.

**Survival Selection Strategy.** We adopt  $(\mu + \lambda)$  strategy to select next generation population from  $(\mu + \lambda)$  population, i.e., the mixed population of the current generation population and the offspring.

**Repair Strategy.** Repair strategy is working when there are individuals which do not satisfy the constraint (15). For an individual needed repair, check every gene in ascending order of fixed cost and change the gene with 0-value to 1 until the individual satisfies the constraint (15).

**How to Determine Y.** For one customer, the selected candidate locations (i.e., locations whose  $X_j = 1$ ) are allocated to it in ascending order of distance, which has been proved the optimal allocation pattern under a certain solution  $\mathbf{X}$  [6] and can satisfy the constraints (12), (13), and (15).

### 4.3 Convergence Metric l3-Value

In order to observe the evolutionary process, a convergence metric l3-value is proposed.

Algorithm 3 is the pseudo-code of the calculation method of l3-value. The new population generated after survival selection is checked, and the number of individuals which also belong to the set *allNeighborInds* is counted (Lines 2–6 in Algorithm 3). Then we calculate the proportion of these individuals in the population as l3-value (Line 7 in Algorithm 3). l3-value can be used to measure the convergence during the evolutionary process. The bigger the l3-value is, the stronger the evolution converges.

---

**Algorithm 3.** Function *getl3Value()*.

---

**Input:** *pop*: the new population after survival selection; *allNeighborInds*: the set of all individuals generated by memorable local search before this generation;

**Output:** l3-value;

```

1: num ← 0;
2: for ind ∈ pop do
3:   if ind ∈ allNeighborInds then
4:     num ← num + 1;
5:   end if
6: end for
7: l3-value ← num/len(pop);
8: Return l3-value

```

---



## 5 Computational Studies

Because this paper proposes a new problem, and there are not any algorithms like EAMLS can be used to compare directly, we compare EAMLS with a GA and CPLEX (a commercial optimization solver of IBM) on two models:  $m=2$  and  $m = \sum_{j \in J} X_j$  models. The difference between the two models is the allocation setting. In the  $m=2$  model, the number of facilities allocated to each customer, i.e.  $m$ , is fixed to 2, which is adopted in much literature. The  $m = \sum_{j \in J} X_j$  model is proposed by us in this paper and  $m$  varies with decision variables  $\mathbf{X}$  during the search process. Section 5.1 shows the experimental design, including instances generation, parameters setting, and experimental environment. The experiments and results of the  $m=2$  and  $m = \sum_{j \in J} X_j$  models are presented in Sect. 5.2. Analyses and discussions are given in Sect. 5.3.

### 5.1 Experimental Design

**Instance Generation.** This paper generates problem instances uniformly at random on different scales. The parameters used to generate instances are shown in Table 2. There are eight 10-node instances, eight 50-node instances, eight 100-node instances, and four 600-node instances.

**Table 2.** Parameters used in instances generation

Parameters	Ranges
Candidate location coordinate	[0,1]
Customer demands	{0,1,...,1000}
Fixed cost of facility	{500,501,...,1500}
Facility failure probability	0.05

**Parameter Setting of Algorithms.** Some parameters' values of GA and EAMLS are shown in Table 3. Table 4 presents the generation number and population size of GA and EAMLS, which associate with the scale of problem instances. The values of parameters in Tables 3 and 4 are chosen arbitrarily on the basis of meeting the following conditions: (1) EAMLS converges at the end of evolution; (2) the number of fitness evaluations (FEs) of GA is not lower than EAMLS. Besides, the default parameters of CPLEX are used.

**Experimental Environment.** The algorithms are implemented in Python 3.7 and run on Dell R370 server which has 2x Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20 GHz CPU, 128G RAM, and CentOS 7.6 operating system.

**Statistical Test.** We use the Wilcoxon sign rank test to determine whether the results between EAMLS and other methods have statistically significant differences. The Wilcoxon sign rank test is a non-parameter test which is suitable for two related or matched samples and compares data in pair, hence it is suitable to use here.

**Table 3.** Some parameters of GA and EAMLS

Parameters	Value
Crossover rate for GA, $c$	0.9
Mutation rate, $m$	0.1
# Local search individual, $n$	10
l3-value threshold, $\beta$	0.8
Step size of population self-adaption, $p$	100

**Table 4.** Parameters associate with instance size

Instance scale (# nodes)	GA		EAMLS	
	# Generation	Population size	# Generation	Population size
10	60	30	10	20
50	200	200	20	20
100	400	200	50	100
600	4600	200	250	200

**5.2 Experiments on the  $m = 2$  and  $m = \sum_{j \in J} X_j$  Models**

For the  $m = 2$  model, We compare EAMLS with the GA and CPLEX on small-scale (10-node), mid-scale (100-node), and large-scale (600-node) instances. There are 30 runs on small and mid-scale instances and 10 runs on large-scale instances because of time. The computational results are shown in Table 5.

For the  $m = \sum_{j \in J} X_j$  model, we compare EAMLS with the GA and CPLEX on 50 and 100-node instances, and there are 30 runs on each instance. Table 6 is the computational results.

**5.3 Analyses and Discussions**

We compare GA, CPLEX, and EAMLS on different scale (10, 100, and 600-node) problem instances for  $m = 2$  model whose allocation setting is often used in literature, and the experimental results are shown in Table 5. Experimental results on 50 and 100-node instances of the new complicated  $m = \sum_{j \in J} X_j$  model are presented in Table 6.

For  $m = 2$  model, from Table 5, we can see that CPLEX performs the best on both solution quality and time for small and mid-scale (10 and 100-node) instances. EAMLS can find solutions as good as CPLEX but need more time. Although CPLEX can solve small and mid-scale instances fast, it needs more RAM space as the problem scale increases. For large-scale problem (600-node) instances, EAMLS can find better solutions in less time compared with GA, while the CPLEX cannot find a solution.

The new  $m = \sum_{j \in J} X_j$  model is more complicated to solve, especially for CPLEX. Table 6 demonstrates that the performance of EAMLS is better than GA and CPLEX on both solution quality and time.

**Table 5.** Computational results on  $m = 2$  model 10 (30 runs), 100 (30 runs), and 600 (10 runs)-node instances. AOV is Average Objective Value. OR is the Optimal Rate and calculated by  $(\# \text{ runs which finding the optimal solution})/(\# \text{ all runs})$ . Gap is calculated by  $(\text{AOV}(\text{other method}) - \text{AOV}(\text{EAMLS}))/\text{AOV}(\text{EAMLS})$ . When Gap is positive, the performance of other methods is worse than EAMLS, otherwise better. The symbol “\*\*” in AOV represents the results between EAMLS and that method have statistically significant differences. The symbol “\_” represents CPLEX cannot solve the instance or the optimal solution is unknown so no results can be given.

Instance No.	GA			CPLEX			EAMLS		
	AOV	Gap (%)	Time	AOV	Gap (%)	Time	AOV	Gap (%)	Time
10-1	<b>2463.19</b>	0.00	<b>1.00</b> 1.52	<b>2463.19</b>	0.00	<b>1.00</b> <b>0.46</b>	<b>2463.19</b>	0.00	1.00 6.14
10-2	<b>2874.03</b>	0.00	<b>1.00</b> 1.51	<b>2874.03</b>	0.00	<b>1.00</b> <b>0.41</b>	<b>2874.03</b>	0.00	1.00 5.46
10-3	<b>2623.35</b>	0.00	<b>1.00</b> 1.74	<b>2623.35</b>	0.00	<b>1.00</b> <b>0.66</b>	<b>2623.35</b>	0.00	1.00 5.41
10-4	<b>2323.92</b>	0.00	<b>1.00</b> 1.93	<b>2323.92</b>	0.00	<b>1.00</b> <b>0.48</b>	<b>2323.92</b>	0.00	1.00 5.86
10-5	<b>2917.87</b>	0.00	<b>1.00</b> 2.48	<b>2917.87</b>	0.00	<b>1.00</b> <b>0.50</b>	<b>2917.87</b>	0.00	1.00 5.71
10-6	<b>3149.31</b>	0.00	<b>1.00</b> 2.72	<b>3149.31</b>	0.00	<b>1.00</b> <b>0.41</b>	<b>3149.31</b>	0.00	1.00 5.59
10-7	<b>3324.98</b>	0.00	<b>1.00</b> 2.39	<b>3324.98</b>	0.00	<b>1.00</b> <b>0.58</b>	<b>3324.98</b>	0.00	1.00 5.64
10-8	<b>3165.87</b>	0.00	<b>1.00</b> 2.10	<b>3165.87</b>	0.00	<b>1.00</b> <b>0.52</b>	<b>3165.87</b>	0.00	1.00 4.58
100-1	13029.83*	22.28	0.00 2374.86	<b>10645.89</b>	-0.10	<b>1.00</b> <b>14.30</b>	10656.11	0.00	0.87 1431.17
100-2	13166.44*	20.95	0.00 2375.71	<b>10885.43</b>	0.00	<b>1.00</b> <b>14.31</b>	<b>10885.43</b>	0.00	1.00 1387.01
100-3	12982.37*	16.90	0.00 2396.42	<b>11105.21</b>	0.00	<b>1.00</b> <b>14.76</b>	11105.39	0.00	0.93 1514.12
100-4	13379.41*	16.66	0.00 2388.67	<b>11468.64</b>	0.00	<b>1.00</b> <b>14.42</b>	<b>11468.64</b>	0.00	1.00 1382.91
100-5	14563.46*	16.39	0.00 2398.34	<b>12505.51</b>	-0.05	<b>1.00</b> <b>14.80</b>	12512.29	0.00	0.90 1415.63
100-6	13189.74*	17.29	0.00 2402.44	<b>11245.55</b>	0.00	<b>1.00</b> <b>14.00</b>	<b>11245.55</b>	0.00	1.00 1447.11
100-7	12841.37*	16.11	0.00 1696.85	<b>11043.70</b>	-0.15	<b>1.00</b> <b>15.49</b>	11059.89	0.00	0.90 1326.41
100-8	13886.78*	18.30	0.00 1242.25	<b>11732.46</b>	-0.05	<b>1.00</b> <b>14.94</b>	11738.83	0.00	0.87 1180.91
600-1	144896.91*	281.04	-	655420.67	-	-	<b>38026.65</b>	0.00	- <b>564432.00</b>
600-2	145508.23*	293.12	-	656832.50	-	-	<b>37013.71</b>	0.00	- <b>572568.34</b>
600-3	141486.28*	283.41	-	654632.01	-	-	<b>36902.36</b>	0.00	- <b>568824.96</b>
600-4	141256.35*	282.80	-	656656.21	-	-	<b>36900.52</b>	0.00	- <b>568546.96</b>

**Table 6.** Computational results on  $m = \sum_{j \in J} X_j$  model 50 and 100-node instances, 30 runs. AOV is Average Objective Value. Gap is calculated by  $((AOV(\text{other method}) - AOV(\text{EAMLS})) / AOV(\text{EAMLS}))$ . When Gap is positive, the performance of other methods is worse than EAMLS, otherwise better. The symbol “\*” in AOV represents the results between EAMLS and that method have statistically significant differences.

Instance No.	GA			CPLEX			EAMLS		
	AOV	Gap (%)	Time	AOV	Gap (%)	Time	AOV	Gap (%)	Time
50-1	7053.71*	0.68	719.76	12589.41*	79.69	4715.56	<b>7006.23</b>	0.00	<b>91.52</b>
50-2	7154.93	-0.13	720.49	15734.80*	119.63	4488.26	<b>7164.20</b>	0.00	<b>90.75</b>
50-3	6890.54*	0.75	713.25	12656.13*	85.06	5219.33	<b>6838.95</b>	0.00	<b>91.10</b>
50-4	7166.63	0.04	698.45	12147.92*	69.58	4702.01	<b>7163.42</b>	0.00	<b>90.04</b>
50-5	6929.29	0.03	714.86	11946.35*	72.46	5281.27	<b>6926.95</b>	0.00	<b>87.42</b>
50-6	6575.09	0.29	696.80	13284.69*	102.64	4836.45	<b>6555.87</b>	0.00	<b>90.59</b>
50-7	7162.83	0.07	685.04	12441.00*	73.81	4495.41	<b>7157.76</b>	0.00	<b>81.19</b>
50-8	7175.89*	0.26	629.19	14433.41*	101.67	4522.35	<b>7156.99</b>	0.00	<b>70.07</b>
100-1	12895.10*	20.47	3976.97	113781.45*	963.00	19451.62	<b>10703.78</b>	0.00	<b>2266.50</b>
100-2	13093.80*	19.77	3820.11	110441.89*	910.18	17159.57	<b>10932.89</b>	0.00	<b>2168.54</b>
100-3	13082.38*	17.21	2719.68	114576.21*	926.52	35836.91	<b>11161.59</b>	0.00	<b>2337.35</b>
100-4	13484.69*	17.04	2551.55	99484.65*	763.50	35129.74	<b>11521.11</b>	0.00	<b>2217.00</b>
100-5	14484.70*	15.22	2579.12	111338.15*	785.68	17249.10	<b>12570.86</b>	0.00	<b>2279.35</b>
100-6	13360.41*	18.20	2626.97	99397.46*	779.39	19433.87	<b>11302.96</b>	0.00	<b>2288.82</b>
100-7	12810.60*	15.20	2553.83	105460.22*	848.32	17271.95	<b>11120.78</b>	0.00	<b>2036.58</b>
100-8	13809.12*	17.05	2548.32	112170.76*	850.82	18667.88	<b>11797.25</b>	0.00	<b>1792.08</b>

According to the observation of computational results, we can get three features of EAMLS: (1) For small- and mid-scale problems, the solutions found by EAMLS are comparable to those found by other methods; (2) For large-scale problems, EAMLS significantly outperforms other methods; (3) EAMLS especially performs well on (a) the new complicated model and (b) large-scale problems. So why is EAMLS effective? Through combining MLS with EA and using  $\lambda$ -value to guide the population size to grow gradually, EAMLS performs a full local search while performing a global search, maintains good population diversity, as well as speeds up the convergence.

Our algorithm EAMLS performs well on large-scale problem instances of both  $m = 2$  and  $m = \sum_{j \in J} X_j$  models, and its advantage will become more apparent as the problem scale increases. However, the larger the problem, the greater the number of FEs needed for EAMLS to converge.

## 6 Conclusion

This paper proposes a new RFLP formulation in which the number of facilities allocated to each customer (i.e.,  $m$ ) is not fixed but varies with decision variables  $\mathbf{X}$ . This non-fixed allocation setting makes the model more close to scenarios in real life.

A hybrid evolutionary algorithm EAMLS (which can also be viewed as a memetic algorithm) is proposed to solve the model. Combining a memorable local search method and EA, EAMLS performs well on the new complicated model and large-scale problems considered in this paper, and its advantage will become more obvious as the problem scale increases. Besides, a convergence metric  $\lambda$ -value is proposed to analyze the algorithm's convergence speed and exam the evolutionary process.

Finally, we explore the large-scale problems of the two models. Under what conditions is a problem a large-scale problem? It is related to the model and whether the problem can be solved by the exact algorithm efficiently. For the  $m = 2$  model which allocates a fixed number of facilities to each customer as in the existing research, we solve large-scale problem instances (600-node) whose scale is much larger than other literature. For the new complicated  $m = \sum_{j \in J} X_j$  model, 100-node instances can be treated as large-scale problems because the exact algorithm or optimization solver cannot solve them effectively. And our algorithm EAMLS has good performance on large-scale problems considered in this paper.

In the future, the model which integrates various factors should be studied, and more complicated FLPs, such as dynamic FLP and FLP under uncertain environments, should be focused. Furthermore, effective meta-heuristic algorithms for large-scale problems should be studied as well.

## References

1. Daskin, M.S., Snyder, L.V., Berger, R.T.: Facility location in supply chain design. In: Langevin, A., Riopel, D. (eds.) *Logistics Systems: Design and Optimization*, pp. 39–65. Springer, Boston (2005). [https://doi.org/10.1007/0-387-24977-X\\_2](https://doi.org/10.1007/0-387-24977-X_2)
2. Farahani, R.Z., Hekmatfar, M.: *Facility Location: Concepts, Models, Algorithms and Case Studies*. Springer, New York (2009). <https://doi.org/10.1007/978-3-7908-2151-2>
3. Owen, S.H., Daskin, M.S.: Strategic facility location: a review. *Eur. J. Oper. Res.* **111**(3), 423–447 (1998)
4. Riopel, D., Langevin, A., Campbell, J.F.: The network of logistics decisions. In: Langevin, A., Riopel, D. (eds.) *Logistics Systems: Design and Optimization*, pp. 1–38. Springer, Boston (2005). [https://doi.org/10.1007/0-387-24977-X\\_1](https://doi.org/10.1007/0-387-24977-X_1)
5. Li, Q., Zeng, B., Savachkin, A.: Reliable facility location design under disruptions. *Comput. Oper. Res.* **40**(4), 901–909 (2013)
6. Snyder, L.V., Daskin, M.S.: Reliability models for facility location: the expected failure cost case. *Transp. Sci.* **39**(3), 400–416 (2005)
7. Peng, P., Snyder, L.V., Lim, A., Liu, Z.: Reliable logistics networks design with facility disruptions. *Transp. Res. Part B Methodological* **45**(8), 1190–1211 (2011)
8. Jabbarzadeh, A., Jalali Naini, S.G., Davoudpour, H., Azad, N.: Designing a supply chain network under the risk of disruptions. *Math. Prob. Eng.* **2012**, 23 pages (2012). <https://doi.org/10.1155/2012/234324>. Article ID 234324
9. Du, B., Zhou, H., Leus, R.: A two-stage robust model for a reliable p-center facility location problem. *Appl. Math. Model.* **77**, 99–114 (2020)
10. Li, Q., Savachkin, A.: A fast tabu search algorithm for the reliable P-median problem. In: Gao, D., Ruan, N., Xing, W. (eds.) *Advances in Global Optimization*, vol. 95, pp. 417–424. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-08377-3\\_41](https://doi.org/10.1007/978-3-319-08377-3_41)
11. Afify, B., Ray, S., Soeanu, A., Awasthi, A., Debbabi, M., Allouche, M.: Evolutionary learning algorithm for reliable facility location under disruption. *Expert Syst. Appl.* **115**, 223–244 (2019)

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



# **Reinforcement Learning**



# Optimality-Based Analysis of XCSF Compaction in Discrete Reinforcement Learning

Jordan T. Bishop<sup>(✉)</sup> and Marcus Gallagher

School of Information Technology and Electrical Engineering,  
The University of Queensland, Brisbane, QLD 4072, Australia  
{j.bishop,marcusg}@uq.edu.au

**Abstract.** Learning classifier systems (LCSs) are population-based predictive systems that were originally envisioned as agents to act in reinforcement learning (RL) environments. These systems can suffer from population bloat and so are amenable to compaction techniques that try to strike a balance between population size and performance. A well-studied LCS architecture is XCSF, which in the RL setting acts as a Q-function approximator. We apply XCSF to a deterministic and stochastic variant of the FrozenLake8x8 environment from OpenAI Gym, with its performance compared in terms of function approximation error and policy accuracy to the optimal Q-functions and policies produced by solving the environments via dynamic programming. We then introduce a novel compaction algorithm (Greedy Niche Mass Compaction—GNMC) and study its operation on XCSF’s trained populations. Results show that given a suitable parametrisation, GNMC preserves or even slightly improves function approximation error while yielding a significant reduction in population size. Reasonable preservation of policy accuracy also occurs, and we link this metric to the commonly used steps-to-goal metric in maze-like environments, illustrating how the metrics are complementary rather than competitive.

**Keywords:** Reinforcement learning · Learning classifier system · XCSF · Compaction

## 1 Introduction

Reinforcement learning (RL) is characterised by an agent learning a behavioural policy in an environment by means of maximising a reward signal. Learning Classifier Systems (LCSs) are a paradigm of cognitive systems that originated via representing agents in this framework, although due to flexibility in implementation have also been widely adapted to other kinds of machine learning (ML) tasks such as classification and clustering [16]. The most widely-studied LCS architecture to date, Wilson’s XCS [18], is at its heart a reinforcement



learner. More recently, an extension of XCS to allow for function approximation, dubbed XCSF [19], has been successfully used in the RL setting for value function approximation [12, 13].

LCSs utilise a combination of evolutionary computation and ML techniques to create population-based solutions to prediction problems. The most common style of LCSs are Michigan-style LCSs, where each individual (classifier) in the population represents a partial solution, and classifiers co-operate in a potentially overlapping piecewise ensemble to define an overall solution [16]. A general issue with Michigan-style LCSs is that of population bloat and/or redundancy. Since these systems learn in an online fashion and regularly refine their population via a genetic algorithm (GA), after learning is complete there are often members of the population that have not had time to properly adapt to the environment and form accurate predictions.

A common way to deal with this issue is to perform a post-processing *compaction* procedure after the system is trained in order to remove low-quality classifiers from the population [16]. Compaction seeks to shrink the population size as much as possible while simultaneously minimising degradation of predictive performance. This is often done as part of an analysis pipeline where the system is being used to “mine” knowledge from the problem via interpretation of the compacted population [17]. Wilson originally described a compaction algorithm for a variant of XCS trained on a classification problem in [20], and other algorithms such as those detailed in [7, 8, 15] extended this line of work. These algorithms all incorporate some kind of greedy heuristic to preferentially retain some classifiers over others, and mainly use metrics related to classification performance. Since compaction is related to knowledge discovery, other works have focused more on this latter task [4, 9, 17]. What all these works have in common is that they study compaction in the context of supervised learning.

In this work we apply XCSF to discrete maze-like RL environments and perform compaction on the trained populations. We are interested in measuring the performance of XCSF with respect to the optimal solutions to the environments, and investigating how performance is impacted when performing compaction. As part of our analysis we introduce a novel compaction algorithm called Greedy Niche Mass Compaction (GNMC) as a generalisation of previous work. We also attempt to connect our optimality metrics to the steps-to-goal metric used by other work applying LCSs to maze-like environments.

## 2 Background

### 2.1 Reinforcement Learning

RL environments can be modelled as a Markov Decision Process (MDP), defined by components  $(S, A, T, R, \gamma)$  where  $S$  is the state space,  $A$  is the action space,  $T$  is the transition function,  $R$  is the reward function and  $\gamma \in [0, 1]$  is the reward discount factor [14]. We consider the case where the agent interacting with the environment seeks to learn a *deterministic* behavioural policy  $\pi : S \rightarrow A$ . From the agent’s perspective,  $T$  and  $R$  are unknown and so learning becomes an act

of balancing exploration with exploitation to sample from  $T$  and  $R$  in order to construct  $\pi$ . If the full definition of the MDP is known, dynamic programming methods such as value iteration can be used to exhaustively obtain an optimal solution to the problem.

Value iteration yields an optimal Q-function  $Q^* : S \times A \rightarrow \mathbb{R}$ , which maps each state-action pair  $(s, a) \in S \times A$  to a real number representing the utility of the pair: the expected amount of cumulative discounted reward that can be obtained from performing action  $a$  in state  $s$ , and acting optimally thereafter. An optimal policy  $\pi^*$  can then be constructed by acting greedily with respect to  $Q^*$ . One of the main approaches to RL is to have the agent build an approximation  $\hat{Q}$  to  $Q^*$  from its environmental experience using e.g. temporal difference learning techniques such as Q-learning [14]. The agent's approximation  $\hat{\pi}$  to  $\pi^*$  can then be constructed by acting greedily with respect to  $\hat{Q}$ .

## 2.2 XCSF

XCSF is an LCS architecture designed to perform function approximation. It differs from XCS in that classifiers *compute* their predictions as a function of their inputs, instead of predicting a scalar value. The system operates by adaptively partitioning the input space into subspaces with classifiers (evolutionary component), in tandem with forming approximations to the target function in the subspaces (ML component) [3].

Classifiers take the form of *IF condition THEN action* rules. Partitioning is accomplished by specifying the *rule representation* to be used by conditions. Common choices include hyperrectangles [12, 19] and hyperellipsoids [5]. In the simplest case, linear functions can be used as a prediction scheme but extensions to the non-linear case have been investigated [11]. Additionally, classifiers have a number of parameters, denoted as *cl.param*, that store or calculate information related to them; the following parameters being important in this work: *fitness*—the predictive accuracy of a classifier relative to other classifiers in the action set(s) (defined below) that it participates in, *numerosity*—the number of copies of a classifier present in the population (necessary because the GA may produce classifiers with duplicate rules), *generality*—a quantity in the range (0, 1] representing the fraction of the input space covered by the classifier's condition. Numerosity yields the concept of *macroclassifiers* and *microclassifiers*, defined as the classifiers in the population with unique rule structures (possibly having numerosity > 1) and individual copies of these unique classifiers, respectively.

Applied as a RL method, XCSF uses a Q-learning style reinforcement component to form classifier predictions. The overall system output  $\hat{Q}$  is computed for each  $(s, a) \in S \times A$  according to:

$$\hat{Q}(s, a) = \frac{\sum_{cl \in [A]} cl.prediction(s) \cdot cl.fitness}{\sum_{cl \in [A]} cl.fitness} \quad (1)$$

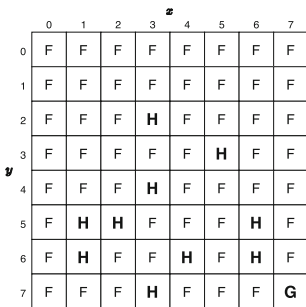
$[A]$  is termed an *action set* and contains classifiers in the population whose conditions match  $s$  and who advocate action  $a$ , i.e. XCSF's current knowledge about a particular *niche* of the environment.

### 3 Environments

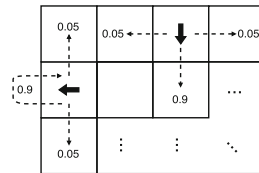
We consider two variations of the FrozenLake environment with grid size 8 (FrozenLake8x8) from OpenAI Gym<sup>1</sup>. FrozenLake is an episodic, fully observable grid navigation environment. In this environment, the agent must navigate across frozen cells to reach a goal, without falling into any holes. If the agent falls into a hole the episode terminates. The state representation used is an  $(x, y)$  co-ordinate representing the location of the agent in the grid, as shown in Fig. 1a. We use  $S$  to indicate the set of *non-terminal* states (frozen cells), and  $S^T$  to represent the set of *terminal* states (holes and the goal). The action space  $A = \{Left, Down, Right, Up\}$ , constant over all  $s \in S$ .

A parameter  $p_{slip}$  controls the level of stochasticity in the environmental transition dynamics. Figure 1b gives examples of transition dynamics with  $p_{slip} = 0.1$ . Transition stochasticity is global over all  $s \in S$ . By default  $p_{slip} = \frac{2}{3}$ , which is quite high. For our variants, we consider the cases where  $p_{slip} = 0$  and  $p_{slip} = 0.1$ , the latter because we wish to preserve the spirit of the default case while making the problem substantially easier through lowering the amount of noise incurred in transitions and therefore the reward signal. The reward function operates as follows: +1 if the agent transitions into G, 0 otherwise. We set  $\gamma = 0.95$  to ensure that there is time pressure to reach the goal. Note that a time step is counted even if the agent does not move to a new state after performing an action (as occurs with 90% probability in the leftmost example of Fig. 1b).

Figure 2 shows the optimal policies for our two variants. In the deterministic case reaching the goal is a shortest path problem, hence in some states there are multiple optimal actions. In the stochastic case the optimal policy is more strict as there is only a single optimal action in every state.



(a) Environment grid structure. Cell labels: F = frozen, H = hole, G = goal.



(b) Transition dynamics with  $p_{slip} = 0.1$ . With probability 0.1 the agent slips to one of the directions perpendicular to its intended direction (mass shared equally across both possible slip directions). As illustrated by the leftmost example, the agent is unable to maneuver off the grid.

**Fig. 1.** FrozenLake8x8 (a) structure and (b) example transition dynamics.

<sup>1</sup> <https://gym.openai.com/envs/FrozenLake8x8-v0/>.

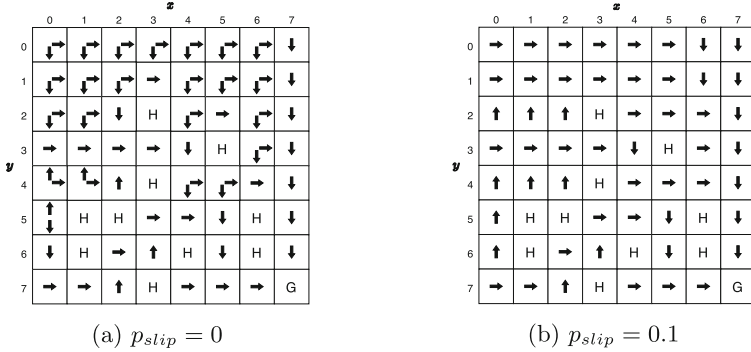


Fig. 2. Optimal policies for FrozenLake8x8,  $\gamma = 0.95$ .

### 4 XCSF Configuration

We use our own implementation of XCSF written in Python<sup>2</sup>, faithful to the base description of XCS given in [2]. We use the same linear prediction scheme as in [19], where each classifier has an associated weight vector and its prediction is computed as a dot product between its weight vector and the input vector, and classifier weight vectors are updated via a normalised least mean squares procedure with the prediction target calculated via the system’s reinforcement component. Also incorporated is the extension to XCS from [10], termed XCS $\mu$ , which is used to estimate uncertainty introduced by stochasticity in the environment. This involves adding a parameter  $\mu$  to each classifier which tracks minimum prediction error in the action sets the classifier participates in, adjusted by a separate learning rate  $\beta_\epsilon$ .

The rule representation used is an interval-based representation, specifically an integer-valued variant of min-percentage representation [6]. Interval minimum alleles are retained but percentage-to-maximum alleles are replaced by “span-to-maximum” alleles; interval maximums calculated as:  $\max = \min + \text{span}$ . The covering and mutation operators from [6] are adopted and modified to work with integer values, resembling those in [21]. Subsumption and calculation of condition generality are the same as in [21]. GA selection is done via tournament selection and uniform crossover is applied on allele sequences.

The chosen rule representation and prediction scheme yield a system that learns linear predictions of value over rectangular regions of the input space. This is suitable for both FrozenLake8x8 environments because it exploits the fact that Q-values decay smoothly (due to discounting) when moving away from the goal. In areas of the state space where there are no holes, accurate generalisation over large areas is possible (refer to e.g. top two rows in Fig. 1a) so only a few classifiers are required to cover such an area. The opposite is true for areas near holes. Compared to other Q-function approximators used in RL (e.g. neural networks),

<sup>2</sup> <https://github.com/jtbish/piecewise>, see also <https://github.com/jtbish/ppsn2020> for experimental code that uses this.

XCSF has the advantage of presenting its knowledge in a piecewise, easily interpretable format that can be reduced to a compact set of classifiers (as is the theme of this work).

## 5 Training Experiments

### 5.1 Setup

We train our implementation of XCSF described in Sect. 4 on the two environments detailed in Sect. 3. For the first environment ( $p_{slip} = 0$ ), the training budget is 400,000 time steps (environmental transitions) and for the second environment ( $p_{slip} = 0.1$ ) the budget is doubled to 800,000 time steps. Hyperparameters for both cases are:  $N=5000$ ,  $\beta=0.1$ ,  $\beta_\epsilon = 0.05$ ,  $\alpha = 0.1$ ,  $\epsilon_0 = 0.01$ ,  $\nu = 5$ ,  $\gamma = 0.95$ ,  $\theta_{GA} = 50$ ,  $\tau = 0.5$ ,  $\chi = 1.0$ ,  $v = 0.5$ ,  $\mu = 0.05$ ,  $\theta_{del} = 50$ ,  $\delta = 0.1$ ,  $\theta_{sub} = 50$ ,  $\epsilon_I = 10^{-3}$ ,  $f_I = 10^{-3}$ ,  $\theta_{mna} = 4$ ,  $doGASubsumption = True$ ,  $doActionSetSubsumption = False$ ,  $r_0 = 4$ ,  $m_0 = 4$ ,  $x_0 = 10$ ,  $\eta = 0.1$ . Hyperparameter meanings correspond to those given in [2, 3, 10, 19, 21], except for  $v$  which is our addition and controls the probability of an allele being crossed over during uniform crossover. The two most critical hyperparameters are  $N$  (maximum population size in number of microclassifiers) and  $\epsilon_0$  (target absolute approximation error). We tuned their values manually, along with the training budget. For other hyperparameters, we followed guidance from [2, 12, 16].

By default, the agent starts each episode in state  $(0, 0)$ , which puts a heavy emphasis on exploration to reach the goal, making learning relatively difficult. To make learning easier, we allow the agent to start a training episode in any  $s \in S$ , selected uniformly at random. We can therefore safely adopt the alternating explore-exploit action selection strategy used elsewhere in the literature, i.e.  $\epsilon$ -greedy with a fixed value of  $\epsilon = 0.5$ .

### 5.2 Metrics

Before training, we use value iteration to compute  $Q^*$  and consequently  $\pi^*$  for each environment. XCSF's  $\hat{Q}$  mean absolute error (MAE) can then be calculated as:

$$\frac{1}{|S||A|} \sum_{s \in S} \sum_{a \in A} |Q^*(s, a) - \hat{Q}(s, a)| \quad (2)$$

MAE is used because we wish to directly compare with  $\epsilon_0$ . To allow for comparison between  $\pi^*$  and  $\hat{\pi}$ , policies are encoded as a series of *binary action advocacy vectors*, one for each  $s \in S$ , whereby if policy  $\pi$  advocates action  $a_i$  in state  $s$ , bit  $i$  of  $\pi(s)$  is set to 1, 0 otherwise. For example, following Fig. 2a the optimal actions in state  $(0, 0)$  are  $\{Down, Right\}$ . Assuming the ordering of actions is  $\{Left, Down, Right, Up\}$ , then the encoding  $\pi^*((0, 0)) = [0, 1, 1, 0]$ . XCSF's  $\hat{\pi}$  accuracy can then be calculated as:

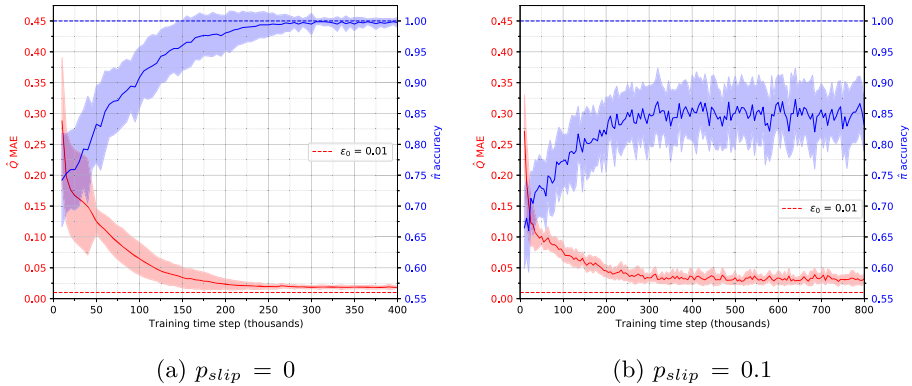
$$\frac{1}{|S|} \sum_{s \in S} C(\pi^*(s), \hat{\pi}(s)) \quad (3)$$

where  $C$  is a Boolean function that accepts two binary action advocacy vectors,  $a^*$  and  $\hat{a}$ , and determines if *at least one* of the actions advocated in  $a^*$  is also advocated in  $\hat{a}$ , i.e. determines if  $\hat{a}$  is “correct”:

$$C(a^*, \hat{a}) = \begin{cases} 1 & \text{count\_ones}(a^* \text{ AND } \hat{a}) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

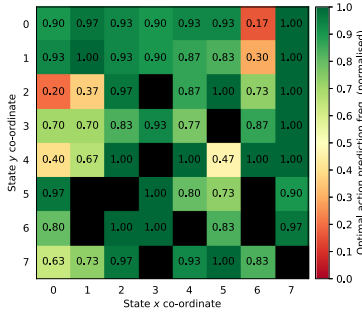
Reusing the previous example, if  $s = (0, 0)$ ,  $\pi^*(s) = [0, 1, 1, 0]$  and also  $\hat{\pi}(s) = [0, 0, 1, 0]$  then  $C$  returns 1 because  $\hat{\pi}$  advocates one of the optimal actions, *Right*.

### 5.3 Results



**Fig. 3.** XCSF training performance curves on FrozenLake8x8 environments. Solid lines are the mean of 30 trials, shaded regions are one standard deviation.

Figure 3 shows XCSF training performance curves for both environments, measured over time are  $\hat{Q}$  MAE and  $\hat{\pi}$  accuracy. In the deterministic case, XCSF converges to a small MAE that is slightly larger than the target error threshold  $\epsilon_0$ , with  $\hat{\pi}$  accuracy very close to the maximum value of 1. In the stochastic case, MAE is still quite small but noticeably larger than in the deterministic case, also with larger variance.  $\hat{\pi}$  accuracy is significantly lower and with much larger variance. We now investigate this reduction of  $\hat{\pi}$  accuracy in the stochastic case in more detail. Figure 4 shows the frequency of optimal action predictions for each  $s \in S$  over the 30 trained instances. From this we can see that XCSF is quite often predicting the optimal action in a majority of states. However, there are a few states that are degrading policy accuracy more than others. Table 1 shows the distributions of actions predicted in the four states with lowest optimal action prediction frequencies, and indicates that for these states if the predicted action is not optimal (*Up* or *Down*) it is at least sensible (*Right*). Thus the situation is not as poor as it first seems.



**Fig. 4.** XCSF optimal action prediction frequency for Frozen-Lake8x8  $p_{slip} = 0.1$ , calculated over 30 instances.

**Table 1.** Distribution of action predictions for the four lowest frequency states in Fig. 4. Optimal actions for each state are set in bold.

		Action				Optimal Freq.
		L	D	R	U	
State	(0, 2)	0	0	24	<b>6</b>	6/30=0.2
	(1, 2)	0	0	19	<b>11</b>	11/30=0.37
	(6, 0)	0	<b>5</b>	25	0	5/30=0.17
	(6, 1)	0	<b>9</b>	21	0	9/30=0.3

## 6 Compaction

We now turn to the main consideration of this work: compaction of trained XCSF populations. Algorithm 1 presents Greedy Niche Mass Compaction (GNMC), a novel compaction algorithm designed for use on LCS populations applied to RL environments with discrete state-action spaces. GNMC considers all environmental action sets (niches) and greedily keeps some number of the best quality classifiers in each. The notion of “best quality” is defined by the parameter  $\lambda$ , which is a function that assigns each classifier a mass (quality weighting) in the action set.  $\rho$  acts as a compression factor and controls the number of classifiers kept in each action set; higher values result in more classifiers being discarded.

---

### Algorithm 1: Greedy Niche Mass Compaction (GNMC)

---

```

Input: Classifier mass function  $\lambda$ , mass removal factor  $\rho \in [0, 1]$ ;
1  $toKeep = \emptyset$ ;
2 for  $(s, a) \in S \times A$  do
3   Create action set  $[A]$  for  $(s, a)$ ;
4   Create set  $[A]'$  by sorting  $[A]$  in descending order according to  $\lambda$ ;
5    $totalMass = \sum_{cl \in [A]'} \lambda(cl)$ ;
6    $targetMass = (1 - \rho) \cdot totalMass$ ;
7    $currentMass = 0$ ;
8   while  $currentMass < targetMass$  do
9      $cl = \text{next classifier in } [A]'$ ;
10     $toKeep = toKeep \cup \{cl\}$ ;
11     $currentMass += \lambda(cl)$ ;
12  end
13 end
14 Remove classifiers not in  $toKeep$  from the population  $[P]$ ;

```

---

GNMC exhibits a number of desirable properties:

1. The exact number of classifiers discarded in each action set is dependent on the distribution of classifier mass;  $\rho$  is sensitive to this distribution.
2.  $\rho$  can be adjusted in a smooth manner without needing prior information about the size of action sets.
3. It is guaranteed that no “gaps” in the predictive mapping are introduced, due to all action sets being considered and at least a single classifier being kept in each action set (because  $\rho$  cannot equal 1).
4. Any classifiers that only match a state  $s \in S^T$  (and so have zero experience and do not contribute to overall predictions) are implicitly removed from the population because they are never added into the *toKeep* set; the for loop on line 2 operates only over  $S$ . This occurs even when  $\rho = 0$ .

Notably, simple compaction strategies such as removing all classifiers with experience less than some threshold do not uphold point 3 listed above. This property is crucial for function approximation in RL where a complete mapping of the state-action space is necessary.

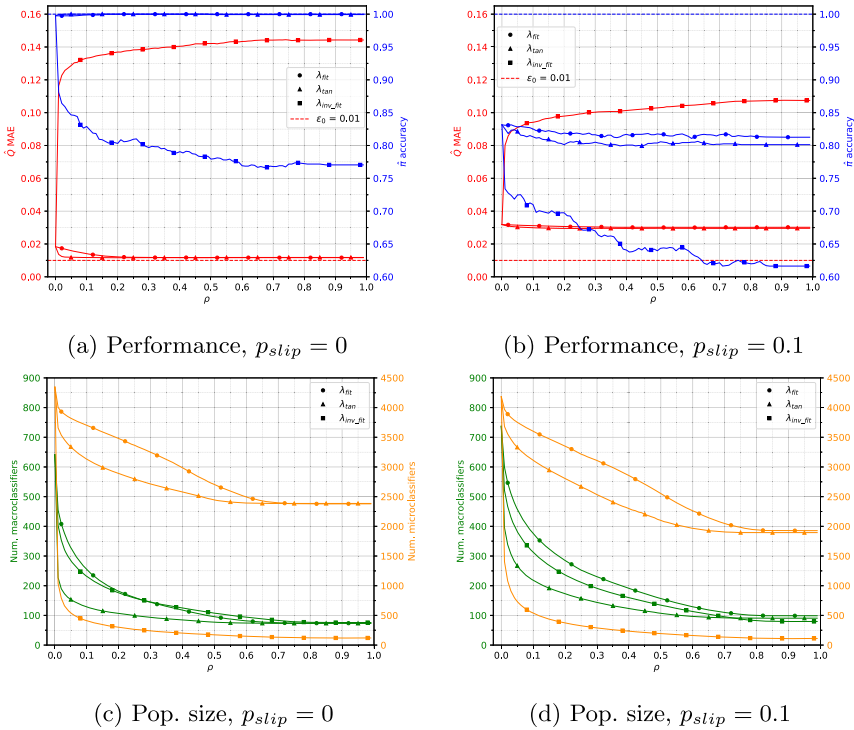
GNMC can be viewed as a generalisation of previous work in the literature. In particular, we consider the work of Tan et al. in [15], where the authors define a compaction algorithm in the context of a classification task, called Parameter Driven Rule Compaction (PDRC). PDRC operates by forming a correct set  $[C]$  for each environmental input (each training set data point) then keeping the classifier in  $[C]$  with the largest product of accuracy, numerosity, and generality. All other classifiers in  $[C]$  are discarded. Translating between classification and RL,  $[C]$  is analogous to  $[A]$  and classifier accuracy is analogous to fitness because Tan et al. employ a UCS (sUPervised Classifier System [1]) variant where accuracy is *equivalent to* fitness. GNMC is therefore equivalent to PDRC when the mass function  $\lambda(cl) = cl.fitness \times cl.numerosity \times cl.generality$  and the mass removal factor  $\rho$  is sufficiently high so as to keep only a single classifier from each action set.

We apply GNMC to our trained XCSF populations, considering three different mass functions, named with subscripts. The first is  $\lambda_{fit} = cl.fitness$ , motivated by the manner in which XCSF calculates its overall predictions: see Eq. 1. Classifiers with higher fitness have more weight in the overall prediction, so using fitness as a mass function makes sense. The second mass function is  $\lambda_{tan} = cl.fitness \times cl.numerosity \times cl.generality$ . The final mass function is an antagonistic variant of the first mass function that is designed to see what happens when GNMC is operating with “bad information”:  $\lambda_{inv\_fit} = \frac{1}{cl.fitness}$ . Figure 5 shows results of applying GNMC with these three mass functions to the XCSF instances trained on both environments, measured are the effect on performance ( $\hat{Q}$  MAE and  $\hat{\pi}$  accuracy) and population size (number of macro and microclassifiers).

In the deterministic case, GNMC retains performance when either  $\lambda_{fit}$  or  $\lambda_{tan}$  is used, for any value of  $\rho$ .  $\hat{Q}$  MAE improves very slightly as  $\rho$  increases, and  $\hat{\pi}$  accuracy is unchanged. Using  $\lambda_{inv\_fit}$  gives smooth degradation in both



metrics. Looking at the population sizes, both  $\lambda_{fit}$  and  $\lambda_{tan}$  exhibit a moderate reduction in the number of microclassifiers and a significant reduction in the number of macroclassifiers as  $\rho$  increases. However  $\lambda_{inv\_fit}$  is different, yielding similar number of macroclassifiers in the extreme, but drastically smaller number of microclassifiers. This conforms to our expectations of its operation; classifiers that are kept by  $\lambda_{inv\_fit}$  have low fitness values, and since low fitness classifiers tend to have low numerosity, the ratio of microclassifiers to macroclassifiers is low. For  $\lambda_{fit}$  and  $\lambda_{tan}$  the ratio is much higher. In the stochastic case, the situation is similar overall with a slight difference when considering the effect on performance: namely that  $\hat{\pi}$  accuracy for both  $\lambda_{fit}$  and  $\lambda_{tan}$  is degraded slightly instead of remaining constant as  $\rho$  increases.



**Fig. 5.** Results of applying GNMC with three different mass functions to both Frozen-Lake8x8 environments. All curves are the mean over 30 instances. Values of  $\rho$  used are from 0 to 0.99 in increments of 0.01.

## 7 Rollout Analysis

We now investigate the relationship between policy accuracy and the commonly used black-box performance metric, steps-to-goal (STG), from other

works applying LCSs to maze-like RL environments, e.g. [10,12]. Note that in such environments, minimising STG is equivalent to maximising cumulative discounted reward. For our analysis we consider only the stochastic variant of FrozenLake8x8, as it showed the most interesting variations in  $\hat{\pi}$  accuracy when GNMC was applied to it in Sect. 6.

**Table 2.** Results of STG testing procedure on FrozenLake8x8  $p_{slip} = 0.1$  for three different groups of XCSF instances. Asterisks for mean and max STG indicate incomplete data. Set in bold are the “worst” values for each column ( $\hat{\pi}$  acc. minimum, others maximum).

Instance num.	No compaction (Group A)				GNMC $\lambda_{fit} \rho = 0.99$ (Group B)				GNMC $\lambda_{inv\_fit} \rho = 0.99$ (Group C)			
	Mean STG	Max STG	Num. roll	$\hat{\pi}$ acc.	Mean STG	Max STG	Num. roll	$\hat{\pi}$ acc.	Mean STG	Max STG	Num. roll	$\hat{\pi}$ acc.
1	15.42	20	114	0.89	15.31	20	117	0.91	15.44	21	114	0.79
2	15.39	20	116	0.87	15.43	20	117	0.79	15.45	20	115	0.68
3	32.36	110	123	0.75	*	*	<b>150</b>	0.72	*	*	<b>150</b>	0.55
4	15.37	20	127	0.83	15.46	21	127	0.83	18.74	89	131	0.55
5	15.38	20	116	0.83	15.38	20	116	0.85	15.38	20	116	0.70
6	15.38	20	116	0.85	16.13	25	126	0.79	33.11	148	115	0.66
7	15.38	20	116	0.87	15.38	20	116	0.85	15.38	20	116	0.74
8	15.38	20	116	0.87	15.38	20	116	0.85	37.46	98	142	0.60
9	15.43	20	115	0.83	33.25	93	118	0.81	15.39	20	116	0.75
10	15.31	20	117	0.85	15.31	20	117	0.91	15.70	23	117	0.68
11	15.38	20	116	0.87	15.45	20	115	0.89	*	*	<b>150</b>	0.55
12	15.66	20	113	0.77	15.39	20	116	0.77	*	*	<b>150</b>	0.64
13	37.61	119	130	0.75	37.43	119	128	0.75	*	*	<b>150</b>	0.57
14	15.38	20	116	0.83	15.33	20	117	0.79	47.94	143	119	0.64
15	15.45	20	115	0.85	15.43	20	117	0.83	15.44	21	114	0.68
16	15.38	20	116	0.91	15.42	20	114	0.92	37.22	112	120	0.55
17	15.43	21	122	0.83	15.32	21	116	0.83	15.48	21	129	0.64
18	15.38	20	116	0.89	15.37	20	119	0.89	*	*	<b>150</b>	0.72
19	15.38	20	116	0.85	15.38	20	116	0.81	15.27	19	121	0.72
20	15.92	21	118	0.79	16.06	25	128	0.75	*	*	<b>150</b>	0.58
21	<b>67.29</b>	<b>193</b>	110	0.79	<b>68.61</b>	<b>193</b>	117	0.75	53.88	110	121	0.57
22	15.38	20	116	0.85	32.79	110	123	0.79	62.20	<b>200</b>	124	0.70
23	15.38	20	116	0.85	15.33	20	117	0.79	53.57	153	115	0.53
24	32.91	100	112	0.79	15.33	20	117	0.75	*	*	<b>150</b>	0.60
25	15.38	20	116	0.89	15.38	20	116	0.87	28.51	86	119	0.62
26	17.21	82	119	0.81	15.85	25	138	0.87	*	*	<b>150</b>	<b>0.38</b>
27	15.74	21	<b>142</b>	<b>0.74</b>	15.96	40	133	0.72	*	*	<b>150</b>	0.47
28	36.96	139	126	0.79	*	*	<b>150</b>	<b>0.70</b>	*	*	<b>150</b>	0.47
29	15.38	20	116	0.83	15.37	19	119	0.77	<b>82.36</b>	197	114	0.62
30	15.38	20	116	0.83	15.38	20	116	0.81	*	*	<b>150</b>	0.55

A testing procedure to measure STG is devised as follows: allow each XCSF instance a budget of 150 rollouts (episodes) and in these 150 rollouts, attempt to record STG (successfully reach the goal) 100 times. If 100 successes are not achieved, STG data is incomplete. In each rollout, the agent’s initial state is  $(0, 0)$  and the random seed of the environment is set to a unique number. Note this is different from training where the agent’s initial state was any  $s \in S$ , selected uniformly at random. This testing procedure is applied to all 30 trained XCSF instances in three groups, representing different levels of GNMC compaction: no compaction, compaction with  $\lambda_{fit} \rho = 0.99$ , and compaction with

$\lambda_{inv\_fit} \rho = 0.99$ . Table 2 shows the collected results: included are mean and max STG, number of rollouts performed, and  $\hat{\pi}$  accuracy (for reference). Note that minimum STG in the environment is 14.

Group A in general achieves admirable mean STG; only 5 out of 30 instances could be considered as outliers. Number of rollouts for all instances is generally not much higher than 110. This indicates that in most instances XCSF is quickly navigating towards the goal, with a low failure rate due to environmental stochasticity. Comparing Group B to Group A, there are two instances in Group B that have failed to collect complete STG data, also having the two lowest policy accuracies. In general, all four measures degrade only slightly between the two groups, which indicates that the compaction applied to Group B is not having a detrimental effect on performance. Transitioning from Group A to Group C however produces noticeable performance loss. 11 out of 30 instances fail to record complete STG data, and generally those that do show degradation in all four measures.

It is difficult to determine the exact relationship between STG and policy accuracy. Some instances (e.g. instance 1) exhibit minimal degradation in both measures between groups, but some exhibit larger degradation (e.g. instance 22). In cases where degradation in policy accuracy is small but degradation in STG is large, the cause is often a minority of states along the edge of the grid that are advocating actions where the only possible way to advance further towards the goal is to slip, e.g. advocating *Right* in any of the states in the rightmost column (where  $x = 7$ ). It is therefore clear that the two metrics are complementary rather than competitive. Policy accuracy is measured globally and is always defined, and STG is measured on a *specific task* (starting state), possibly being undefined/incomplete. The starting state is an arbitrary choice and if altered results in differing STG but unchanged policy accuracy.

## 8 Conclusion

We trained XCSF on a deterministic and stochastic variant of FrozenLake8x8, measuring its performance with respect to the optimal solutions produced via dynamic programming. Results show that in both cases XCSF achieved low Q-function approximation error, and in the deterministic case XCSF converged to maximum policy accuracy. In the stochastic case policy accuracy was noticeably degraded both because of increased problem difficulty and increased strictness of the optimal policy. Next we introduced Greedy Niche Mass Compaction (GNMC), a compaction algorithm designed for LCSs applied to discrete RL environments. We showed GNMC is a generalisation of previous work and applied it to our trained XCSF instances. Given a suitable mass function, GNMC can yield a significant reduction in population size without increasing function approximation error and only slightly decreasing policy accuracy. Finally we linked our policy accuracy metric to the steps-to-goal metric used in previous work across multiple groups of compacted XCSF instances. This highlighted how the two metrics are complementary rather than competitive. Suggested future work

includes applying GNMC to environments where populations are larger/more complex and the mass removal factor has more impact on performance. GNMC's concept could also be extended to continuous state and/or action spaces.

## References

1. Bernadó-Mansilla, E., Garrell-Guiu, J.M.: Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evol. Comput.* **11**(3), 209–238 (2003)
2. Butz, M.V., Wilson, S.W.: An algorithmic description of XCS. *Soft Comput. - Fusio Found. Methodol. Appl.* **6**(3–4), 144–153 (2002). <https://doi.org/10.1007/s005000100111>
3. Butz, M.V.: Learning classifier systems. In: Kacprzyk, J., Pedrycz, W. (eds.) *Springer Handbook of Computational Intelligence*, pp. 961–981. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-43505-2\\_47](https://doi.org/10.1007/978-3-662-43505-2_47)
4. Butz, M.V., Lanzi, P.L., Llorà, X., Goldberg, D.E.: Knowledge extraction and problem structure identification in XCS. In: Yao, X., et al. (eds.) *PPSN 2004. LNCS*, vol. 3242, pp. 1051–1060. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30217-9\\_106](https://doi.org/10.1007/978-3-540-30217-9_106)
5. Butz, M.V., Lanzi, P.L., Wilson, S.W.: Function approximation with XCS: hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Trans. Evol. Comput.* **12**(3), 355–376 (2008)
6. Dam, H.H., Abbass, H.A., Lokan, C.: Be real! XCS with continuous-valued inputs. In: *Proceedings of the 2005 Workshops on Genetic and Evolutionary Computation - GECCO 2005*, p. 85. ACM Press, Washington, D.C. (2005)
7. Dixon, P.W., Corne, D.W., Oates, M.J.: A ruleset reduction algorithm for the XCS learning classifier system. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) *IWLCS 2002. LNCS (LNAI)*, vol. 2661, pp. 20–29. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-40029-5\\_2](https://doi.org/10.1007/978-3-540-40029-5_2)
8. Fu, C., Davis, L.: A modified classifier system compaction algorithm. In: *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 920–925. Morgan Kaufmann (2002)
9. Kharbat, F., Odeh, M., Bull, L.: New approach for extracting knowledge from the XCS learning classifier system. *Int. J. Hybrid Intell. Syst.* **4**, 49–62 (2007)
10. Lanzi, P.L., Colombetti, M.: An extension to the XCS classifier system for stochastic environments. In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation, GECCO 1999*, vol. 1, pp. 353–360. Morgan Kaufmann Publishers Inc., Orlando, July 1999
11. Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.E.: Extending XCSF beyond linear approximation. In: *GECCO 2005: Genetic and Evolutionary Computation Conference: Volume*, pp. 1827–1834. ACM Press (2005)
12. Lanzi, P.L., Loiacono, D., Wilson, S.W., Goldberg, D.E.: XCS with computed prediction in multistep environments. In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO 2005*, pp. 1859–1866. Association for Computing Machinery, Washington DC, June 2005
13. Lanzi, P., Loiacono, D., Wilson, S., Goldberg, D.: XCS with computed prediction in continuous multistep environments. In: *2005 IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2032–2039, September 2005

14. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. Adaptive Computation and Machine Learning Series, 2nd edn. The MIT Press, Cambridge (2018)
15. Tan, J., Moore, J., Urbanowicz, R.: Rapid rule compaction strategies for global knowledge discovery in a supervised learning classifier system. In: Advances in Artificial Life, ECAL 2013, pp. 110–117. MIT Press, September 2013
16. Urbanowicz, R.J., Browne, W.N.: Introduction to Learning Classifier Systems. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-3-662-55007-6>
17. Urbanowicz, R.J., Granizo-Mackenzie, A., Moore, J.H.: An analysis pipeline with statistical and visualization-guided knowledge discovery for Michigan-style learning classifier systems. *IEEE Comput. Intell. Mag.* **7**(4), 35–45 (2012)
18. Wilson, S.W.: Classifier fitness based on accuracy. *Evol. Comput.* **3**(2), 149–175 (1995)
19. Wilson, S.W.: Classifiers that approximate functions. *Nat. Comput.* **1**, 1–2 (2001)
20. Wilson, S.W.: Compact rulesets from XCSI. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2001. LNCS (LNAI), vol. 2321, pp. 197–208. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-48104-4\\_12](https://doi.org/10.1007/3-540-48104-4_12)
21. Wilson, S.W.: Mining oblique data with XCS. In: Luca Lanzi, P., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2000. LNCS (LNAI), vol. 1996, pp. 158–174. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44640-0\\_11](https://doi.org/10.1007/3-540-44640-0_11)



# Hybridizing the 1/5-th Success Rule with Q-Learning for Controlling the Mutation Rate of an Evolutionary Algorithm

Arina Buzdalova<sup>1</sup>, Carola Doerr<sup>2(✉)</sup>, and Anna Rodionova<sup>1</sup>

<sup>1</sup> ITMO University, 49 Kronverkskiy Avenue, 197101 Saint Petersburg, Russia  
abuzdalova@gmail.com

<sup>2</sup> Sorbonne Université, CNRS, LIP6, Paris, France  
Carola.Doerr@lip6.fr

**Abstract.** It is well known that evolutionary algorithms (EAs) achieve peak performance only when their parameters are suitably tuned to the given problem. Even more, it is known that the best parameter values can change during the optimization process. Parameter control mechanisms are techniques developed to identify and to track these values.

Recently, a series of rigorous theoretical works confirmed the superiority of several parameter control techniques over EAs with best possible static parameters. Among these results are examples for controlling the mutation rate of the  $(1 + \lambda)$  EA when optimizing the OneMax problem. However, it was shown in [Rodionova et al., GECCO'19] that the quality of these techniques strongly depends on the offspring population size  $\lambda$ .

We introduce in this work a new hybrid parameter control technique, which combines the well-known one-fifth success rule with Q-learning. We demonstrate that our HQL mechanism achieves equal or superior performance to all techniques tested in [Rodionova et al., GECCO'19] and this – in contrast to previous parameter control methods – simultaneously for all offspring population sizes  $\lambda$ . We also show that the promising performance of HQL is not restricted to OneMax, but extends to several other benchmark problems.

**Keywords:** Parameter control · Q-learning · Offspring population size

## 1 Introduction

The problem of selecting suitable parameter configurations for an evolutionary algorithm is frequently considered to be one of the most essential drawbacks of evolutionary computation methods, and possibly a major obstacle towards wider application of these optimization techniques in practice [31].

Automated configuration techniques such as SPOT [3], irace [32], SMAC [24], hyperband [30], MIP-EGO [42], BOHB [22], and many others have been developed to assist the user in the decisive task of selecting suitable parameter configurations. These *parameter tuning* methods, however, require to test different

parameter combinations before presenting a recommendation. They are therefore rather time-consuming, and are not applicable when the possibility for such training is not given, e.g., when the problem is truly black-box, with no/only little information about its fitness landscape structure.

An orthogonal approach to solve the algorithm configuration problem is *parameter control*, which does not require a priori training, and aims at identifying suitable parameter combinations *on the fly*, i.e., while executing the optimization [21,27,31]. Apart from being more generally applicable than parameter tuning, parameter control also bears the advantage of being able to adjust the search behavior of the evolutionary algorithm to the different stages of the optimization process. Most state of the art evolutionary algorithms therefore make use of parameter control, in particular in the continuous domain, where a decreasing search radius is needed to eventually converge towards an optimal point. However, one should not forget that parameter control mechanisms, too, introduce their own hyperparameters, which need to be adequately set by the user prior to running the algorithm. Here again one can apply parameter tuning (e.g., via so-called per-instance algorithm configuration [4]), but the general hope is that the setting of the hyperparameters is less critical to achieve reasonable performance.

However, while parameter control is routinely used in numerical optimization, its potential remains far from being well exploited in the optimization of problems with discrete decision variables, where it has only recently re-gained momentum as a now very active area of research. In particular in the sub-domain of runtime analysis, parameter control has enjoyed rising attention in the last years, as summarized in [10].

A particularly well-researched topic in the theory literature for parameter control in discrete optimization heuristics is the  $(1 + \lambda)$  Evolutionary Algorithm (EA) with dynamic mutation rates and fixed offspring population size  $\lambda$  optimizing the ONEMAX problem (the problem of controlling  $\lambda$  has also been addressed, e.g., in [29], but has received much less attention so far). Not only was this problem one of the first ones for which dynamic mutation schemes were approximated [2], and not only is it frequently used as a test case for empirical works [7], but it is also one of the few problems for which we have a very solid theoretical understanding.

Extending the previous work from [18], we have presented at GECCO'19 a comparative empirical study of several mechanisms suggested in the theory literature [37]. Among other findings, we demonstrated that the efficiency of all benchmarked techniques depends to a large extent on the offspring population size  $\lambda$ . For example, we observed that the 2-rate  $(1 + \lambda)$  EA suggested in [15] is the best among the tested algorithms when  $\lambda$  is smaller than 50. For larger offspring population sizes, however, this algorithm is outperformed by a  $(1 + \lambda)$  EA which uses the one-fifth success rule to control the mutation rate. We also observed in [37] that the ranking of the algorithms was identical for all tested dimensions  $n \in [10^4 .. 10^5]$ .

*Our Results.* The results presented in [37] raise the question if one can achieve stable performance across all offspring population sizes  $\lambda$ . We address this problem by introducing a new parameter control scheme, which hybridizes the one-fifth success rule with Q-learning. More precisely, we first introduce the  $(1 + \lambda)$  QEA, which uses Q-learning only to control the mutation rate. The  $(1 + \lambda)$  QEA learns for each optimization state whether it should increase or decrease the current mutation rate (we use constant factor changes). We show that the  $(1 + \lambda)$  QEA performs efficiently on ONEMAX for all observed values of  $\lambda$  when an appropriate lower bound  $p_{\min}$  for the mutation rate is used. In absence of a well-tuned lower bound, however, the performance of the  $(1 + \lambda)$  QEA drops significantly. We show that this dependence on the value of  $p_{\min}$  can be mitigated by a hybridization of the  $(1 + \lambda)$  QEA with the one-fifth success rule. More precisely, the hybrid Q-learning EA (the  $(1 + \lambda)$  HQEA) extends the  $(1 + \lambda)$  QEA by using the one-fifth success rule in states that have not been visited before and for those for which the  $(1 + \lambda)$  QEA is ambiguous with respect to the two available actions.

We show that, on ONEMAX, the  $(1 + \lambda)$  HQEA outperforms or at least performs on par with all algorithms tested in [37], and this simultaneously for all tested values of  $\lambda \in [1..2^{12}]$  and also for both considered lower bounds for the mutation rate,  $p_{\min} = 1/n$  and  $p_{\min} = 1/n^2$ , respectively. It therefore solves the issue of the other control mechanisms previously suggested in the theory literature. Note here that we do not have a theoretical convergence analysis of the  $(1 + \lambda)$  HQEA. Given its complexity, it may be beyond the current state of the art in runtime analysis, as it requires to keep track of multiple states, which are highly dependent. We are nevertheless confident that the robust performance of the  $(1 + \lambda)$  HQEA encourages further work on learning-based parameter control, and their hybridization with other classical control methods.

In the last parts of this paper we also show that the promising performance of the  $(1 + \lambda)$  HQEA is not restricted to ONEMAX. More precisely, we show that it performs well also on the LEADINGONES function, as well as on several benchmark functions suggested in [17].

*Related Work.* We are not the first to use reinforcement learning (RL) as a parameter control technique. An exhaustive survey of RL-based parameter control approaches can be found in [27]. Particularly, there are parameter control approaches based on techniques for the *Multi-Armed Bandit Problem (MAB)*, see [23] (and references mentioned therein) and [13] for a theoretical investigation of MAB-based parameter control.

In many of the known approaches, RL algorithms are used to select the parameter values directly. For numerical parameters, however, most common techniques require to either discretize the value space [25] or to make use of quite sophisticated techniques [1, 20, 38], which are rather difficult to grasp without expert knowledge.

In contrast to such a direct selection of the parameter values, we use in this work an indirect approach which uses as actions the possibility to increase the current parameter value by some fixed multiplier, or decrease it. As we shall



see below, this yields a simple, yet efficient, control mechanism. Like most common parameter control techniques, including those studied in this work, this indirect approach has the advantage of a smoother transition of the mutation rates between consecutive iterations. This behavior is beneficial if the optimal parameter values do not change abruptly, which is the case in many problems analyzed in theoretical works [11, 14], but also the case in many applications of evolutionary algorithms to machine learning problems, including hyperparameter optimization itself [35]. Exceptions to this rule exist, of course, and the jump functions [19] are a classical example for a problem requiring such an abrupt change. In such cases it may take the parameter control mechanisms some time to adjust the mutation rate to the appropriate scale.

We note that a similar indirect control approach has been described in [34], where an indirect control of the step size of the  $(1+1)$  evolution strategy (ES) is described. In contrast to our work, however, this approach (which uses SARSA – another common reinforcement learning algorithm – instead of Q-learning) did not manage to outperform the  $(1+1)$  ES with suitably tuned static step sizes.

## 2 Previous $(1+\lambda)$ EAs with Dynamic Mutation Rates

We briefly review the algorithms studied in [37] and summarize their main findings. We assume in our presentation that the algorithms operate on a problem  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ , with the objective to maximize this function.

**The  $(1+\lambda)$  EA.** The standard  $(1+\lambda)$  EA is an elitist algorithm, which always keeps a current best solution  $x$  in its memory. The  $(1+\lambda)$  EA is initialized with a point chosen from the search space  $\{0, 1\}^n$  uniformly at random. In each iteration,  $\lambda$  offspring are sampled by applying standard bit mutation to the parent  $x$ , i.e., the algorithm creates  $\lambda$  offspring  $y^{(1)}, \dots, y^{(\lambda)}$  by creating  $\lambda$  copies of  $x$  and flipping each bit in these copies with some probability  $0 < p < 1$ . The variable  $p$  is commonly referred to as the *mutation rate*. We set it to  $p = 1/n$  in our experiments, which is a standard recommendation and often a fall-back value if no indication is given that larger values could be beneficial. The best of the  $\lambda$  offspring (ties broken uniformly at random) replaces the parent if it is at least as good. The  $(1+\lambda)$  EA continues until some user-defined termination criterion is met (see “implementation details” below for our setting).

**The  $(1+\lambda)$  EA( $A, b$ ).** The  $(1+\lambda)$  EA( $A, b$ ) extends the  $(1+\lambda)$  EA by an adaptive choice of the mutation rate  $p$ . Its  $(1+1)$  variant was suggested in [16], and we use a straightforward extension to the  $(1+\lambda)$  EA by updating the mutation rate  $p$  by  $Ap$  if the best of the  $\lambda$  offspring is at least as good as the parent and by decreasing the mutation rate to  $bp$  otherwise. It is ensured that the mutation rate does not fall below some minimal mutation rate  $p_{\min} > 0$  and that it does not exceed  $p_{\max} = 1/2$ , by capping the value of  $p$  appropriately where required. As argued in [12], this update rule is essentially a one-fifth success rule, even if this term was not mentioned in [16]. The one-fifth success rule was originally suggested in [9, 36, 39] and its interpretation for the discrete optimization is due

to [28]. More precisely, the idea is that the mutation rate should remain constant if a certain ratio of iterations is successful (i.e., produces a solution of better than previous-best quality). In our work, this success ratio is  $1/2$ , whereas the traditional rule suggests a success ratio of  $1/5$ .

The  $(1 + \lambda)$  EA( $A, b$ ) has three hyperparameters,  $A$ ,  $b$ , and  $p_{\min}$ . In our experiments, we set  $A = 2$ ,  $b = 1/2$ , and consider  $p_{\min} \in \{1/n, 1/n^2\}$ . We initialize  $p$  by  $1/n$ . Note that these values are not specifically tuned, but we chose them to be consistent with previous works, and in particular with [37]. The reader interested in the sensitivity of the performance of the  $(1 + \lambda)$  EA( $A, b$ ) with respect to these parameters is referred to [16] and [12] for an empirical and a theoretical investigation, respectively.

**The 2-rate  $(1 + \lambda)$  EA $_{r/2, 2r}$ .** The  $(1 + \lambda)$  EA $_{r/2, 2r}$  suggested in [15] uses two different mutation rates in each iteration: half the offspring are created with mutation rate  $p/2$  and the other  $\lambda/2$  offspring are sampled with mutation rate  $2p$ . The mutation rate is parametrized as  $p = r/n$  in the  $(1 + \lambda)$  EA $_{r/2, 2r}$ . The value of  $r$  is updated after each iteration by a random decision which gives preference to the rate by which the best offspring has been created. The latter is selected with probability  $3/4$ , whereas the other one of the two tested mutation rates is chosen with probability  $1/4$ . As in the  $(1 + \lambda)$  EA( $A, b$ ), the mutation rate is capped at  $p_{\min} \in \{1/n^2, 1/n\}$  and  $p_{\max} = 1/2$ , respectively.

*Implementation Details.* We briefly summarize a few common assumptions made in all our algorithms.

**Shift Mutation Strategy.** All algorithms described above use standard bit mutation as variation operator. To avoid sampling offspring that are identical to the parent (these offspring would not bring any new information to our optimization process, and are therefore useless), we use the “shift” operation suggested in [6]. If an offspring equals its parent, this strategy simply flips a randomly chosen bit. We write  $y \leftarrow \text{mutate}(x, p)$  if  $y$  is sampled by applying the shift mutation operator with mutation rate  $p$  to  $x$ .

**Termination Criterion and Runtime Measure.** We focus in this work on the *runtime* (also known as *optimization time*), which we measure in terms of generations that are needed until an optimal solution is evaluated for the first time. Since we only study algorithms with static offspring population size  $\lambda$ , the classical runtime in terms of function evaluations is easily obtained by multiplication with  $\lambda$ . As common in the academic benchmarking of EAs, our termination criterion is thus the state  $f(x) = \max\{f(y) \mid y \in \{0, 1\}^n\}$ .

**Strict vs. Non-strict Update Rules.** We have presented in the previous section the algorithms as originally suggested in the literature. However, in our initial experiments we have made an interesting observation that the  $(1 + \lambda)$  EA( $A, b$ ) can substantially benefit from a slightly different parameter update rule, which replaces  $p$  by  $Ap$  only if the best offspring  $y$  is *strictly better* than the parent, i.e., if it satisfies  $f(y) > f(x)$ . We perform all experiments for the strict and the classical (non-strict) update rules, which – together with the

two lower bounds  $p_{\min} = 1/n^2$  and  $p_{\min} = 1/n$  – yields four different settings for each benchmark problem. For reasons of space we can only comment on a few selected cases below. The detailed results are available at [5]. We mostly focus on the case of the strict update rule, if not stated otherwise.

### 3 Hybridizing Q-Learning and the 1/5-th Success Rule

The main contribution of our work is an algorithm that avoids the drawbacks of the above-mentioned  $(1 + \lambda)$  EA variants observed on ONEMAX, and shows stable performance for all values of  $\lambda$ . We will achieve this by hybridizing the  $(1 + \lambda)$  EA( $A, b$ ) with Q-learning.

**Q-learning** is a method that falls into the broader category of reinforcement learning (RL). Q-learning aims at learning, from the data that it observes, a policy that tells an *agent* which *action* to apply in a given situation. For this, it maintains a state-action matrix, in which it records its guess for what the expected *reward* of each action in each of the states is. For a given *state*  $s$ , the action  $a$  maximizing this expected reward is chosen and executed. The environment returns a numerical reward and a representation of its state. The reward is used to update the state-action matrix, according to some rules that we shall discuss in the next paragraphs. The Q-learning process repeats until some termination criterion is met. The goal of the agent is to maximize the total reward. A smooth introduction to RL can be found in [40].

**The  $(1 + \lambda)$  QEA.** We apply Q-learning to control the mutation rate of the  $(1 + \lambda)$  EA with fixed offspring population size  $\lambda$ . We first present in Algorithm 1 the basic  $(1 + \lambda)$  QEA. Its hybridization with the 1/5-th success rule will be explained further below. The  $(1 + \lambda)$  QEA considers only two actions: whether to multiply the current mutation rate  $p$  by the factor  $A > 1$  (action  $a_{\text{mult}}$ ) or whether to multiply it by the factor  $b < 1$  (action  $a_{\text{divide}}$ ). As mentioned in the introduction, the advantage of this action space is a smooth transition of the mutation rates between consecutive iterations, compared to a possibly abrupt change when operating directly on the parameter values.

We use as reward the relative fitness gain, i.e.,  $(\max f(y^{(i)}) - f(x))/f(x)$  (where we use the same notation as in the description of the  $(1 + \lambda)$  EA, i.e.,  $x$  denotes the parent individual and  $y^{(1)}, \dots, y^{(\lambda)}$  its  $\lambda$  offspring). This reward is computed in line 12. Note here that several other reward definitions would have been possible. We tried different suggestions made in [26] and found this variant to be the most efficient. The new state  $s'$  is computed as the number of offspring  $y^{(i)}$  that are strictly better than the parent (lines 13–16). With the reward and the new state at hand, the efficiency estimation  $Q(s, a)$  is updated in line 18, through a standard Q-learning update rule. Note here that action  $a$  is the one that was selected in the previous iteration (lines 20–23), and it resulted in moving from the previous state  $s$  to the current state  $s'$ .

After this update, the  $(1 + \lambda)$  QEA selects the action to be used in the next iteration, through simple greedy selection if possible, and through an unbiased

**Algorithm 1:** The  $(1 + \lambda)$  QEA, Q-learning highlighted in blue font

---

```

1 Input: population size  $\lambda$ , learning rate  $\alpha$ , learning factor  $\gamma$ ;
2 Initialization:
3    $x \leftarrow$  random string from  $\{0, 1\}^n$ ;
4    $p \leftarrow 1/n$ ;
5   for all states  $s_i \in [0 \dots \lambda]$  and all actions  $a_i \in \{a_{\text{mult}}, a_{\text{divide}}\}$  do
6      $Q(s_i, a_i) \leftarrow 0$ ;
7      $s, a \leftarrow$  undefined;
8 Optimization: while termination criterion not met do
9   for  $i = 1, \dots, \lambda$  do  $y^{(i)} \leftarrow$  mutate( $x, p$ );
10   $x^* \leftarrow \arg \max_{y^{(i)}} f(y^{(i)})$ ;
11   $x_{\text{old}} \leftarrow x$ ;
12  if  $f(x^*) \geq f(x)$  then  $x \leftarrow x^*$ ;
13   $r \leftarrow \frac{f(x^*)}{f(x_{\text{old}})} - 1$ ; // reward calculation
14   $s' \leftarrow 0$ ;
15  for  $i = 1, \dots, \lambda$  do
16    if  $f(y^{(i)}) > f(x_{\text{old}})$  then // state calculation
17       $s' \leftarrow s' + 1$ ;
18  if  $s \neq$  undefined and  $a \neq$  undefined then
19     $Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$ ;
20   $s \leftarrow s'$ ;
21  if  $Q(s', a_{\text{mult}}) = Q(s', a_{\text{divide}})$  then
22     $a \leftarrow$  select  $a_{\text{mult}}$  or  $a_{\text{divide}}$  equiprobably;
23  else
24     $a \leftarrow \arg \max_{a'} Q(s', a')$ ;
25   $p \leftarrow ap$ ; // update mutation rate
26   $p \leftarrow \min(\max(p_{\text{min}}, p), p_{\text{max}})$ ; // capping mutation rate

```

---

random choice otherwise; see lines 20–23. The mutation rate  $p$  is then updated by this action (line 24) and capped to remain within the interval  $[p_{\text{min}}, p_{\text{max}}]$  if needed (line 25).

*Hyperparameters.* The  $(1 + \lambda)$  QEA has six hyperparameters, the constant factors of the actions  $a_{\text{mult}}$  and  $a_{\text{divide}}$ , the upper and lower bounds for the mutation rate  $p_{\text{min}}$  and  $p_{\text{max}}$ , and two hyperparameters originating from the Q-learning methodology itself (line 18), the *learning rate*  $\alpha$  and the *discount factor*  $\gamma$ . In our experiments, we use  $a_{\text{mult}} = 2$ ,  $a_{\text{divide}} = 1/2$ ,  $p_{\text{max}} = 1/2$ ,  $\alpha = 0.8$ , and  $\gamma = 0.2$ . These values were chosen in a preliminary tuning step, details of which we have to leave for the full report due to space restrictions. For  $p_{\text{min}}$  we show results for two different values,  $1/n^2$  and  $1/n$ , just as we do for the other parameter control mechanisms.

**The  $(1 + \lambda)$  HQEA, the Hybrid Q-Learning EA.** In the hybridized  $(1 + \lambda)$  QEA, the  $(1 + \lambda)$  HQEA, we reconsider the situation when the  $Q(s, a)$

estimations are equal. This situation arises in two cases: when the state  $s$  is visited for the first time or when the same estimation was learned for both actions  $a_{\text{mult}}$  and  $a_{\text{divide}}$ . In these cases, the learning mechanism cannot decide which action is better, and an action is selected uniformly randomly. The  $(1+\lambda)$  HQEA, in contrast, borrows in this case the update rule from the  $(1+\lambda)$  EA( $A, b$ ) algorithm, i.e., action  $a_{\text{mult}}$  is selected if the best offspring is strictly better than the parent, otherwise  $a_{\text{divide}}$  is chosen. Formally, we obtain the  $(1+\lambda)$  HQEA by replacing in Algorithm 1 line 18 by the following text:

$$\mathbf{if} f(x^*) > f(x_{\text{old}}) \mathbf{then} a \leftarrow a_{\text{mult}} \mathbf{else} a \leftarrow a_{\text{divide}}. \quad (1)$$

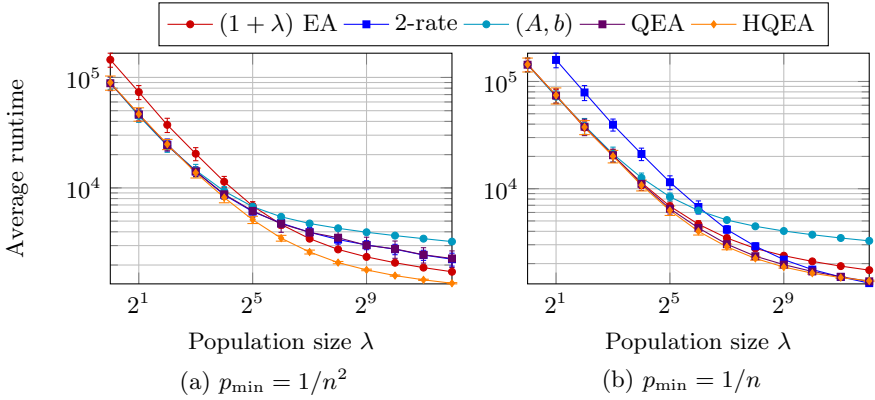
**Strict vs. Non-strict Update Rules.** As mentioned at the end of Sect. 2, we experiment both with a strict and a non-strict update rule. Motivated by the better performance of the strict update rule, the description of the  $(1+\lambda)$  QEA and the  $(1+\lambda)$  HQEA use this rule. The non-strict update rules can be obtained from Algorithm 1 by replacing the strict inequality in line 15 by the non-strict one. Similarly, for the  $(1+\lambda)$  HQEA, we also replace “ $\mathbf{if} f(x^*) > f(x_{\text{old}})$ ” in (1) by “ $\mathbf{if} f(x^*) \geq f(x_{\text{old}})$ ”.

## 4 Empirical Comparison of Parameter Control Algorithms

We now demonstrate that, despite the seemingly minor change, the  $(1+\lambda)$  HQEA outperforms both its origins, the  $(1+\lambda)$  QEA and the  $(1+\lambda)$  EA( $A, b$ ), on several benchmark problems. We recall that the starting point of our investigations were the results presented in [37], which showed that the performance of the  $(1+\lambda)$  EA variants discussed in Sect. 2 on ONEMAX strongly depends on (1) the offspring population size  $\lambda$ , and on (2) the bound  $p_{\text{min}}$  at which we cap the mutation rate. The  $(1+\lambda)$  HQEA, in contrast, is shown to yield stable performance for all tested values of  $\lambda$  and for both tested values of  $p_{\text{min}}$ .

**Experimental Setup.** All results shown below are simulated from 100 independent runs of each algorithm. We report statistics for the optimization time, i.e., for the random variable counting the number of steps needed until an optimal solution is queried for the first time. Since the value of  $\lambda$  is static, we report the optimization times as number of generations; classical running time in terms of function evaluations can be obtained from these values by multiplying with  $\lambda$ . For ONEMAX, we report average optimization times, for consistency with the results in [37] and with theoretical results. However, for some of the other benchmark problems, the dispersion of the running times can be quite large, so that we report median values and interquartile ranges instead. Please also note that we use logarithmic scales in all runtime plots.

In the cases of large dispersion, we also performed the rank-sum Wilcoxon test to question statistical significance [8]. More precisely, we compared the  $(1+\lambda)$  HQEA to each of the other algorithms. As the input data for the test, the



**Fig. 1.** Average number of generations and its standard deviation needed to locate the optimum of the ONEMAX problem

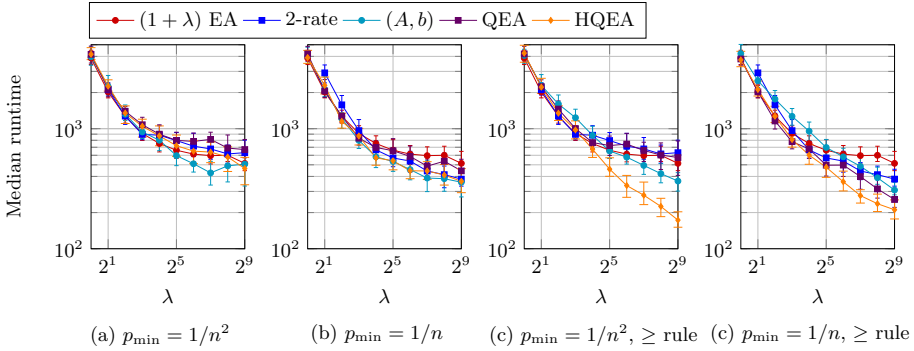
runtimes of all 100 runs of each of the two compared algorithms were used. The significance level was set to  $p_0 = 0.01$ .

The value of  $\lambda$  is parameterized as  $2^t$ , with  $t$  taking all integer values ranging from 0 to 12 for ONEMAX and from 0 to 9 for all other problems. The problem dimension, in contrast, is chosen in a case-by-case basis. We recall that it was shown in [37] that the dimension did not have any influence on the ranking of the algorithms on ONEMAX. This behavior can be confirmed for the here-considered algorithm portfolio (results not shown due to space limitations).

### 4.1 Stable Performance on OneMax

Figure 1 summarizes our empirical results for the  $10^4$ -dimensional ONEMAX problem, the problem of maximizing the function  $OM : \{0, 1\}^n \rightarrow [0..n], x \mapsto \sum_{i=1}^n x_i$ . For  $p_{\min} = 1/n^2$ , our key findings can be summarized as follows. **(i)** For small  $\lambda$  up to  $2^4$ , all the parameter control algorithms perform similarly and all of them seem to be significantly better than the  $(1 + \lambda)$  EA with static mutation rates. **(ii)** Starting from  $\lambda > 2^5$  for the  $(1 + \lambda)$  EA( $A, b$ ) and from  $\lambda > 2^6$  for the  $(1 + \lambda)$  QEA and the  $(1 + \lambda)$  EA $_{r/2, 2r}$ , these algorithms are outperformed by the  $(1 + \lambda)$  EA. **(iii)** The  $(1 + \lambda)$  HQEA is the only parameter control algorithm that substantially improves the performance of the  $(1 + \lambda)$  EA for all considered values of  $\lambda$ . The advantage varies from 21% for  $\lambda = 2^{12}$  to 38% for  $\lambda = 1$ .

For the less generous  $p_{\min} = 1/n$  lower bound, we observe the following. **(i)** Overall, the performance is worsened compared to the  $1/n^2$  lower bound. In particular, for small values of  $\lambda$ , most of the algorithms are indistinguishable from the  $(1 + \lambda)$  EA, except for the  $(1 + \lambda)$  EA $_{r/2, 2r}$ , which is even substantially worse. **(ii)** However, for  $\lambda \geq 2^9$ , the  $(1 + \lambda)$  EA $_{r/2, 2r}$  starts to outperform the  $(1 + \lambda)$  EA, in strong contrast to the situation for the  $1/n^2$  lower bound. **(iii)** Our  $(1 + \lambda)$  HQEA is the only method which is never worse than the  $(1 + \lambda)$  EA and still outperforms it for  $\lambda > 2^6$ . With the growth of  $\lambda$ , the advantage grows as



**Fig. 2.** Median number of generations and the corresponding interquartile ranges needed to locate the optimum of the NEUTRALITY problem

well: while the  $(1 + \lambda)$  EA with  $\lambda = 2^{12}$  needs 1738 generations, on average, the  $(1 + \lambda)$  HQEA only requires 1379 generations, an advantage of more than 20%. (iv) It is worth noting that the  $(1 + \lambda)$  QEA in this case performs on par with the  $(1 + \lambda)$  HQEA.

Overall, we thus see that the  $(1 + \lambda)$  HQEA is the only considered parameter control algorithm, which stably performs on par or better than the  $(1 + \lambda)$  EA and all of the other algorithms for all values of  $\lambda$  and for both values  $p_{\min} \in \{1/n^2, 1/n\}$ .

### 4.2 Stable Performance on Other Benchmark Problems

**LeadingOnes.** The LEADINGONES problem asks to maximize functions of the type  $LO_{z,\sigma} : \{0,1\}^n \rightarrow \mathbb{R}, x \mapsto \max\{i \in [n] \mid \forall j \leq i : x_{\sigma(j)} = z_{\sigma(j)}\}$ , where  $\sigma$  is simply a permutation of the indices  $1, \dots, n$  (the classic LO function uses the identity). We study the  $n = 10^3$ -dimensional variant of this problem.

For  $p_{\min} = 1/n^2$  all the methods – including the  $(1 + \lambda)$  EA – show very similar performance, with the difference between the best and the worst of the five algorithms varying from 3% to 6% for each offspring population size  $\lambda$ , which is of the same order as the corresponding standard deviations. For the  $1/n$  lower bound, the situation is similar, except that the  $(1 + \lambda)$  EA<sub>r/2,2r</sub> performs substantially worse than the  $(1 + \lambda)$  EA for all considered values of  $\lambda$ , and the difference varies from 45% to 93%.

As a result, the  $(1 + \lambda)$  HQEA generally performs on par with the  $(1 + \lambda)$  EA for all considered values of  $\lambda$  and both considered lower bounds on the mutation rate. Particularly, for  $p_{\min} = 1/n^2$  it is strictly better in 6 of the 10 cases, and in the other cases the disadvantages are 0.3%, 0.3%, 0.7%, and 1.1%.

**Neutrality.** The NEUTRALITY function is a W-model transformation [43] that we apply to ONEMAX. It is calculated the following way: a bit string  $x$  is split into blocks of length  $k$  each, and each block contributes 0 or 1 to the fitness value

according to the majority of values within the block. In line with [43] and [17] we considered  $k = 3$ . We study the  $n = 10^3$ -dimensional version of this problem. The results are summarized in Fig. 2.

For  $p_{\min} = 1/n^2$  we obtain the following observations. Most of the parameter control methods perform poorly, i.e. worse than the  $(1 + \lambda)$  EA. The exception is  $(1 + \lambda)$  EA( $A, b$ ), which performs better than the  $(1 + \lambda)$  EA for several values of  $\lambda$  (in particular,  $\lambda = 2^6, 2^7$ ).

The lower bound  $p_{\min} = 1/n$  turns out to be preferable for all the algorithms: for large offspring population sizes  $\lambda$ , they all perform better than the standard  $(1 + \lambda)$  EA. Our  $(1 + \lambda)$  HQEA is usually one of the best algorithms, but however, for  $\lambda = 2^7$  and  $\lambda = 2^8$  it seems to be worse than the  $(1 + \lambda)$  EA( $A, b$ ). The Wilcoxon test results did not confirm the significance of this difference though (the p-values are greater than 0.04 in both cases).

For this problem we also observe that switching from the strict update rule to the non-strict version is beneficial for the  $(1 + \lambda)$  HQEA, the  $(1 + \lambda)$  QEA, and the  $(1 + \lambda)$  EA( $A, b$ ), regardless of the value of  $p_{\min}$ . It is worth noting that with these values of hyper-parameters the  $(1 + \lambda)$  HQEA performs significantly better on high values of the population size ( $\lambda \geq 2^5$ ) than all the other considered methods (the p-values are between  $1.6 \cdot 10^{-9}$  and  $3.9 \cdot 10^{-18}$ ).

**Plateaus.** Plateau is an extension of the W-model suggested in [17]. This transformation operates on the function values, by setting  $\text{PLATEAU}(f(x)) := \lfloor f(x)/k \rfloor + 1$ , for a parameter  $k$  that determines the size of the plateau. We superpose this transformation to ONEMAX, and study performances for dimension  $n = 1000$ .

*Small Plateaus,  $k = 2$ .* For  $k = 2$ ,  $p_{\min} = 1/n^2$ , and  $2 \leq \lambda \leq 2^6$ , all considered parameter control algorithms improve the performance of the  $(1 + \lambda)$  EA. For large values of  $\lambda$  (starting from  $\lambda = 2^7$ ), however, the runtimes of the  $(1 + \lambda)$  EA and the parameter control algorithms are hardly distinguishable. The only exception for large  $\lambda$  is the proposed  $(1 + \lambda)$  HQEA, which performs a bit better than the  $(1 + \lambda)$  EA. The Wilcoxon test suggests that the difference is significant with the p-values less than  $3.9 \cdot 10^{-18}$ .

The results obtained when using  $p_{\min} = 1/n$  are less successful, as most of the parameter control methods just perform on par with the  $(1 + \lambda)$  EA in this case. The  $(1 + \lambda)$  HQEA shows nevertheless a stable and comparatively good performance for all offspring population sizes  $\lambda$ . The  $(1 + \lambda)$  EA $_{r/2, 2r}$  performs worse than the  $(1 + \lambda)$  EA in this case.

*Plateaus with  $k = 3$ .* We also considered a harder version of the problem with a larger size of the plateau, for which we use  $k = 3$ . As the total running time for this problem is much larger than for  $k = 2$ , we had to restrict our experiments to a smaller problem size  $n = 100$ .

For  $p_{\min} = 1/n^2$  we cannot see any clear improvement of parameter control over the  $(1 + \lambda)$  EA any more. Moreover, for  $\lambda \geq 2^7$ , the  $(1 + \lambda)$  EA seems to be the best performing algorithm.

Interestingly, for the  $1/n$  lower bound the situation is pretty similar to the  $k = 2$  case. All the parameter control algorithms perform on par with the  $(1 + \lambda)$  EA



(with only slight differences at  $\lambda = 2^4, 2^6$ ), except for the  $(1 + \lambda)$  EA $_{r/2, 2r}$ , which performs worse. It seems that as the problem gets harder, a larger lower bound is preferable, which seems to be natural, as with a bigger plateau, a higher mutation rate is needed to leave it. Let us also mention that the  $(1 + \lambda)$  HQEA performs stably well for all considered values of  $\lambda$  in this preferable configuration.

**Ruggedness.** We also considered the W-Model extension F9 from [17], which adds local optima to the fitness landscape by mapping the fitness values to  $r_2(f(x)) := f(x) + 1$  if  $f(x) \equiv n \pmod{2}$  and  $f(x) < n$ ,  $r_2(f(x)) := \max\{f(x) - 1, 0\}$  for  $f(x) \equiv n + 1 \pmod{2}$  and  $f(x) < n$ , and  $r_2(n) := n$ . This transformation is superposed on ONEMAX of size  $n = 100$ .

For  $p_{\min} = 1/n^2$ , all the considered parameter control algorithms significantly worsen the performance of the  $(1 + \lambda)$  EA. Even the  $(1 + \lambda)$  EA $_{r/2, 2r}$ , which, untypically, performs the best among all these algorithms, is still significantly worse than the  $(1 + \lambda)$  EA.

The situation improves for  $p_{\min} = 1/n$  and the parameter control algorithms show similar performance as the  $(1 + \lambda)$  EA. The only exception is again  $(1 + \lambda)$  EA $_{r/2, 2r}$ , whose performance did not change much compared to the case  $p_{\min} = 1/n^2$ .

## 5 Conclusions and Future Work

To address the issue of unstable performance of several parameter control algorithms on different values of population size reported in [37], we proposed the Q-learning based parameter control algorithm, the  $(1 + \lambda)$  QEA, and its hybridization with the  $(1 + \lambda)$  EA( $A, b$ ), the  $(1 + \lambda)$  HQEA. The algorithms were compared empirically on ONEMAX and five more benchmark problems with different characteristics, such as neutrality, plateaus and presence of local optima. Our main findings may be summarized as follows.

On simple problems, i.e. ONEMAX, LEADINGONES, and PLATEAU with  $k = 2$  the  $(1 + \lambda)$  HQEA is the only algorithm which always performs on par or better than the other tested algorithms for all the considered values of  $\lambda$  and both mutation rate lower bounds.

On the harder problems, i.e., NEUTRALITY, PLATEAU with  $k = 3$ , and RUGGEDNESS, the  $(1 + \lambda)$  HQEA performance depends on the lower bound (the same is true for the other algorithms). For  $p_{\min} = 1/n$ , the  $(1 + \lambda)$  HQEA still performs on par with or better than the other algorithms for all values of  $\lambda$  in almost all cases.

The  $(1 + \lambda)$  QEA is usually worse than the  $(1 + \lambda)$  HQEA. There are a number of examples where  $(1 + \lambda)$  EA( $A, b$ ) is significantly worse as well. The hybridization of these two algorithms seems to be essential for the observed good performance of the  $(1 + \lambda)$  HQEA.

As next steps, we plan on investigating *more possible actions* for the Q-learning part. For example, one may use several different multiplicative update rules, to allow for a faster adaptation when the current rate is far from optimal. This might in particular be relevant in *dynamic environments*, in which the

fitness functions (and with it the optimal parameter values) change over time. We also plan on identifying ways to automatically select the configuration of the Q-learning algorithms, with respect to its hyper-parameters, but also with respect to whether to use the strict or the non-strict update rule. In this context, we are investigating exploratory landscape analysis [33, 41].

**Acknowledgments.** The reported study was funded by RFBR and CNRS, project number 20-51-15009, by the Paris Ile-de-France Region, and by a public grant as part of the Investissement d’avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH.

## References

1. Aleti, A., Moser, I.: Entropy-based adaptive range parameter control for evolutionary algorithms. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2013), pp. 1501–1508 (2013)
2. Bäck, T.: The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In: Proceedings of Parallel Problem Solving from Nature (PPSN 1992), pp. 87–96. Elsevier (1992)
3. Bartz-Beielstein, T., Flasch, O., Koch, P., Konen, W.: SPOT: a toolbox for interactive and automatic tuning in the R environment. In: Proceedings of the 20th Workshop on Computational Intelligence, pp. 264–273. Universitätsverlag Karlsruhe (2010)
4. Belkhir, N., Dréo, J., Savéant, P., Schoenauer, M.: Per instance algorithm configuration of CMA-ES with limited budget. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2017), pp. 681–688. ACM (2017)
5. Buzdalova, A., Doerr, C., Rodionova, A.: Hybridizing the 1/5-th success rule with Q-learning for controlling the mutation rate of an evolutionary algorithm (2020). <http://arxiv.org/abs/2006.11026>
6. Carvalho Pinto, E., Doerr, C.: Towards a more practice-aware runtime analysis of evolutionary algorithms (2018). <https://arxiv.org/abs/1812.00493>
7. Costa, L.D., Fialho, Á., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2008), pp. 913–920. ACM (2008)
8. Derrac, J., Garcia, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **1**(1), 3–18 (2011)
9. Devroye, L.: The compound random search. Ph.D. dissertation, Purdue University, West Lafayette, IN (1972)
10. Doerr, B., Doerr, C.: Theory of parameter control for discrete black-box optimization: provable performance gains through dynamic parameter choices. *Theory of Evolutionary Computation*. NCS, pp. 271–321. Springer, Cham (2020). <https://doi.org/10.1007/978-3-030-29414-4.6>. Also available online at <https://arxiv.org/abs/1804.05650>
11. Doerr, B.: Analyzing randomized search heuristics via stochastic domination. *Theor. Comput. Sci.* **773**, 115–137 (2019)
12. Doerr, B., Doerr, C., Lengler, J.: Self-adjusting mutation rates with provably optimal success rules. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2019). ACM (2019)

13. Doerr, B., Doerr, C., Yang, J.:  $k$ -bit mutation with self-adjusting  $k$  outperforms standard bit mutation. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 824–834. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_77](https://doi.org/10.1007/978-3-319-45823-6_77)
14. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2016), pp. 1123–1130. ACM (2016)
15. Doerr, B., Gießen, C., Witt, C., Yang, J.: The  $(1 + \lambda)$  evolutionary algorithm with self-adjusting mutation rate. *Algorithmica* **81**, 593–631 (2019). <https://doi.org/10.1007/s00453-018-0502-x>
16. Doerr, C., Wagner, M.: On the effectiveness of simple success-based parameter selection mechanisms for two classical discrete black-box optimization benchmark problems. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2018), pp. 943–950. ACM (2018)
17. Doerr, C., Ye, F., Horesh, N., Wang, H., Shir, O.M., Bäck, T.: Benchmarking discrete optimization heuristics with IOHprofiler. *Appl. Soft Comput.* **88**, 106027 (2020)
18. Doerr, C., Ye, F., van Rijn, S., Wang, H., Bäck, T.: Towards a theory-guided benchmarking suite for discrete black-box optimization heuristics: profiling  $(1 + \lambda)$  EA variants on onemax and leadingones. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2018), pp. 951–958. ACM (2018)
19. Droste, S., Jansen, T., Wegener, I.: On the analysis of the  $(1+1)$  evolutionary algorithm. *Theor. Comput. Sci.* **276**, 51–81 (2002)
20. Eiben, A.E., Horvath, M., Kowalczyk, W., Schut, M.C.: Reinforcement learning for online control of evolutionary algorithms. In: Proceedings of the 4th International Conference on Engineering Self-Organising Systems, pp. 151–160 (2006)
21. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* **3**, 124–141 (1999)
22. Falkner, S., Klein, A., Hutter, F.: BOHB: robust and efficient hyperparameter optimization at scale. In: Proceedings of International Conference on Machine Learning (ICML 2018), pp. 1436–1445 (2018)
23. Fialho, Á., Costa, L.D., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. *Ann. Math. Artif. Intell.* **60**, 25–64 (2010)
24. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-25566-3\\_40](https://doi.org/10.1007/978-3-642-25566-3_40)
25. Karafotias, G., Eiben, Á.E., Hoogendoorn, M.: Generic parameter control with reinforcement learning. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2014), pp. 1319–1326 (2014)
26. Karafotias, G., Hoogendoorn, M., Eiben, A.E.: Evaluating reward definitions for parameter control. In: Mora, A.M., Squillero, G. (eds.) EvoApplications 2015. LNCS, vol. 9028, pp. 667–680. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-16549-3\\_54](https://doi.org/10.1007/978-3-319-16549-3_54)
27. Karafotias, G., Hoogendoorn, M., Eiben, A.: Parameter control in evolutionary algorithms: trends and challenges. *IEEE Trans. Evol. Comput.* **19**, 167–187 (2015)
28. 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**, 77–112 (2004)

29. Lässig, J., Sudholt, D.: Adaptive population models for offspring populations and parallel evolutionary algorithms. In: Proceedings of Foundations of Genetic Algorithms (FOGA 2011), pp. 181–192. ACM (2011)
30. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: a novel bandit-based approach to hyperparameter optimization. arXiv preprint [arXiv:1603.06560](https://arxiv.org/abs/1603.06560) (2016)
31. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence, vol. 54. Springer, Heidelberg (2007). <https://doi.org/10.1007/978-3-540-69432-8>
32. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Stützle, T., Birattari, M.: The irace package: iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **3**, 43–58 (2016)
33. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of Genetic and Evolutionary Conference (GECCO 2011), pp. 829–836. ACM (2011)
34. Müller, S.D., Schraudolph, N.N., Koumoutsakos, P.D.: Step size adaptation in evolution strategies using reinforcement learning. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), pp. 151–156 (2002)
35. Pushak, Y., Hoos, H.: Algorithm configuration landscapes: more benign than expected? In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 271–283. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_22](https://doi.org/10.1007/978-3-319-99259-4_22)
36. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Fromman-Holzboorg Verlag, Stuttgart (1973)
37. Rodionova, A., Antonov, K., Buzdalova, A., Doerr, C.: Offspring population size matters when comparing evolutionary algorithms with self-adjusting mutation rates. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2019), pp. 855–863. ACM (2019)
38. Rost, A., Petrova, I., Buzdalova, A.: Adaptive parameter selection in evolutionary algorithms by reinforcement learning with dynamic discretization of parameter range. In: Proceedings of Genetic and Evolutionary Computation Conference Companion (GECCO 2016), pp. 141–142 (2016)
39. Schumer, M.A., Steiglitz, K.: Adaptive step size random search. *IEEE Trans. Autom. Control* **13**, 270–276 (1968)
40. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
41. Vérel, S.: Apport à l’analyse des paysages de fitness pour l’optimisation mono-objective et multiobjective: Science des systèmes complexes pour l’optimisation par méthodes stochastiques. (Contributions to fitness landscapes analysis for single- and multi-objective optimization: Science of complex systems for optimization with stochastic methods) (2016). <https://tel.archives-ouvertes.fr/tel-01425127>
42. Wang, H., Emmerich, M., Bäck, T.: Cooling strategies for the moment-generating function in Bayesian global optimization. In: Proceedings of Congress on Evolutionary Computation (CEC 2018), pp. 1–8 (2018)
43. Weise, T., Wu, Z.: Difficult features of combinatorial optimization problems and the tunable w-model benchmark problem for simulating them. In: Proceeding of Genetic and Evolutionary Computation Conference Companion (GECCO 2018), pp. 1769–1776 (2018)



# Fitness Landscape Features and Reward Shaping in Reinforcement Learning Policy Spaces

Nathaniel du Preez-Wilkinson<sup>(✉)</sup> and Marcus Gallagher

School of Information Technology and Electrical Engineering,  
The University of Queensland, Brisbane, QLD 4072, Australia  
{uqndupre,marcusg}@uq.edu.au

**Abstract.** Reinforcement learning (RL) algorithms have received a lot of attention in recent years. However, relatively little work has been dedicated to analysing RL problems; which are thought to contain unique challenges, such as sparsity of the reward signal. Reward shaping is one approach that may help alleviate the sparse reward problem.

In this paper we use fitness landscape features to study how reward shaping affects the underlying optimisation landscape of RL problems. Our results indicate that features such as deception, ruggedness, searchability, and symmetry can all be greatly affected by reward shaping; while neutrality, dispersion, and the number of local optima remain relatively invariant. This may provide some guidance as to the potential effectiveness of reward shaping for different algorithms, depending on what features they are sensitive to. Additionally, all of the reward functions we studied produced policy landscapes that contain a single local optimum and very high neutrality. This suggests that algorithms that explore spaces globally, rather than locally, may perform well on RL problems; and may help explain the success of evolutionary methods on RL problems. Furthermore, we suspect that the high neutrality of these landscapes is connected to the issue of reward sparsity in RL.

**Keywords:** Fitness landscapes · Global features · Reinforcement learning · Reward shaping

## 1 Introduction

Reinforcement learning (RL) is an optimisation problem. The goal is to find an optimal policy,  $\pi$ , that maximises the return,  $R(\pi)$ . Unlike supervised learning, a meaningful training signal is not available for every input-output pair: reward functions are sparse. One technique for dealing with this is reward shaping.

Reward shaping is the act of replacing the reward function for a problem with a surrogate that is *intended* to be easier to learn, while resulting in the same optimal behaviour. We investigate how changing the reward function for an RL problem affects the underlying optimisation landscape, using features from the fitness landscape literature.

The main contributions of this paper are two-fold: 1) we extend the work presented in [21] to analyse more features of RL problems; and 2) we examine the effects of reward shaping on RL optimisation landscapes.

## 2 Related Work

Evolutionary algorithms (EAs) have been applied to RL for over 20 years [13], and have demonstrated performance on par with modern deep RL algorithms [3, 18, 26]. Fitness landscapes are a concept that is used to analyse optimisation problems, usually from the EA literature [12]. Despite the connection between EAs and RL, there has been almost no work that studies RL problems using fitness landscape analysis.

Recently, local optima networks [14] were used to analyse the structure, modality, fitness distribution, and neutrality of RL problems [21]. Two key findings from this work are: 1) RL problems appear to have high levels of neutrality; and 2) the problems studied contained only one local optimum. We extend this work by analysing the additional features of deception [7], dispersion [10], ruggedness [11], symmetry [23], and searchability [9]. We also investigate the effects of reward shaping on fitness landscape features.

Work has been done in recent years analysing RL parameter spaces. Ilyas et al. [6] visualise the parameter landscape of the Humanoid-v2 MuJoCo task locally<sup>1</sup> using 1) steps in the algorithm (PPO [20]) direction, and 2) steps in a random direction as two axes over which the fitness varies. Ahmed et al. [1] investigate the effects of stochastic policies by sampling from a hypersphere around a point in parameter space, and by interpolating between two points in the space. Very recently, Oller et al. [15] studied the fitness distributions of several neural network parameter spaces using random weight guessing [19]. Our work differs from this recent work in the following ways: 1) we analyse the model-independent policy space of problems; 2) we calculate features from the fitness landscape literature; and 3) we investigate the effects of reward shaping on the landscape.

## 3 Background

### 3.1 Fitness Landscapes

A discrete fitness landscape consists of a set of solutions,  $x \in X$ , a neighbourhood function,  $N(x)$ , and a fitness function,  $f(x) : X \rightarrow \mathbb{R}$ . The set of solutions contains all possible solutions for a problem, good and bad. The neighbourhood function takes a candidate solution,  $x$ , and returns the set of neighbours of  $x$ . The fitness function takes a solution and scores it, returning a real number that represents how “good” the solution is.

We describe some global features of fitness landscapes in Sect. 4.

---

<sup>1</sup> Relative to a point in the parameter space.

### 3.2 Reinforcement Learning

In an RL problem, an agent interacts with an environment; and, through trial and error, learns the best way to do so [22]. The environment can take on one of a finite number of *states*,  $s \in S$ , and the agent interacts with the environment through *actions*,  $a \in A$ , that transition the environment from the current state,  $s_t$ , to a new state,  $s_{t+1}$ . After an agent’s action changes the state of the environment, the agent receives feedback in the form of a *reward*,  $r_t = r(s_t, a_t, s_{t+1})$ , that indicates how desirable the choice of that action was while in that state. The goal of RL is to develop an optimal *policy*,  $\pi : S \rightarrow A$ , a mapping from states to actions, for a given problem. An optimal policy is one which maximises the total cumulative reward<sup>2</sup>,  $R$ :

$$R = \sum_{t=0}^T r(s_t, a_t, s_{t+1})$$

where  $T$  is the total time the agent interacts with the environment.

All of the RL problems studied in this work abide by the following assumptions: 1) discrete state space; 2) discrete action space; 3) discrete time; 4) finite time limit; and 5) deterministic environment.

**Reinforcement Learning Fitness Landscapes.** There are currently two high-level choices for how to define an RL fitness landscape: over *parameter space*, or over *policy space*. In both cases, the set of solutions is the set of possible policies,  $\pi \in \Pi$ , and the fitness function is the return,  $R(\pi)$ . The two differ in the encoding of the policies, and the definition of the neighbourhood function.

In parameter space, we consider the return of a parameterised policy,  $R(\pi_\theta)$ , as the return of the parameters,  $R(\theta)$ , and we define a neighbourhood function on them. This results in landscapes that are dependent on the choice of agent model.

In policy space, we represent a policy as a direct lookup table from states to actions. This can be equivalently represented as a string of actions, with each state specifying a unique index. For the neighbourhood function, we follow the example of [21] and define two policies as neighbours if the Hamming Distance between them is equal to one.

We use policy space rather than parameter space in order to analyse RL problems independently from the choice of algorithm and agent model.

**Reward Shaping.** Reward shaping [25] is the act of replacing the reward function of a problem,  $r(s, a, s')$ , with a surrogate reward function,  $r'(s, a, s')$ , that is *intended* to be easier to learn, while resulting in the same optimal policy. However, it is not uncommon to encounter “reward shaping surprises” where the shaped reward function accidentally results in a different optimal policy. We are interested in how reward shaping affects underlying optimisation landscapes.

<sup>2</sup> Note that we are using the total reward ([22] Eqn 3.1) instead of the more commonly used discounted reward ([22] Eqn 3.2). We are able to do this because we guarantee that all our problems terminate in finite time. This eliminates the hyper-parameter  $\gamma$ , and is consistent with previous work on RL landscape features [21].

## 4 Fitness Landscape Features

In this section we describe the features that we have used, and how we have calculated them for RL problems. See [12] for a more thorough treatment of fitness landscape features, and the various ways to quantify them.

### 4.1 Modality

The modality of a landscape refers to the number of local optima. A landscape is unimodal if it contains only one local optimum<sup>3</sup>, and multimodal if it contains multiple local optima. We use the definition of local optimum presented in [5], which counts connected regions of equal fitness as a single optimum. We quantify modality by counting the number of local maxima and minima in a landscape.

### 4.2 Fitness Distribution

The fitness distribution of a landscape measures the frequency of occurrence of different fitness values. To calculate this, we enumerate the search space and calculate the fraction of solutions with each fitness value.

### 4.3 Searchability

We quantify searchability using *Accumulated Escape Probability* [9]. In [9] a sample of points is collected using Metropolis-Hastings sampling. For each solution in the sample, the fraction of neighbours with higher fitness is calculated. The Accumulated Escape Probability is calculated as the mean of these fractions. We do the same, but over the entire search space instead of a sample. The searchability value that we calculate can be interpreted as the fraction of neighbours with higher fitness for the “average” solution in the search space.

### 4.4 Neutrality

A fitness landscape is said to have high neutrality if it contains a large number of connected solutions with the same fitness. Such a landscape looks very “flat”, and often contains plateaus with steep discontinuous jumps in fitness. We measure neutrality using an extension of the *average neutrality ratio* presented in [24].

The neutrality ratio,  $n_r$ , of a solution,  $x$ , is the fraction of neighbours of  $x$  that share the same fitness.

$$n_r(x) = \frac{\sum_{x' \in N(x)} \delta_{f(x)f(x')}}{|N(x)|}$$

The average neutrality ratio in [24] is calculated as the mean of the neutrality ratios within a given neutral network. We introduce the “*Accumulated Neutrality Ratio*” ( $n_r^a$ ) as the mean neutrality ratio over the entire search space.

<sup>3</sup> Which is also the global optimum.



$$n_r^a(X) = \frac{1}{|X|} \sum_{x \in X} n_r(x)$$

The value of  $n_r^a$  can be interpreted as the fraction of neighbours with equal fitness for the “average” solution in the search space.

### 4.5 Ruggedness

Ruggedness is related to how smooth a landscape is. A landscape that is very rugged will have a lot of local “bumps” up and down in fitness value [11]. We quantify ruggedness using the entropy measure provided in [11].

A random walk of length  $L + 1$  is conducted on the landscape, and the fitness values are recorded in a string,  $F = f_0 f_1 \dots f_L$ . Each pair of adjacent values is assigned an encoding,  $\Psi_i$ , depending on the difference between them:

$$\Psi_i(\epsilon) = \begin{cases} \bar{1} & \text{if } f_i - f_{i-1} < -\epsilon \\ 0 & \text{if } |f_i - f_{i-1}| \leq \epsilon \\ 1 & \text{if } f_i - f_{i-1} > \epsilon \end{cases}$$

Each pair of adjacent encodings,  $(\Psi_{i-1}, \Psi_i)$ , is then classified as neutral (00), smooth ( $\bar{1}\bar{1}, 11$ ), or rugged (01, 0 $\bar{1}$ , 10, 1 $\bar{1}$ ,  $\bar{1}0, \bar{1}1$ ). An entropic measure is defined:

$$H(\epsilon) = - \sum_{p=\bar{1},0,1} \sum_{q=\bar{1},0,1} \frac{n_{[pq]}}{L} \log_6 \frac{n_{[pq]}}{L} \mid p \neq q$$

where  $n_{[pq]}$  is the number of  $pq$  pairs. The ruggedness measure,  $R_f$ , is [11]:

$$R_f = \max_{\forall \epsilon \in [0, \epsilon^*]} \{H(\epsilon)\}$$

where  $\epsilon^*$  is the minimum value for  $\epsilon$  that makes the landscape appear completely flat. Large values of  $R_f$  indicate more ruggedness, and small values indicate less. We take  $\epsilon^*$  to be the difference between the maximum and minimum fitness values for a problem. We average  $R_f$  over 100 random walks of length 1000.

### 4.6 Deception

Deception is an indication of how a landscape may lead search algorithms to explore away from global optima. We use the *fitness distance correlation* measure from [7] to quantify deception.

For every solution,  $x$ , in the space, the fitness,  $f(x)$ , and distance,  $d(x, x')$ , to the nearest global optimum are calculated as a pair. The correlation between these pairs is then calculated using the Pearson correlation coefficient. Results are in the range  $[-1, 1]$ . Values closer to  $-1$  indicate less deception, and values closer to  $+1$  indicate more deception<sup>4</sup>.

---

<sup>4</sup> For a maximisation problem.

## 4.7 Dispersion

The dispersion metric [10] measures how spread out high fitness solutions are. We start by calculating the mean pairwise distance,  $\bar{d}_1$ , between a uniform sample of  $n$  solutions,  $x_1 \dots x_n$  (we calculate  $\bar{d}_1$  from the best  $n$  of  $n$  solutions). We then collect a second sample of size  $m > n$ , and calculate the mean pairwise distance,  $\bar{d}_2$ , between the solutions with the top  $n$  fitness values (we calculate  $\bar{d}_2$  from the best  $n$  of  $m$  solutions). We calculate the dispersion metric as:

$$\text{Dispersion} = \frac{\bar{d}_2 - \bar{d}_1}{|S|}$$

We divide by the number of RL states in order to normalise<sup>5,6</sup> the result into  $[-1, 1]$ . In our experiments  $n = 100$  and  $m = |X|$ . Values closer to  $+1$  indicate that high fitness solutions are spread out. Values closer to  $-1$  indicate that high fitness solutions lie closer together.

## 4.8 Symmetry

Symmetry refers to transformations of a landscape under which the fitness function is invariant. We use the extension of spin-flip symmetry to alphabets with more than two characters that is proposed in [23].

We consider an alphabet as an ordered list of characters. A solution is a string that maps position  $i$  in the string to the character at index  $j$  in the alphabet. We consider a symmetric transformation as a permutation on the order of characters in the alphabet, while keeping the solution mappings the same. e.g. The string “aaab” with alphabet (‘a’, ‘b’, ‘c’) becomes “bbba” under alphabet (‘b’, ‘a’, ‘c’).

We can think of a transformation on the alphabet as defining a “symmetry neighbourhood”. We quantify symmetry as the Accumulated Neutrality Ratio over the space using this neighbourhood. These symmetry values can be interpreted as the fraction of transformations under which the fitness of the “average” solution remains invariant.

We calculate three different types of symmetry: “*Symmetry G*” - symmetry calculated over the entire, **G**lobal, space; “*Symmetry GM*” - symmetry over the subspace containing solutions with fitness **G**reater than the **M**inimum; and “*Symmetry O*” - symmetry over the subspace containing only **O**ptimal solutions.

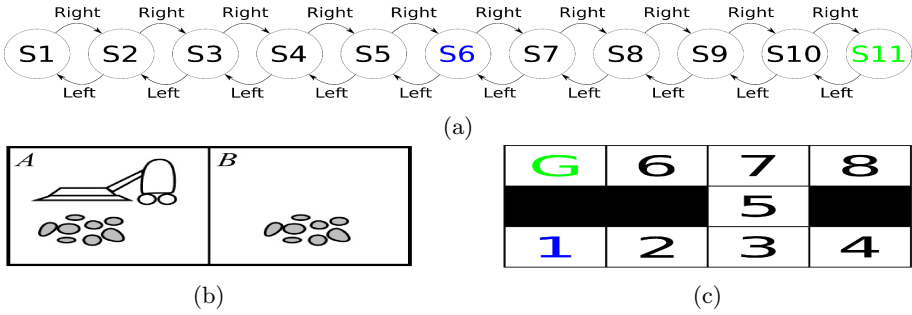
## 5 Environments

Reinforcement learning policy spaces grow exponentially as  $|A|^{|S|}$ , and the number of neighbours for each solution grows as<sup>7</sup>  $(|A| - 1)|S|$ . We seek to calculate features exactly (where possible) by enumerating entire policy spaces. This restricts the size of problems that we are able to analyse.

<sup>5</sup> The original metric from [10] is not normalised.

<sup>6</sup> The maximum Hamming distance between two strings is equal to their length, and our solutions are strings of length  $|S|$ .

<sup>7</sup> With the neighbourhood function that we are using.



**Fig. 1.** Visualisations of the four environments used in our experiments: (a) the 1D Chain; (b) the Vacuum World from [17]; and (c) the layout for the two mazes. The black squares in the 2D maze represent either walls or hazards, depending on the context.

### 5.1 1D Chain

The first environment we study is a one dimensional chain with eleven states. The agent starts in the middle of the chain, and can move left or right. This yields a policy space with  $|X| = |A|^{|S|} = 2^{11} = 2048$  possible solutions. Each solution in the space has 11 neighbours. The goal is to get to the right-most state, within a time limit of  $T = 5$  time steps. We are interested in this problem as a first look into the effects of reward shaping. This problem is simple enough that we should be able to easily understand the effects of different reward functions – there should be no “reward shaping surprises”, as described in Sect. 3.2. The problem is visualised in Fig. 1a.

For the 1D chain problem, we use nine reward functions  $(r_0^{1D}, r_1^{1D}, \dots, r_8^{1D})$ :

$$r_i^{1D}(s, a, s') = \begin{cases} 1 & a = \text{RIGHT} \wedge s' \in S_G(i) \\ 0 & \text{otherwise} \end{cases}$$

where  $S_G(i)$  is defined such that

$$s_j \in S_G(i) \iff j \geq n - i$$

where  $i \in [0, 8]$  and  $n = 11$ . This set of reward functions is designed to spread the information that “going right is good” from  $s_{11}$  to  $s_3$ <sup>8</sup>.

### 5.2 Vacuum World

In the classic Vacuum World problem [17] (Fig. 1b), a robot needs to suck up dirt from two rooms. The robot can move to the left, move to the right, or suck up the dirt in the current room. We follow the example of [21] and give the agent a time limit of  $T = 10$  time steps. The size of the policy space for Vacuum World is  $|X| = 3^8 = 6561$ . Each solution in the space has 16 neighbours.

<sup>8</sup> Note that it is impossible to enter  $s_2$  from the left due to the time limit.

We study five different reward functions for Vacuum World. The *action penalty* and *goal based* reward functions are standard RL functions [8]. The *goal based* function gives the agent a reward of +1 for reaching the goal state, and 0 otherwise. The *action penalty* function penalises an agent for every time step, resulting in lower penalties for reaching the goal faster. We use these two reward functions on all remaining environments in this paper.

The other reward functions attempt to give information about progress towards the goal. The *suck dirt* function rewards the agent for sucking up a piece of dirt. The *leave clean room* function rewards the agent for leaving a room that has no dirt. The *suck dirt and action penalty* function is designed to add incentive to finish quickly to the *suck dirt* function (Table 1).

**Table 1.** The different reward functions that we use for Vacuum World.

Name	Definition
Action penalty	$r_{\text{ap}}^{\text{vw}}(s, a, s') = -1$
Goal based	$r_{\text{gb}}^{\text{vw}}(s, a, s') = \begin{cases} 1 & \text{both rooms clean} \\ 0 & \text{otherwise} \end{cases}$
Suck Dirt	$r_{\text{sd}}^{\text{vw}}(s, a, s') = \begin{cases} 1 & \text{dirt sucked} \\ 0 & \text{otherwise} \end{cases}$
Leave clean room	$r_{\text{lc}}^{\text{vw}}(s, a, s') = \begin{cases} 1 & \text{left clean room} \\ 0 & \text{otherwise} \end{cases}$
Suck dirt and action penalty	$r_{\text{sdap}}^{\text{vw}}(s, a, s') = r_{\text{sd}}^{\text{vw}}(s, a, s') + r_{\text{ap}}^{\text{vw}}(s, a, s')$

### 5.3 Wall Maze

Two dimensional mazes are common in RL [22], and have previously been used for feature analysis [21]. We study the maze in Fig. 1c. The agent starts in the bottom-left corner (state 1), and must reach the goal within a time limit of  $T = 7$  time steps. The agent can move: up, down, left, and right. Attempts to move out of bounds, or into walls, cause the agent to remain in place. The size of the policy space is  $|X| = 4^8 = 65536$ . Each solution in the space has 24 neighbours.

**Table 2.** The different reward functions that we use for the Wall Maze.

Name	Definition
Action penalty	$r_{\text{ap}}^{\text{wm}}(s, a, s') = -1$
Goal based	$r_{\text{gb}}^{\text{wm}}(s, a, s') = \begin{cases} 1 & s' = s_G \\ 0 & \text{otherwise} \end{cases}$
Manhattan distance	$r_{\text{md}}^{\text{wm}}(s, a, s') = 7 - ( s'.x - s_G.x  +  s'.y - s_G.y )$
Manhattan distance 2	$r_{\text{md2}}^{\text{wm}}(s, a, s') = r_{\text{md}}^{\text{wm}}(s, a, s') + 100 \times r_{\text{gb}}^{\text{wm}}(s, a, s')$

In addition to the *goal based* and *action-penalty* reward functions, we study two reward functions based on distance to the goal. The *Manhattan distance* function rewards the agent more for being closer to the goal. The value of 7 is the maximum possible distance from the goal, and keeps the output positive. The *Manhattan distance 2* function adds incentive for reaching the goal over staying in the maze accumulating rewards.

## 5.4 Hazard Maze

Hazard mazes provide obstacles that penalise the agent and terminate the episode when they are touched. We study a hazard maze with the same layout and time limit as the wall maze in Sect. 5.3 by replacing the wall tiles with hazard tiles.

Most of the reward functions for the Hazard Maze are the same as for the Wall Maze, but with a penalty added for entering a hazard. For notational convenience, we represent this using the *Hazard* reward function. The *Action Penalty 2* function penalises entering hazards more than other actions (Table 3).

**Table 3.** The different reward functions that we use for the Hazard Maze. The reward functions defined in this table reference those defined in Table 2.

Name	Definition
Hazard	$r_h^{\text{hm}}(s, a, s') = \begin{cases} -1 & s' \text{ is a hazard} \\ 0 & \text{otherwise} \end{cases}$
Action penalty	$r_{\text{ap}}^{\text{hm}}(s, a, s') = r_{\text{ap}}^{\text{wm}}(s, a, s') + r_h^{\text{hm}}(s, a, s')$
Goal based	$r_{\text{gb}}^{\text{hm}}(s, a, s') = r_{\text{gb}}^{\text{wm}}(s, a, s') + r_h^{\text{hm}}(s, a, s')$
Manhattan distance	$r_{\text{md}}^{\text{hm}}(s, a, s') = r_{\text{md}}^{\text{wm}}(s, a, s') + r_h^{\text{hm}}(s, a, s')$
Manhattan distance 2	$r_{\text{md}2}^{\text{hm}}(s, a, s') = r_{\text{md}2}^{\text{wm}}(s, a, s') + r_h^{\text{hm}}(s, a, s')$
Action penalty 2	$r_{\text{ap}2}^{\text{hm}}(s, a, s') = r_{\text{ap}}^{\text{wm}}(s, a, s') + 100 \times r_h^{\text{hm}}(s, a, s')$

## 6 Results

Tables 4, 5 and 6 show the scalar feature values for the different RL problems. As we are interested in the effects of reward shaping on features,  $F$ , we also report the maximum relative difference (as a percentage) over the reward functions for each feature<sup>9</sup>:  $\Delta(F) = 100 \times \frac{\max(\text{abs}(F)) - \min(\text{abs}(F))}{\min(\text{abs}(F))}$ . Figure 2 shows the fitness distributions.

<sup>9</sup> Some relative difference values cannot be calculated, due to a division by zero.

**Table 4.** Scalar feature results for reward shaping on the 1D Chain problem.

Feature	$r_0^{1D}$	$r_1^{1D}$	$r_2^{1D}$	$r_3^{1D}$	$r_4^{1D}$	$r_5^{1D}$	$r_6^{1D}$	$r_7^{1D}$	$r_8^{1D}$	$\Delta$
#Local maxima	1	1	1	1	1	1	1	1	1	0
#Local minima	1	1	1	1	1	1	1	1	1	0
Searchability	0.02	0.03	0.04	0.07	0.08	0.11	0.11	0.11	0.11	450
Neutrality	0.97	0.94	0.91	0.86	0.84	0.79	0.79	0.78	0.78	24
Ruggedness	0.13	0.20	0.29	0.39	0.45	0.53	0.53	0.55	0.54	323
Deception	-0.40	-0.47	-0.53	-0.58	-0.53	-0.48	-0.49	-0.51	-0.53	45
Dispersion	-0.15	-0.15	-0.15	-0.15	-0.16	-0.15	-0.15	-0.15	-0.15	7
Symmetry G	0.94	0.88	0.75	0.50	0.00	0.00	0.00	0.00	0.00	-
Symmetry GM	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-
Symmetry O	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-

**Table 5.** Scalar feature results for reward shaping on Vacuum World.

Feature	$r_{ap}^{vw}$	$r_{gb}^{vw}$	$r_{sd}^{vw}$	$r_{lc}^{vw}$	$r_{sdap}^{vw}$	$\Delta$
#Local maxima	1	1	1	1	1	0
#Local minima	1	1	1	1	1	0
Searchability	0.02	0.02	0.07	0.05	0.07	250
Neutrality	0.96	0.96	0.86	0.91	0.86	12
Ruggedness	0.16	0.17	0.41	0.31	0.42	163
Deception	-0.41	-0.52	-0.62	-0.49	-0.49	51
Dispersion	-0.30	-0.27	-0.28	-0.29	-0.30	11
Symmetry G	0.90	0.90	0.39	0.72	0.39	131
Symmetry GM	0.00	0.00	0.28	0.03	0.27	-
Symmetry O	0.00	0.00	0.02	0.00	0.00	-

**Table 6.** Scalar feature results for reward shaping on the Wall and Hazard Mazes.

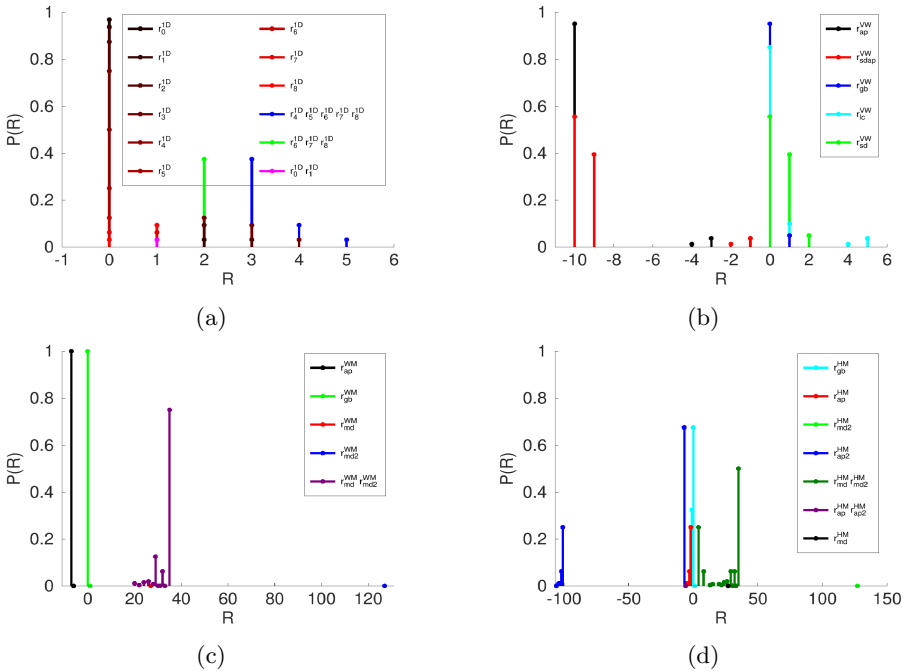
Feature	$r_{ap}^{wm}$	$r_{gb}^{wm}$	$r_{md}^{wm}$	$r_{md2}^{wm}$	$\Delta$	$r_{ap}^{hm}$	$r_{gb}^{hm}$	$r_{md}^{hm}$	$r_{md2}^{hm}$	$r_{ap2}^{hm}$	$\Delta$
#Local maxima	1	1	1	1	0	1	1	1	1	1	0
#Local minima	1	1	1	1	0	1	1	1	1	1	0
Searchability	0.0002	0.0002	0.05	0.05	34900	0.05	0.05	0.07	0.07	0.05	40
Neutrality	0.9996	0.9996	0.90	0.90	11	0.90	0.91	0.85	0.85	0.90	7
Ruggedness	0.0044	0.0017	0.31	0.32	18724	0.32	0.32	0.42	0.42	0.33	31
Deception	-0.0663	-0.0663	-0.88	+0.35	1227	-0.90	-0.02	-0.80	+0.02	-0.02	4350
Dispersion	-0.3145	-0.3153	-0.40	-0.30	27	-0.41	-0.31	-0.40	-0.30	-0.31	32
Symmetry G	0.9995	0.9995	0.58	0.58	72	0.50	0.54	0.31	0.31	0.50	74
Symmetry GM	0.0000	0.0000	0.58	0.58	-	0.18	0.66	0.33	0.33	0.50	267
Symmetry O	0.0000	0.0000	0.74	0.00	-	0.22	0.00	0.48	0.00	0.00	-

## 7 Discussion

### 7.1 Reward Shaping

We can see that fitness distributions, ruggedness, searchability, and symmetry all vary greatly as reward functions are changed. Deception varies greatly only in the Wall Maze and Hazard Maze environments. Neutrality, dispersion, and the number of local optima remain relatively invariant to changes in the reward function. This suggests that reward shaping could be beneficial if an algorithm is sensitive to ruggedness, searchability, symmetry, or deception. Reward shaping may provide little benefit if an algorithm is sensitive to neutrality or dispersion.

The consistently high neutrality is surprising, as some reward functions (e.g. for the 1D Chain) were designed to spread information and create a rich, easy to solve landscape. Similarly, the *Manhattan distance* reward functions ( $r_{md}^{wm}$ ,  $r_{md2}^{wm}$ ,  $r_{md}^{hm}$ ,  $r_{md2}^{hm}$ ) for the mazes were intended to create additional local optima where the optimal policy is for the agent to remain at the start. However, the *Manhattan distance* functions shifted the global optimum to the start because there was no incentive to finish the maze; and the *Manhattan distance 2* functions shifted it back to the goal-seeking policy. At no point did two local optima exist.



**Fig. 2.** Fitness distributions for: (a) the 1D chain; (b) vacuum world; (c) the wall maze; and (d) the hazard maze. Reward functions are plotted in different colours. Separate colours show when reward functions have overlapping values.

The disconnect between reward function intuition and optimisation landscape reality may explain why it is difficult to design a good reward function; both for reward shaping, and in general.

Due to the lack of a time penalty, the *goal based* and *suck dirt* reward functions for vacuum world have multiple optimal policies. However, they still encourage the general behaviour of cleaning both rooms within the time limit. On the other hand, the *leave clean room* reward function is a reward shaping accident resulting in undesired behaviour. The optimal policy for this reward function is to clean one room, and then move back and forth forever.

Finally, caution is required when using the *action penalty* reward function in an environment with multiple termination conditions. The *action penalty* reward function for the hazard maze encourages the agent to jump into the nearest hazard as quickly as possible.

## 7.2 General Observations

By combining the information from a few features, we can get a picture of the overall structure of these landscapes. A single local maximum means that there is only one hill to climb. High neutrality indicates that this hill is made up of a series of large plateaus and steep cliffs. A single local minimum tells us that these plateaus are completely flat - they are not pocketed with lots of little “pits”. The fitness distributions tell us that the size of a plateau is usually inversely proportional to its fitness, with the least optimal plateaus occupying the most space. Results for other features can be explained using this picture of the landscapes.

Searchability, while greatly affected by reward shaping in relative terms, is consistently low across the problems we studied. Intuitively, improving the fitness of a solution by local search is directly affected by the amount of neutrality.

Except for the  $r_{\text{md}2}^{\text{wm}}$  function, all problems we studied had low deception<sup>10</sup> and dispersion<sup>11</sup>; which is expected in problems with one local optimum.

Many of the problems we studied have high levels of symmetry for poor performing policies (Symmetry G), and little to no symmetry for good (Symmetry GM) or optimal policies (Symmetry O). The sub-optimal policies appear to occupy large flat sections of the search space, such that a symmetric operation on them would result in a policy from the same flat region. Meanwhile, optimal policies are rare; and symmetric operations cause them to “fall off the mountain” onto the lower fitness plateaus.

Ruggedness seems to vary by a large degree across both environments and reward functions. However, there is little we can say about ruggedness, as there does not seem to be a noticeable pattern to the variation.

The results of high neutrality and a single local optimum are consistent with previous work on RL landscapes [21]. The high neutrality is worth emphasising.

<sup>10</sup> A measure of how local search can navigate away from global optima.

<sup>11</sup> A measure of how spread out high fitness solutions are.



The lowest  $n_r^a$  value we observed was 0.78. In the problem with the **least** neutrality, 78% of the neighbours of the “average” solution were neutral neighbours.

We suspect a connection between the concepts of neutrality in optimisation landscapes and sparsity in RL. The problem of sparse rewards has gained a lot of attention in recent years [2, 4, 16]. Sparsity in RL refers to reward functions that provide very little useful information for training. This is similar to the concept of neutrality: very flat regions of equal fitness with no information about where to go. Additionally, while sparsity is regarded as one of the dominating factors of RL problems, it seems that neutrality is the dominating factor of RL landscapes.

## 8 Conclusion

We have studied how reward shaping can affect underlying RL optimisation landscapes using fitness landscape features. We found that fitness distributions, deception, ruggedness, searchability, and symmetry can all be affected greatly by reward shaping. If an algorithm is most sensitive to one of those features, then reward shaping may be helpful; meanwhile, if an algorithm is sensitive to neutrality or dispersion, then reward shaping may provide little benefit.

More generally, it seems that very high neutrality is the dominating feature of the problems we studied. This suggests that algorithms that explore spaces globally, rather than locally, may perform well on RL problems. This might help explain the success of evolutionary methods on RL problems [3, 18, 26]. We also suspect that the high neutrality of these landscapes is connected to the well known problem of reward sparsity in RL.

## References



1. Ahmed, Z., Le Roux, N., Norouzi, M., Schuurmans, D.: Understanding the impact of entropy on policy optimization. In: Proceedings of the 36th International Conference on Machine Learning (2019)
2. Chentanez, N., Barto, A.G., Singh, S.P.: Intrinsically motivated reinforcement learning. In: Advances in Neural Information Processing Systems 17, pp. 1281–1288. MIT Press (2005)
3. Chrabaszcz, P., Loshchilov, I., Hutter, F.: Back to basics: benchmarking canonical evolution strategies for playing atari. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 1419–1426. AAAI Press (2018)
4. Dosovitskiy, A., Koltun, V.: Learning to act by predicting the future. arXiv preprint [arXiv:1611.01779](https://arxiv.org/abs/1611.01779) (2016)
5. Horn, J., Goldberg, D.E.: Genetic algorithm difficulty and the modality of fitness landscapes. *Found. Genetic Algorithms* **3**, 243–269 (1995)
6. Ilyas, A., et al.: Are deep policy gradient algorithms truly policy gradient algorithms? arXiv preprint [arXiv:1811.02553](https://arxiv.org/abs/1811.02553) (2018)
7. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the 6th International Conference on Genetic Algorithms (1995)

8. Koenig, S., Simmons, R.G.: The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Mach. Learn.* **22**(1–3), 227–250 (1996)
9. Lu, G., Li, J., Yao, X.: Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms. In: Merz, P., Hao, J.-K. (eds.) *EvoCOP 2011*. LNCS, vol. 6622, pp. 108–117. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-20364-0\\_10](https://doi.org/10.1007/978-3-642-20364-0_10)
10. Lunacek, M., Whitley, D.: The dispersion metric and the CMA evolution strategy. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006*, pp. 477–484. Association for Computing Machinery, New York (2006)
11. Malan, K.M., Engelbrecht, A.P.: Quantifying ruggedness of continuous landscapes using entropy. In: *2009 IEEE Congress on Evolutionary Computation*, pp. 1440–1447 (2009)
12. Malan, K.M., Engelbrecht, A.P.: A survey of techniques for characterising fitness landscapes and some possible ways forward. *Inf. Sci.* **241**, 148–163 (2013)
13. Moriarty, D.E., Schultz, A.C., Grefenstette, J.J.: Evolutionary algorithms for reinforcement learning. *J. Artif. Intell. Res.* **11**, 241–276 (1999)
14. Ochoa, G., Tomassini, M., Vérel, S., Darabos, C.: A study of NK landscapes’ basins and local optima networks. In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO 2008*, pp. 555–562. Association for Computing Machinery, New York (2008)
15. Oller, D., Glasmachers, T., Cuccu, G.: Analyzing reinforcement learning benchmarks with random weight guessing. arXiv preprint [arXiv:2004.07707](https://arxiv.org/abs/2004.07707) (2020)
16. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: *Proceedings of the 34th International Conference on Machine Learning* (2017)
17. Russel, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, London (2013)
18. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint [arXiv:1703.03864](https://arxiv.org/abs/1703.03864) (2017)
19. Schmidhuber, J., Hochreiter, S., Bengio, Y.: Evaluating benchmark problems by random guessing. In: Kolen, J., Cremer, S. (eds.) *A Field Guide to Dynamical Recurrent Networks*, pp. 231–235 (2001)
20. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) (2017)
21. Stapelberg, B., Malan, K.M.: Global structure of policy search spaces for reinforcement learning. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019*, pp. 1773–1781. Association for Computing Machinery, New York (2019)
22. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*, vol. 1. MIT Press, Cambridge (1998)
23. Van Hoyweghen, C., Naudts, B.: Symmetry in the search space. In: *Proceedings of the 2000 Congress on Evolutionary Computation, CEC00* (Cat. No. 00TH8512), vol. 2, pp. 1072–1078 (2000)
24. Vanneschi, L., Pirola, Y., Collard, P., Tomassini, M., Verel, S., Mauri, G.: A quantitative study of neutrality in GP Boolean landscapes. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006*, pp. 895–902. Association for Computing Machinery, New York (2006)

25. Wiewiora, E.: Reward shaping. In: Sammut, C., Webb, G.I. (eds.) *Encyclopedia of Machine Learning*, pp. 863–865. Springer, Boston (2010). [https://doi.org/10.1007/978-0-387-30164-8\\_731](https://doi.org/10.1007/978-0-387-30164-8_731)
26. Wilson, D.G., Cussat-Blanc, S., Luga, H., Miller, J.F.: Evolving simple programs for playing atari games. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018*, pp. 229–236. Association for Computing Machinery, New York (2018)



# ClipUp: A Simple and Powerful Optimizer for Distribution-Based Policy Evolution

Nihat Engin Toklu<sup>1,2</sup>(✉), Pawel Liskowski<sup>1,2</sup>,  
and Rupesh Kumar Srivastava<sup>1,2</sup>

<sup>1</sup> NNAISENSE, Lugano, Switzerland

<sup>2</sup> NNAISENSE, Austin, USA

{engin,pawel,rupesh}@nnaisense.com

**Abstract.** Distribution-based search algorithms are a powerful approach for evolutionary reinforcement learning of neural network controllers. In these algorithms, gradients of the reward function with respect to the policy parameters are estimated using a population of solutions drawn from a search distribution, and then used for policy optimization with stochastic gradient ascent. A common choice is to use the Adam optimization algorithm for obtaining an adaptive behavior during gradient ascent, due to its success in a variety of supervised learning settings. As an alternative to Adam, we propose to enhance classical momentum-based gradient ascent with two simple-yet-effective techniques: gradient normalization and update clipping. We argue that the resulting optimizer called ClipUp (short for clipped updates) is a better choice for distribution-based policy evolution because its working principles are simple and easy to understand and its hyperparameters can be tuned more intuitively in practice. Moreover, it avoids the need to re-tune hyperparameters if the reward scale changes. Experiments show that ClipUp is competitive with Adam despite its simplicity and is effective at some of the most challenging continuous control benchmarks, including the Humanoid control task based on the Bullet physics simulator.

## 1 Introduction

We propose a simple and competitive optimizer (an adaptive gradient following mechanism) for use within distribution-based evolutionary search algorithms for training reinforcement learning (RL) agents. Distribution-based search [9, 10, 14, 18–21, 24] is a simple but powerful category of evolutionary algorithms. The common principles of distribution-based search algorithms can be summarized as follows:

**0:** Initialize the current solution.

**1:** Sample neighbor solutions from a search distribution centered at the current solution.

**2:** Evaluate each neighbor solution, estimate a gradient which is in the direction of the weighted average (in terms of solution fitnesses) of the neighbor solutions.

- 3:** Update the current solution using the gradient.
- 4:** Go to 1.

In step 3, it is possible to use any stochastic gradient ascent algorithm to potentially speed up policy optimization through the use of adaptive updates e.g. momentum [16] or Adam [12]. These adaptive optimizers are commonly used in the supervised deep learning community since they take into account not just the current gradient, but often its previous values as well to compute a more informed update. Following its success in supervised learning, the Adam optimizer in particular has been commonly used in recent work on neuroevolutionary RL [6, 8, 18].

Using adaptive optimizers instead of plain gradient ascent can potentially speed up training, but confronts the practitioner with new challenges. A basic question to consider is whether optimizers designed to work in the supervised learning setting (where gradients are obtained through differentiation) address the issues that arise when training RL agents with distribution-based search. Secondly, these optimizers often introduce several additional hyperparameters, all of which must be tuned in order to capitalize on their abilities even for supervised learning, as demonstrated recently by [2]. But many hyperparameters are non-intuitive to tune in our setting, and most practitioners instead tune one or two primary hyperparameters using a few trials while keeping the rest at their default values. This potentially leaves performance gains on the table.

We contribute a potential solution to these issues with a new general-purpose adaptive optimizer that is especially suitable for embedding into the framework of distribution-based search. It combines a few simple techniques: stochastic gradient ascent with heavy ball momentum, gradient normalization, and update clipping. We refer to it as *ClipUp*, short for “clipped updates”, and argue that it is a valuable tool for RL practitioners because its hyperparameters are very easy to understand, providing valuable intuitions for tuning them for a given problem. In a series of experiments using a representative distribution-based search algorithm, we compare it to Adam and show that (i) ClipUp is insensitive to reward function rescaling while Adam needs to be returned for each scale; and (ii) ClipUp performs on par with Adam on **Walker2d-v2** and **Humanoid-v2** robot control benchmarks based on the Mujoco simulator. Finally, we demonstrate that ClipUp can also solve the PyBullet [4] humanoid control task, a challenging RL environment which has been reported to be “much harder” [3] than its MuJoCo counterpart.

## 2 Background

### 2.1 Policy Gradients with Parameter-Based Exploration

All experiments in this study use *policy gradients with parameter-based exploration* (PGPE; [21]) as a representative distribution-based search algorithm. A variant of PGPE was also used in [7, 8], demonstrating that it can be successful on recent RL benchmarks. Although PGPE draws inspiration from an RL-focused study [25], it is a general-purpose derivative-free optimization algorithm, and can be considered to be a variant of evolutionary gradient search algorithms [19, 20].

The PGPE algorithm is described in Algorithm 1. Each iteration of PGPE works as follows. First (line 2), a new population is built by sampling neighbor solutions around the current solution  $x_k$ . These neighbors are sampled from a Gaussian distribution whose shape is expressed by the standard deviation vector  $\sigma_k$ . Like in [20], solutions are sampled symmetrically: when a new neighbor solution  $x_k + \delta$  is added to the population, its *mirror* counterpart  $x_k - \delta$  is added as well. We denote our population of directions sampled at iteration  $k$  as  $D_k$ . The next step (line 3) of the algorithm is to find the gradient for updating the current solution, which is computed by the weighted average of all the fitness gains along the directions. The algorithm then (line 4) computes the gradient for updating the standard deviation vector. Finally these gradients are used for computing the new current solution and the new standard deviation (line 5).

---

**Algorithm 1** The PGPE algorithm [21]

---

**Hyperparameters:** Population size  $\lambda$

Initial solution  $x_1$  (in our study, set as near zero)

Initial standard deviation vector  $\sigma_1$

Standard deviation learning rate  $\Omega$

AdaptiveOptimizer  $\in \{\text{Adam}, \text{ClipUp}\}$

- 1: **for** iteration  $k = 1, 2, \dots$  **do**
- 2: Build a population of directions

$$D_k \leftarrow \left\{ (d_i^+, d_i^-) \mid \begin{array}{l} d_i^+ = x_k + \delta_i^+, \\ d_i^- = x_k - \delta_i^-, \\ \delta_i \sim (\mathcal{N}(0, I) \cdot \sigma_k), \\ i \in \{1, 2, \dots, \lambda/2\} \end{array} \right\}$$

- 3: Estimate the gradient for updating the current solution  $x$

$$\nabla_{x_k} \leftarrow \sum_{(d^+, d^-) \in D_k} \left[ \frac{(d^+ - x_k) \cdot (f(d^+) - f(d^-))}{2 \cdot |D_k|} \right]$$

- 4: Estimate the gradient for updating the standard deviation vector  $\sigma$

$$\nabla_{\sigma_k} \leftarrow \sum_{(d^+, d^-) \in D_k} \left( \frac{f(d^+) + f(d^-)}{2} - b \right) \cdot \left( \frac{(d^+ - x_k)^2 - (\sigma_k)^2}{\sigma_k} \right) \cdot \frac{1}{|D_k|}$$

where  $b$  is the average fitness of all the solutions in  $D_k$

- 5: Perform the updates

$$x_{k+1} \leftarrow x_k + \text{AdaptiveOptimizer}(\nabla_{x_k})$$

$$\sigma_{k+1} \leftarrow \sigma_k + \Omega \cdot \nabla_{\sigma_k}$$

- 6: **end for**

---

Division between two vectors, and squaring of a vector are elementwise operations.

---

In [7,8], the authors enhanced PGPE in three ways: (i) solutions were fitness-ranked (from worst to best, the ranks range linearly from  $-0.5$  to  $0.5$ ) and their ranks were used for gradient computations instead of their raw fitnesses; (ii) the Adam optimizer was used for following the gradients in an adaptive manner; (iii) to make sure that the standard deviation updates remain stable, the updates for the standard deviation were clipped in each dimension to 20% of their original values. (i) and (ii) were also previously shown to be successful in the evolution strategy variant studied in [18]. We adopt these enhancements in this study and incorporate two further RL-specific enhancements listed below.

**Adaptive Population Size** [18]. When considering locomotion problems where the agent bodies are unstable, wrong actions cause the agents to fall, breaking constraints and ending the trajectories abruptly. In the beginning, most of the agents fall immediately. Therefore, to find reliable gradients at the beginning of the search, very large populations are required so that they can explore various behaviors. However, such huge populations might be unnecessary once the search finds a reliable path to follow. Therefore, in addition to the population size  $\lambda$ , we introduce a hyperparameter  $T$ , which is the total number of environment timesteps (i.e. number of interactions done with the simulator) that must be completed within an iteration. If, after evaluating all the solutions within the population, the total number of timesteps is below  $T$ , the trajectories are considered to be too short (most agents fell down) and the current population size is increased (by  $\lambda$  more solutions in our implementation) until the total number of environment timesteps reaches  $T$ , or the extended population size reaches an upper bound  $\lambda^{\max}$ . This mechanism results in an automatic decay of the population size during the evolution process.

**Observation Normalization** [14,18]. We normalize observations using the running statistics over all the observations received by all the agents until the current iteration.

In the remainder of this paper, we use the notation PGPE+ClipUp to refer to PGPE combined with ClipUp as the adaptive gradient following algorithm. Similarly, we use the notation PGPE+Adam for when Adam is used instead of ClipUp.

## 2.2 Heavy Ball Momentum

Proposed by [16], the heavy ball method is a very early momentum-based optimizer for speeding up the convergence. Considering the current solution as a ball moving in the solution space, each gradient contributes to the velocity of this ball. This means that the directions consistently pointed to by the recent gradients are followed more confidently (because the velocity accumulates towards those directions), and similarly, directions rarely pointed to are followed more cautiously (or they are not followed at all, instead, they just contribute negatively to the current velocity up to some extent).

When using distribution-based evolutionary search algorithms, the gradients can be very noisy because (i) they are estimated stochastically using a sampled population; and (ii) the objective function is a simulator which itself might be stochastic (e.g. because the simulator is a physics engine relying on stochastic heuristics, or it deliberately injects uncertainty to encourage more robust policies). The concept of momentum can be useful when dealing with noisy gradients, because, the velocity will accumulate towards the historically consistent components of the noisy gradients, and misleading inconsistent components of the gradients will cancel out.

Note that the Adam optimizer inherits the concept of momentum as well. Evolution strategy with covariance matrix adaptation (CMA-ES; [9,10]) also implements a variant of the momentum mechanism called “evolution path”.

### 2.3 Gradient Normalization

Used in [19] in the context of evolutionary search, gradient normalization has the useful effect of decoupling the direction of a gradient and its magnitude. The magnitude of the gradient can then be re-adjusted or overwritten by another mechanism, or simply by a constant.

When there is no gradient normalization, the magnitude of a gradient would be computed as a result of the weighted average performed over the fitness values of the population. The most important problem with unnormalized gradients is that one has to tune the step size according to the scales of the fitness values, which vary from problem to problem, or even from region to region within the solution space of the same RL problem. To counter the varying fitness scale issue, one can employ fitness ranking as done in prior work e.g. [8,10,18,24]. However, even then, the step size must be tuned according to the scale imposed by the chosen fitness ranking method.

On the other hand, let us now consider the simple mechanism of normalizing the gradient as  $\alpha \cdot (g / \|g\|)$ , where  $\alpha$  is the step size, and  $g$  is the unnormalized gradient. With this mechanism, the step size  $\alpha$  becomes a hyperparameter for tuning the Euclidean distance expressed by the normalized gradient, independent of the scale of the fitness values or ranks. In addition to the advantage of being scale independent, we argue that with this mechanism, it is easy to come up with sensible step size values for updating a policy.

### 2.4 Gradient Clipping

In the supervised learning community, gradient clipping [15,26] is a common practice for avoiding instabilities due to exploding gradients [11]. The technique used in this paper is related but slightly different. We clip the updated velocity of the heavy ball (just before updating the current solution), which is why we call it *update clipping*. It works as follows: if the Euclidean norm of the velocity is larger than a maximum speed threshold, then the velocity is clipped such that its magnitude is reduced to the threshold, but its direction remains unchanged. The intuition behind clipping the velocity of the heavy ball method is to prevent it from gaining very large velocities that can overshoot the (local) optimum point.



### 3 Formal Definition of the ClipUp Optimizer

We now explain the ClipUp optimizer, which can be seen as the combination of the heavy ball momentum, gradient normalization, and update clipping techniques discussed in Sect. 2.

Let us consider an optimization problem with the goal of maximizing  $f(x)$ , where  $x$  is a solution vector. We denote the gradient of  $f(x)$  as  $\nabla f(x)$ . In the context of evolutionary RL, it is usually the case that  $f(x)$  is not differentiable, therefore, it is estimated by using the fitness-weighted (or rank-weighted) average of the population of neighboring solutions.

In a setting without any adaptive optimizer, at iteration  $k$  with step size  $\alpha$ , the following simple update rule would be followed:

$$x_{k+1} \leftarrow x_k + \alpha \cdot \nabla f(x).$$

With ClipUp, the update rule becomes:

$$x_{k+1} \leftarrow x_k + \text{ClipUp}(\nabla f(x))$$

where ClipUp is defined in Algorithm 2. First (line 1), the algorithm normalizes the gradient, multiplies it by the step size  $\alpha$  (fixing the gradient's magnitude to  $\alpha$ ), and then computes a new velocity by adding the  $\alpha$ -sized gradient to the decayed velocity of the previous iteration (where decaying means that the previous velocity is multiplied by the momentum factor  $m$ , usually set as 0.9). The next step of the algorithm (lines 2 to 6) is to clip this newly computed velocity if its magnitude exceeds the threshold imposed by the hyperparameter  $\|v\|^{\max}$ . When clipped, the velocity's magnitude is reduced to  $\|v\|^{\max}$ , its direction remaining unchanged. Finally, the procedure ends by returning the clipped velocity (line 7).

With the normalization and the clipping operations employed by ClipUp, the two hyperparameters  $\alpha$  and  $\|v\|^{\max}$  gain intuitive meanings and become tunable directly in the scale of mutation one would like to apply on the current solution. The step size  $\alpha$  is now the fixed Euclidean norm of the vector that updates the velocity, and the maximum speed  $\|v\|^{\max}$  now expresses the maximum norm of the update to be done on the current solution. Moreover, they are completely independent of the fitness scale of the problem, or the fitness-based ranking employed on the population of solutions.

Like Adam, ClipUp does not make any assumptions about the search algorithm employing it. Therefore, although we use PGPE as our search algorithm, in theory it is possible to use ClipUp with other similar evolutionary algorithms as well, such as the evolution strategy variant used in [18].

### 4 Tuning Heuristics for PGPE+ClipUp

Having the step size  $\alpha$  and the maximum speed  $\|v\|^{\max}$  in the same scale allows us to come up with simple-yet-effective hyperparameter tuning rules. We have

observed that the simple rule  $\alpha = \|v\|^{\max}/2$  is transferable across the RL tasks we considered. We can also select the initial standard deviation vector  $\sigma_1$  once  $\|v\|^{\max}$  is known, by considering the radius of the initial spherical search distribution (see Fig. 1). When considering an initial radius  $r$ ,  $\sigma_1$  becomes a vector filled with:  $\sqrt{(r^2)/n}$ , where  $n$  is the dimensionality of the solution vector. We have observed that the setting  $r \approx 15 \cdot \|v\|^{\max}$  is transferable across several RL tasks. Depending on the hyperparameter tuning budget available, we suggest tuning this multiplier in the range [10, 20]. For this study, we used the multiplier 18 for our Humanoid-v2 experiments, and 15 for all our other experiments. Although it is difficult to claim that a certain set of hyperparameters is the best for all RL tasks in general, it is advantageous to have sensible default values, as they reduce the grid sizes to consider during initial hyperparameter tuning procedures in practice. The default settings above are visualized in Fig. 1.

---

**Algorithm 2** The ClipUp optimizer
 

---

**Initialization:** Velocity  $v_1 = 0$

**Hyperparameters:** Step size  $\alpha$   
 Maximum speed  $\|v\|^{\max}$   
 Momentum  $m$

**Input:** Estimated gradient  $\nabla f(x_k)$

```

1:  $v'_{k+1} \leftarrow m \cdot v_k + \alpha \cdot (\nabla f(x_k) / \|\nabla f(x_k)\|)$ 
2: if  $\|v'_{k+1}\| > \|v\|^{\max}$  then
3:    $v_{k+1} \leftarrow \|v\|^{\max} \cdot (v'_{k+1} / \|v'_{k+1}\|)$ 
4: else
5:    $v_{k+1} \leftarrow v'_{k+1}$ 
6: end if
7: return  $v_{k+1}$ 

```

---

## 5 Experiments

In this section, we present the results we obtained with PGPE+ClipUp, and also the comparisons made against PGPE+Adam. For ClipUp, we apply the tuning heuristics we proposed in Sect. 4:  $\alpha = \|v\|^{\max}/2$ ,  $r = 15 \cdot \|v\|^{\max}$  (18 instead of 15 in the case of Humanoid-v2). The momentum coefficient  $m$  was fixed at 0.9. For Adam, the following hyperparameter values from the original paper [12] were adopted:  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 1e-8$  (the same default values were used by [18]). A single episode of interaction with the environment was used to compute  $f$  during training. For testing the current solution at any point, the average return over 16 episodes was recorded for reporting in tables and plots. When a comparison is made between PGPE+ClipUp and PGPE+Adam, we use our best known search distribution radius for both, and tune the step size of Adam for each RL environment.

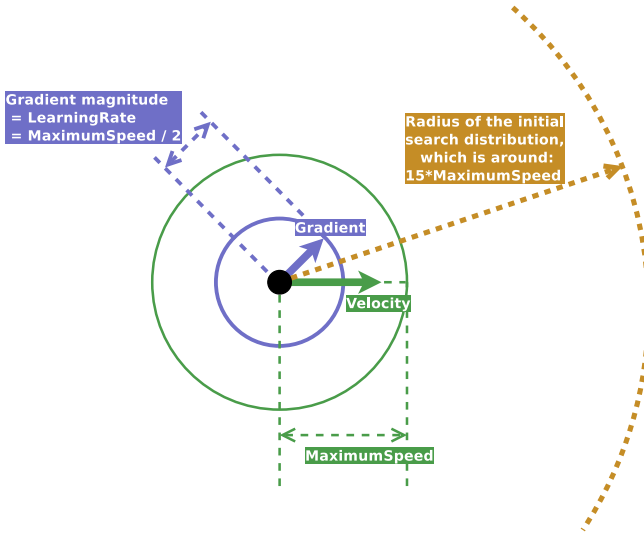


Fig. 1. Visualization of the default hyperparameters for ClipUp.

### 5.1 Fitness Scale (in)sensitivity

We argued that the most important factor contributing to the intuitiveness of ClipUp is that its step size is configured directly in terms of the mutation magnitude. The normalization operator employed within ClipUp ensures that this step size configuration is not affected by the fitness scale. To support this argument, we now compare the behaviors of PGPE+ClipUp and PGPE+Adam on the RL environment LunarLanderContinuous-v2 using multiple fitness scales. The environment has 8-dimensional observations and 2-dimensional actions in the range  $[-1, 1]$ . We compare four setups, leading to four different fitness scales for the same task: (i) 0-centered fitness ranking; (ii) raw (original) reward values; (iii) raw reward values multiplied by 1000; and (iv) raw reward values divided by 1000.

PGPE was executed for 50 iterations with a fixed population size of 200. The standard deviation learning rate was set as  $\Omega = 0.1$ . A linear policy was used, and the radius of the search distribution in the parameter space was set to 4.5. Following the tuning heuristics in Sect. 4, this means that the maximum speed for ClipUp is 0.3. For Adam, we tuned the step size in  $\{0.1, 0.125, 0.15, 0.175, 0.2\}$ . For step sizes 0.1 and 0.125, Adam’s performance dropped overall. The remaining step sizes did not clearly dominate each other, and therefore are discussed here.

For each reward scale, each algorithm, and each step size for Adam, we ran 10 experiment runs. The overall score of each group of 10 runs was recorded as the average of their final testing scores. These results are shown in Table 1.

It can be seen from the table that ClipUp is not affected at all by various reward scales. The small amount of deviation observed for ClipUp can be attributed to random noise. With Adam, different step size settings seemed to

**Table 1.** Behavior of PGPE+ClipUp and PGPE+Adam across four reward scales on LunarLanderContinuous-v2. The numbers outside the parentheses represent the final score, averaged across 10 runs. The numbers inside parentheses represent how much (as percentage) the score deviates from the same method’s result with fitness ranking.

	ClipUp stepsize = 0.15	Adam stepsize = 0.15
Fitness ranking	269.95 (100.00%)	255.32 (100.00%)
Raw rewards	270.15 (100.07%)	197.93 (77.52%)
Rewards $\times$ 1000	262.98 (97.42%)	139.61 (54.68%)
Rewards/1000	263.06 (97.44%)	200.34 (78.46%)
	Adam stepsize = 0.175	Adam stepsize = 0.2
Fitness ranking	241.94 (100.00%)	263.73 (100.00%)
Raw rewards	235.72 (97.43%)	199.25 (75.55%)
Rewards $\times$ 1000	211.84 (87.56%)	245.67 (93.15%)
Rewards/1000	187.25 (77.39%)	111.76 (42.38%)

introduce different sensitivities to the reward scale. The most stable setting for step size was 0.175. With step size 0.15, its performance dropped significantly when the rewards were multiplied by 1000. On the other hand, with step size 0.2, the performance dropped when the rewards were divided by 1000. Overall, we conclude that the performance of ClipUp was consistent across fitness scales while that of Adam was not.

## 5.2 MuJoCo Continuous Control Tasks

Next we consider the continuous control tasks `Walker2d-v2` and `Humanoid-v2` defined in the Gym [1] library, simulated using the MuJoCo [23] physics engine. The goal in these tasks is to make a robot walk forward. In `Walker2d-v2`, the robot has a two-legged simplistic skeleton based on [5]. In `Humanoid-v2`, originally from [22], the robot has a much more complex humanoid skeleton.

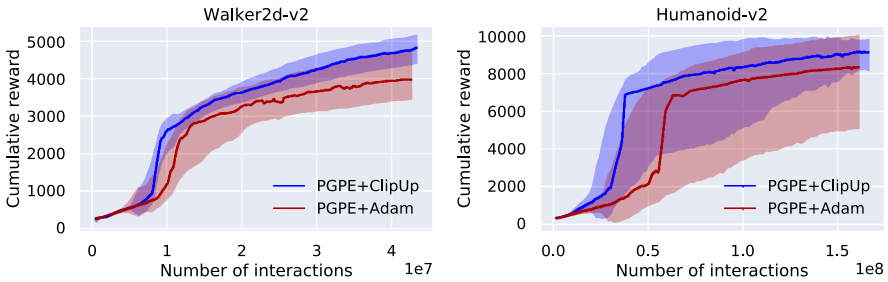
Previous studies [14, 17] have demonstrated that a linear policy is sufficient to solve these tasks, and therefore we also adopt this approach. The policy has the form  $\text{action} = \text{observation} \cdot M + b$ , where  $M$  is a matrix, and  $b$  is a bias vector which has the same length with the action vector. In total, this results in 108 optimization variables for `Walker2d-v2`, and 6409 optimization variables for `Humanoid-v2`.

In these RL environments, by default the agents are rewarded a certain amount of “alive bonus” at each simulator timestep for not falling. Mania et al. [14] reported that this alive bonus causes the optimization to be driven towards agents that stand still to collect the bonus and do not learning to walk. We experienced the same issue in our experiments, and therefore, following [14], removed these alive bonuses.

Both tasks were solved using both PGPE+Adam and PGPE+ClipUp, each with 30 runs. For ClipUp, we set the maximum speed as  $1.5e-2$  for both tasks. When using Adam, we used the same radius values we used with ClipUp, and then searched for suitable Adam step sizes. With *Walker2d-v2*, we considered the step size set  $\{4e-4, 5e-4, \dots, 9e-4, 1.2e-3, 1.3e-3, \dots, 1.8e-3, 2e-3, 3e-3, \dots, 6e-3\}$ , ran PGPE+Adam 10 times for each, and then found  $4e-3$  to perform the best. With *Humanoid-v2*, we considered the step size set  $\{4e-4, 5e-4, \dots, 9e-4\}$ , ran PGPE+Adam 10 times for each, and then found  $6e-4$  to perform the best. This basic tuning setup reflects a few tuning trials a programmer may typically use in practice to ascertain the performance of an algorithm on an RL task.

Among the shared PGPE hyperparameters for *Walker2d-v2*, we set  $\lambda = 100$ ,  $\lambda^{\max} = 800$ , and also declared that a population must complete  $T = 75000$  interactions with the simulator. The value 75 000 comes from  $100 \cdot 1000 \cdot (3/4)$ , that is, more or less the 3/4 of the solutions in a population must complete their 1000-step episodes to the end, otherwise the size of that population is increased. For *Humanoid-v2*, we set  $\lambda = 200$ ,  $\lambda^{\max} = 3200$ , and  $T = 150000$  (150 000 being  $200 \cdot 1000 \cdot (3/4)$ ). The standard deviation learning rate  $\Omega$  was fixed to 0.1.

Results obtained with ClipUp and Adam are compared in Fig. 2. In both cases, the eventual performance of the two algorithms was very similar, but, according to the reported medians, ClipUp jumped to high cumulative rewards earlier for *Humanoid-v2*. Both algorithms scored over 6000 on *Humanoid-v2*, clearing the official solving threshold.



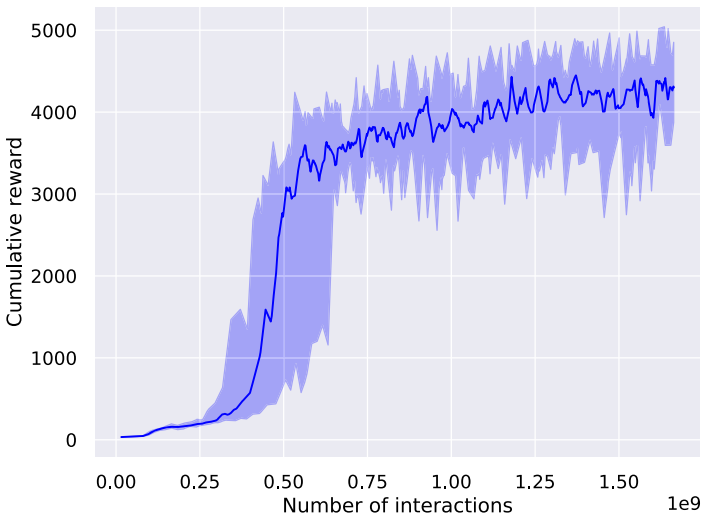
**Fig. 2.** Performances of PGPE+ClipUp and PGPE+Adam on *Walker2d-v2* and *Humanoid-v2*. Both PGPE+ClipUp and PGPE+Adam were run 30 times on each task. Each run’s reported cumulative reward at a time is the result of 16 re-evaluations averaged. The x-axis represents the number of interactions made with the simulator (i.e. the number of simulator timesteps). Dark lines mark the median cumulative reward values. The shaded regions are bounded by the mean  $\pm$  standard deviations of the cumulative rewards.

### 5.3 PyBullet Humanoid

As a final stress test of ClipUp’s utility, we attempted to use ClipUp and our proposed hyperparameter tuning scheme for reliably solving the challenging RL task labeled `HumanoidBulletEnv-v0`, defined in and simulated by the PyBullet [4] library. This task also involves teaching a humanoid skeleton to walk forward. However, as noted by the author of PyBullet [3], this version of the task is much harder than its MuJoCo counterpart. Perhaps because of this mentioned difficulty, successful results for it are rarely reported.

As in MuJoCo experiments, the default alive bonus for this task was removed. In addition, trajectory length upper bound was decreased from 1000 timesteps to 200 timesteps, since the hardest part of the task is starting a forward gait (the terrain is flat and there are no randomized traps). A neural network policy was used with a single hidden layer of 64 neurons, resulting in 3985 optimization variables.

For PGPE+ClipUp, we set  $\lambda = 10000$ ,  $\lambda^{\max} = 80000$ ,  $T = 1500000$  (computed as  $10000 \cdot 200 \cdot (3/4)$ ),  $\Omega = 0.1$ , and  $\|v\|^{\max} = 0.15$ . Each run was on an Amazon EC2 m4.16xlarge instance (64 vCPUs). The performance of PGPE+ClipUp vs number of environment interactions is shown in Fig. 3. It can be seen that the median curve stayed mostly above 3500 (which is the solving threshold defined in [13]) after about  $0.75e9$  steps (about 15 h of training), and then mostly above 4000 after  $1e9$  steps (about 24 h). This result confirms that despite its simplicity, ClipUp is effective at solving hard control problems.



**Fig. 3.** Performance of PGPE+ClipUp on `HumanoidBulletEnv-v0` over 10 runs. Each run’s reported cumulative reward is the result of 16 re-evaluations averaged. The dark line marks the median cumulative reward values. The shaded region is bounded by the minimum and the maximum cumulative rewards.

## 6 Conclusions

Targeting the field of distribution-based evolutionary RL, we proposed ClipUp, a simple yet powerful optimizer that combines clipping and normalization techniques for stabilizing gradient-based search. We have argued that tuning ClipUp is intuitive, mainly thanks to the following:

- the step size and the maximum speed of ClipUp (which are the two main hyperparameters affecting the step size) are configured directly in the scale of the magnitude (norm) of mutation one would like to apply on the current solution;
- the step size configuration of ClipUp is not affected by the fitness scale of the optimization problem at hand;
- one can only tune the maximum speed of ClipUp, and decide on the step size and the initial search distribution’s radius by following simple heuristic rules (e.g. step size as half the maximum speed, and radius about 15 to 18 times the maximum speed).

These properties can save practitioners valuable time and effort when applying distribution-based search to RL problems. Moreover, it was found to be competitive against the well-known Adam optimizer on the MuJoCo continuous control tasks `Walker2d-v2` and `Humanoid-v2`. Finally, we showed that PGPE with ClipUp can successfully solve the `HumanoidBulletEnv-v0` benchmark, demonstrating its applicability to highly challenging control tasks.

Although we used PGPE in our experiments, ClipUp can take the optimizer role in any evolution strategy variant where the solution update is in the form of gradient estimation (e.g. [18]). With this broad applicability, we hope ClipUp to be a valuable tool for neuroevolutionary RL.

## References

1. Brockman, G., et al.: OpenAI Gym. arXiv preprint [arXiv:1606.01540](https://arxiv.org/abs/1606.01540) (2016)
2. Choi, D., Shallue, C.J., Nado, Z., Lee, J., Maddison, C.J., Dahl, G.E.: On empirical comparisons of optimizers for deep learning. arXiv preprint [arXiv:1910.05446](https://arxiv.org/abs/1910.05446) (2019)
3. Coumans, E.: Pybullet repository - issues. <https://github.com/bulletphysics/bullet3/issues/1718#issuecomment-393198883> (2018)
4. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. <http://pybullet.org> (2016-2019)
5. Erez, T., Tassa, Y., Todorov, E.: Infinite-horizon model predictive control for periodic tasks with contacts. In: Durrant-Whyte, H.F., Roy, N., Abbeel, P. (eds.) *Robotics: Science and Systems VII*, University of Southern California, Los Angeles, CA, USA, 27–30 June 2011 (2011). <https://doi.org/10.15607/RSS.2011.VII.010>. <http://www.roboticsproceedings.org/rss07/p10.html>
6. Freeman, D., Ha, D., Metz, L.: Learning to predict without looking ahead: World models without forward prediction. In: *Advances in Neural Information Processing Systems*, pp. 5380–5391 (2019)

7. Ha, D.: A visual guide to evolution strategies. *blog.otoro.net* (2017). <http://blog.otoro.net/2017/10/29/visual-evolution-strategies/>
8. Ha, D.: Reinforcement learning for improving agent design. *Artificial Life* **25**(4), 352–365 (2019). <https://doi.org/10.1162/artl.a.00301>. PMID: 31697584
9. Hansen, N., Ostermeier, A.: Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The  $(\mu/\mu_I, \lambda)$ -cma-es. *Eufit* **97**, 650–654 (1997)
10. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* **9**(2), 159–195 (2001)
11. Hochreiter, S.: Untersuchungen zu dynamischen neuronalen Netzen. Masters thesis, Technische Universität München, München (1991)
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: *Proceedings of 3rd International Conference on Learning Representations* (2015)
13. Klimov, O., Schulman, J.: Roboschool. *OpenAI blog* (2017). <https://openai.com/blog/roboschool/>
14. Mania, H., Guy, A., Recht, B.: Simple random search of static linear policies is competitive for reinforcement learning. In: *Advances in Neural Information Processing Systems*, pp. 1800–1809 (2018)
15. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: *International conference on machine learning*, pp. 1310–1318 (2013)
16. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **4**(5), 1–17 (1964)
17. Rajeswaran, A., Lowrey, K., Todorov, E.V., Kakade, S.M.: Towards generalization and simplicity in continuous control. In: *Advances in Neural Information Processing Systems*, pp. 6550–6561 (2017)
18. Salimans, T., Ho, J., Chen, X., Sidor, S., Sutskever, I.: Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017)
19. Salomon, R.: Evolutionary algorithms and gradient search: similarities and differences. *IEEE Trans. Evol. Comput.* **2**(2), 45–55 (1998)
20. Salomon, R.: Inverse mutations: making the evolutionary-gradient-search procedure noise robust. In: *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, pp. 322–327 (2005)
21. Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., Schmidhuber, J.: Parameter-exploring policy gradients. *Neural Netw.* **23**(4), 551–559 (2010)
22. Tassa, Y., Erez, T., Todorov, E.: Synthesis and stabilization of complex behaviors through online trajectory optimization. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913. *IEEE* (2012)
23. Todorov, E., Erez, T., Tassa, Y.: MuJoCo: A physics engine for model-based control. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. *IEEE* (2012)
24. Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., Schmidhuber, J.: Natural evolution strategies. *J. Mach. Learn. Res.* **15**(1), 949–980 (2014)
25. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**(3–4), 229–256 (1992). <https://doi.org/10.1007/BF00992696>
26. Zhang, J., He, T., Sra, S., Jadbabaie, A.: Why gradient clipping accelerates training: a theoretical justification for adaptivity. In: *International Conference on Learning Representations* (2020)





# Warm-Start AlphaZero Self-play Search Enhancements

Hui Wang<sup>(✉)</sup>, Mike Preuss, and Aske Plaat

Leiden Institute of Advanced Computer Science, Leiden University,  
Leiden, The Netherlands

[h.wang.13@liacs.leidenuniv.nl](mailto:h.wang.13@liacs.leidenuniv.nl)

<http://www.cs.leiden.edu>

**Abstract.** Recently, AlphaZero has achieved landmark results in deep reinforcement learning, by providing a single self-play architecture that learned three different games at super human level. AlphaZero is a large and complicated system with many parameters, and success requires much compute power and fine-tuning. Reproducing results in other games is a challenge, and many researchers are looking for ways to improve results while reducing computational demands. AlphaZero’s design is purely based on self-play and makes no use of labeled expert data or domain specific enhancements; it is designed to learn from scratch. We propose a novel approach to deal with this cold-start problem by employing simple search enhancements at the beginning phase of self-play training, namely Rollout, Rapid Action Value Estimate (RAVE) and dynamically weighted combinations of these with the neural network, and Rolling Horizon Evolutionary Algorithms (RHEA). Our experiments indicate that most of these enhancements improve the performance of their baseline player in three different (small) board games, with especially RAVE based variants playing strongly.

**Keywords:** Reinforcement learning · MCTS · Warm-start enhancements · RHEA · AlphaZero-like self-play

## 1 Introduction

The AlphaGo series of programs [1–3] achieve impressive super human level performance in board games. Subsequently, there is much interest among deep reinforcement learning researchers in self-play, and self-play is applied to many applications [4, 5]. In self-play, *Monte Carlo Tree Search* (MCTS) [6] is used to train a deep neural network, that is then employed in tree searches, in which MCTS uses the network that it helped train in previous iterations.

On the one hand, self-play is utilized to generate game playing records and assign game rewards for each training example automatically. Thereafter, these examples are fed to the neural network for improving the model. No database of labeled examples is used. Self-play learns tabula rasa, from scratch. However, self-play suffers from a cold-start problem, and may also easily suffer from bias

since only a very small part of the search space is used for training, and training samples in reinforcement learning are heavily correlated [2, 7].

On the other hand, the MCTS search enhances performance of the trained model by providing improved training examples. There has been much research into enhancements to improve MCTS [6, 8], but to the best of our knowledge, few of these are used in Alphazero-like self-play, which we find surprising, given the large computational demands of self-play and the cold-start and bias problems.

This may be because AlphaZero-like self-play is still young. Another reason could be that the original AlphaGo paper [1] remarks about AMAF and RAVE [9], two of the best known MCTS enhancements, that “AlphaGo does not employ the *all-moves-as-first* (AMAF) or *rapid action value estimation* (RAVE) heuristics used in the majority of Monte Carlo Go programs; when using policy networks as prior knowledge, these biased heuristics do not appear to give any additional benefit”. Our experiments indicate otherwise, and we believe there is merit in exploring warm-start MCTS in an AlphaZero-like self-play setting.

We agree that when the policy network is well trained, then heuristics may not provide significant added benefit. However, when this policy network has not been well trained, especially at the beginning of the training, the neural network provides approximately random values for MCTS, which can lead to bad performance or biased training. The MCTS enhancements or specialized evolutionary algorithms such as *Rolling Horizon Evolutionary Algorithms* (RHEA) may benefit the searcher by compensating the weakness of the early neural network, providing better training examples at the start of iterative training for self-play, and quicker learning. Therefore, in this work, we first test the possibility of MCTS enhancements and RHEA for improving self-play, and then choose MCTS enhancements to do full scale experiments, the results show that MCTS with warm-start enhancements in the start period of AlphaZero-like self-play improve iterative training with tests on 3 different regular board games, using an AlphaZero re-implementation [10].

Our main contributions can be summarized as follows:

1. We test MCTS enhancements and RHEA, and then choose warm-start enhancements (Rollout, RAVE and their combinations) to improve MCTS in the start phase of iterative training to enhance AlphaZero-like self-play. Experimental results show that in all 3 tested games, the enhancements can achieve significantly higher Elo ratings, indicating that warm-start enhancements can improve AlphaZero-like self-play.
2. In our experiments, a weighted combination of Rollout and RAVE with a value from the neural network always achieves better performance, suggesting also for how many iterations to enable the warm-start enhancement.

The paper is structured as follows. After giving an overview of the most relevant literature in Sect. 2, we describe the test games in Sect. 3. Thereafter, we describe the AlphaZero-like self-play algorithm in Sect. 4. Before the full length experiments in Sect. 6, an orientation experiment is performed in Sect. 5. Finally, we conclude our paper and discuss future work.

## 2 Related Work

Since MCTS was created [11], many variants have been studied [6, 12], especially in games [13]. In addition, enhancements such as RAVE and AMAF have been created to improve MCTS [9, 14]. Specifically, [14] can be regarded as one of the early prologues of the AlphaGo series, in the sense that it combines online search (MCTS with enhancements like RAVE) and offline knowledge (table based model) in playing small board Go.

In self-play, the large number of parameters in the deep network as well as the large number of hyper-parameters (see Table 2) are a black-box that precludes understanding. The high decision accuracy of deep learning, however, is undeniable [15], as the results in Go (and many other applications) have shown [16]. After AlphaGo Zero [2], which uses an MCTS searcher for training a neural network model in a self-play loop, the role of self-play has become more and more important. The neural network has two heads: a policy head and a value head, aimed at learning the best next move, and the assessment of the current board state, respectively.

Earlier works on self-play in reinforcement learning are [17–21]. An overview is provided in [8]. For instance, [17, 19] compared self-play and using an expert to play backgammon with temporal difference learning. [21] studied co-evolution versus self-play temporal difference learning for acquiring position evaluation in small board Go. All these works suggest promising results for self-play.

More recently, [22] assessed the potential of classical Q-learning by introducing Monte Carlo Search enhancement to improve training examples efficiency. [23] uses domain-specific features and optimizations, but still starts from random initialization and makes no use of outside strategic knowledge or preexisting data, that can accelerate the AlphaZero-like self-play.

However, to the best of our knowledge there is no further study on applying MCTS enhancements in AlphaZero-like self-play despite the existence of many practical and powerful enhancements.

## 3 Tested Games

In our experiments, we use the games Othello [24], Connect Four [25] and Gobang [26] with  $6 \times 6$  board size. All of these are two-player games. In Othello, any opponent’s color pieces that are in a straight line and bounded by the piece just placed and another piece of the current player’s are flipped to the current player’s color. While there is no legal move (the board is full), the player who has less pieces loses the game. Figure 1(a) shows the initial state of Othello. For Connect Four, players take turns dropping their own pieces from the top into a vertically suspended grid. The pieces fall down straightly and occupy the lowest position within the column. The player who first connects a line of four pieces horizontally, vertically, or diagonally wins the game. Figure 1(b) is a game termination example for  $6 \times 6$  Connect Four where the red player wins the game. As another connection game, Gobang is traditionally played on a Go board. Players

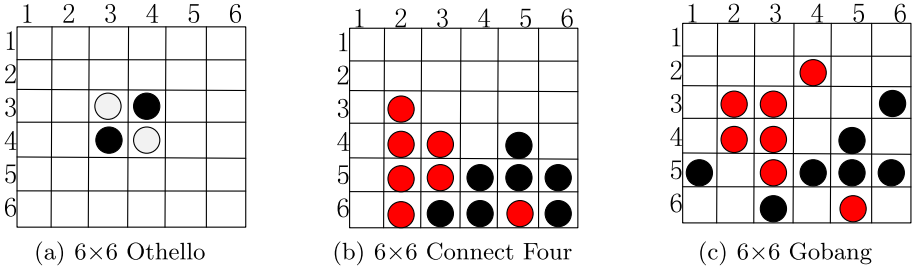


Fig. 1. Starting position for Othello, example positions for Connect Four and Gobang

also alternate turns, placing a stone of their own color on an empty position. The winner is the first player to connect an unbroken horizontal, vertical, or diagonal chain of 4 stones. Figure 1(c) is a termination example for 6 × 6 Gobang where the black player wins the game with 4 stones in a line.

A lot of methods on implementing game-playing programs to play these three games were studied. For instance, Buro used logistic regression to create Logistello [27] to play Othello. In addition, Chong et al. described the evolution of neural networks to play Othello with learning [28]. Thill et al. employed temporal difference learning to play Connect Four [29]. Zhang et al. studied evaluation functions for Gobang [30]. Moreover, Banerjee et al. tested transfer learning in General Game Playing on smaller games including 4 × 4 Othello [31]. Wang et al. assessed the potential of classical Q-learning based on small games including 4 × 4 Connect Four [32]. Varying the board size allows us to reduce or increase the computational complexity of these games. In our experiments, we use AlphaZero-like learning [33].

## 4 AlphaZero-Like Self-play Algorithms

### 4.1 The Algorithm Framework

According to [3, 33], the basic structure of AlphaZero-like self-play is an iterative process over three different stages (see Algorithm 1).

The first stage is a **self-play** tournament. The player plays several games against itself to generate game playing records as training examples. In each step of a game episode, the player runs MCTS (or one of the MCTS enhancements before  $I'$  iteration) to obtain, for each move, an enhanced policy  $\pi$  based on the probability  $\mathbf{p}$  provided by the policy network  $f_\theta$ . The hyper-parameters, and the abbreviation that we use in this paper is given in Table 2. In MCTS, hyper-parameter  $C_p$  is used to balance exploration and exploitation of the tree search, and we abbreviate it to  $c$ . Hyper-parameter  $m$  is the number of times to search down from the root for building the game tree, where the value ( $v$ ) of the states is provided by  $f_\theta$ . In (self-)play game episode, from  $T'$  steps on,

**Algorithm 1.** AlphaZero-like Self-play Algorithm

---

```

1: function ALPHAZEROGENERALWITHENHANCEMENTS
2:   Initialize  $f_\theta$  with random weights; Initialize retrain buffer  $D$  with capacity  $N$ 
3:   for iteration= $1, \dots, I', \dots, I$  do ▷ play curriculum of  $I$  tournaments
4:     for episode= $1, \dots, E$  do ▷ stage 1, play tournament of  $E$  games
5:       for  $t=1, \dots, T', \dots, T$  do ▷ play game of  $T$  moves
6:          $\pi_t \leftarrow$  MCTS Enhancement before  $I'$  or MCTS after  $I'$  iteration
7:          $a_t =$  randomly select on  $\pi_t$  before  $T'$  or  $\arg \max_a(\pi_t)$  after  $T'$  step
8:         executeAction( $s_t, a_t$ )
9:         Store every  $(s_t, \pi_t, z_t)$  with game outcome  $z_t$  ( $t \in [1, T]$ ) in  $D$ 
10:      Randomly sample minibatch of examples  $(s_j, \pi_j, z_j)$  from  $D$  ▷ stage 2
11:      Train  $f_{\theta'}$   $\leftarrow$   $f_\theta$ 
12:       $f_\theta = f_{\theta'}$  if  $f_{\theta'}$  is better than  $f_\theta$  using MCTS mini-tournament ▷ stage 3
13:   return  $f_\theta$ ;

```

---

the player always chooses the best action based on  $\pi$ . Before that, the player always chooses a random move according to the probability distribution of  $\pi$  to obtain more diverse training examples. After game ends, the new examples are normalized as a form of  $(s_t, \pi_t, z_t)$  and stored in  $D$ .

The second stage consists of **neural network training**, using data from stage 1. Several epochs are usually employed for the training. In each epoch ( $ep$ ), training examples are randomly selected as several small batches [34] based on the specific batch size ( $bs$ ). The neural network is trained with a learning rate ( $lr$ ) and dropout ( $d$ ) by minimizing [35] the value of the *loss function* which is the sum of the mean-squared error between predicted outcome and real outcome and the cross-entropy losses between  $\mathbf{p}$  and  $\pi$ . Dropout is a probability to randomly ignore some nodes of the hidden layer to avoid overfitting [36].

The last stage is the **arena comparison**, where a competition between the newly trained neural network model ( $f'_{\theta}$ ) and the previous neural network model ( $f_{\theta}$ ) is run. The winner is adopted for the next iteration. In order to achieve this, the competition runs  $n$  rounds of the game. If  $f_{\theta'}$  wins more than a fraction of  $u$  games, it is accepted to replace the previous best  $f_{\theta}$ . Otherwise,  $f_{\theta'}$  is rejected and  $f_{\theta}$  is kept as current best model. Compared with AlphaGo Zero, AlphaZero does not employ this stage anymore. However, we keep it to make sure that we can safely recognize improvements.

## 4.2 MCTS

In self-play, MCTS is used to generate high quality examples for training the neural network. A recursive MCTS pseudo code is given in Algorithm 2. For each search, the value from the value head of the neural network is returned (or the game termination reward, if the game terminates). During the search, for each visit of a non-leaf node, the action with the highest P-UCT value is selected to investigate next [2, 37]. After the search, the average win rate value  $Q(s, a)$  and visit count  $N(s, a)$  in the followed trajectory are updated correspondingly.

**Algorithm 2.** Neural Network Based MCTS

---

```

1: function MCTS( $s, f_\theta$ )
2:   Search( $s$ )
3:    $\pi_s \leftarrow \text{normalize}(Q(s, \cdot))$ 
4:   return  $\pi_s$ 
5: function SEARCH( $s$ )
6:   Return game end result if  $s$  is a terminal state
7:   if  $s$  is not in the Tree then
8:     Add  $s$  to the Tree, initialize  $Q(s, \cdot)$  and  $N(s, \cdot)$  to 0
9:     Get  $P(s, \cdot)$  and  $v(s)$  by looking up  $f_\theta(s)$ 
10:    return  $v(s)$ 
11:  else
12:    Select an action  $a$  with highest UCT value
13:     $s' \leftarrow \text{getNextState}(s, a)$ 
14:     $v \leftarrow \text{Search}(s')$ 
15:     $Q(s, a) \leftarrow \frac{N(s,a)*Q(s,a)+v}{N(s,a)+1}$ 
16:     $N(s, a) \leftarrow N(s, a) + 1$ 
17:  return  $v$ ;

```

---

The P-UCT formula that is used is as follows (with  $c$  as constant weight that balances exploitation and exploration):

$$U(s, a) = Q(s, a) + c * P(s, a) \frac{\sqrt{N(s, \cdot)}}{N(s, a) + 1} \quad (1)$$

In the whole training iterations (including the first  $I'$  iterations), the **Baseline** player always runs neural network based MCTS (i.e line 6 in Algorithm 1 is simply replaced by  $\pi_t \leftarrow \text{MCTS}$ ).

### 4.3 MCTS Enhancements

In this paper, we introduce 2 individual enhancements and 3 combinations to improve neural network training based on MCTS (Algorithm 2).

**Rollout.** Algorithm 2 uses the value from the value network as return value at leaf nodes. However, if the neural network is not yet well trained, the values are not accurate, and even random at the start phase, which can lead to biased and slow training. Therefore, as warm-start enhancement we perform a classic MCTS random rollout to get a value that provides more meaningful information. We thus simply add a random rollout function which returns a terminal value after line 9 in Algorithm 2, written as *Get result  $v(s)$  by performing random rollout until the game ends.*<sup>1</sup>

**RAVE** is a well-studied enhancement for improving the cold-start of MCTS in games like Go (for details see [9]). The same idea can be applied to other domains

<sup>1</sup> In contrast to AlphaGo [1], where random rollouts were mixed in with all value-lookups, in our scheme they replace the network lookup at the start of the training.

where the playout-sequence can be transposed. Standard MCTS only updates the  $(s, a)$ -pair that has been visited. The RAVE enhancement extends this rule to any action  $a$  that appears in the sub-sequence, thereby rapidly collecting more statistics in an off-policy fashion. The idea to perform RAVE at startup is adapted from AMAF in the game of Go [9]. The main pseudo code of RAVE is similar to Algorithm 2, the differences are in line 3, line 12 and line 16. For RAVE, in line 3, policy  $\pi_s$  is normalized based on  $Q_{rave}(s, \cdot)$ . In line 12, the action  $a$  with highest  $UCT_{rave}$  value, which is computed based on Eq. 2, is selected. After line 16, the idea of AMAF is applied to update  $N_{rave}$  and  $Q_{rave}$ , which are written as:  $N_{rave}(s_{t_1}, a_{t_2}) \leftarrow N_{rave}(s_{t_1}, a_{t_2}) + 1$ ,  $Q_{rave}(s_{t_1}, a_{t_2}) \leftarrow \frac{N_{rave}(s_{t_1}, a_{t_2}) * Q_{rave}(s_{t_1}, a_{t_2}) + v}{N_{rave}(s_{t_1}, a_{t_2}) + 1}$ , where  $s_{t_1} \in VisitedPath$ , and  $a_{t_2} \in A(s_{t_1})$ , and for  $\forall t < t_2, a_t \neq a_{t_2}$ . More specifically, under state  $s_t$ , in the visited path, a state  $s_{t_1}$ , all legal actions  $a_{t_2}$  of  $s_{t_1}$  that appear in its sub-sequence ( $t \leq t_1 < t_2$ ) are considered as a  $(s_{t_1}, a_{t_2})$  tuple to update their  $Q_{rave}$  and  $N_{rave}$ .

$$UCT_{rave}(s, a) = (1 - \beta) * U(s, a) + \beta * U_{rave}(s, a) \quad (2)$$

where

$$U_{rave}(s, a) = Q_{rave}(s, a) + c * P(s, a) \frac{\sqrt{N_{rave}(s, \cdot)}}{N_{rave}(s, a) + 1}, \quad (3)$$

and

$$\beta = \sqrt{\frac{equivalence}{3 * N(s, \cdot) + equivalence}} \quad (4)$$

Usually, the value of equivalence is set to the number of MCTS simulations (i.e.  $m$ ), as is also the case in our following experiments.

**RoRa.** Based on Rollout and Rave enhancement, the first combination is to simply add the random rollout to enhance RAVE.

**WRo.** As the neural network model is getting better, we introduce a weighted sum of rollout value and the value network as the return value. In our experiments,  $v(s)$  is computed as follows:

$$v(s) = (1 - weight) * v_{network} + weight * v_{rollout} \quad (5)$$

**WRoRa.** In addition, we also employ a weighted sum to combine the value a neural network and the value of RoRa. In our experiments, weight  $weight$  is related to the current iteration number  $i, i \in [0, I']$ .  $v(s)$  is computed as follows:

$$v(s) = (1 - weight) * v_{network} + weight * v_{rora} \quad (6)$$

where

$$weight = 1 - \frac{i}{I'} \quad (7)$$

## 5 Orientation Experiment: MCTS(RAVE) vs. RHEA

Before running full scale experiments on warm-start self-play that take days to weeks, we consider other possibilities for methods that could be used instead of MCTS variants. Justesen et al. [38] have recently shown that depending on the type of game that is played, RHEA can actually outperform MCTS variants also on adversarial games. Especially for long games, RHEA seems to be strong because MCTS is not able to reach a good tree/opening sequence coverage.

The general idea of RHEA has been conceived by Perez et al. [39] and is simple: they directly optimize an action sequence for the next actions and apply the first action of the best found sequence for every move. Originally, this has been applied to one-player settings only, but recently different approaches have been tried also for adversarial games, as the co-evolutionary variant of Liu et al. [40] that shows to be competitive in 2 player competitions [41]. The current state of RHEA is documented in [42], where a large number of variants, operators and parameter settings is listed. No one-beats-all variant is known at this moment.

Generally, the horizon (number of actions in the planned sequence) is often much too short to reach the end of the game. In this case, either a value function is used to assess the last reached state, or a rollout is added. For adversarial games, opponent moves are either co-evolved, or also played randomly. We do the latter, with a horizon size of 10. In preliminary experiments, we found that a number of 100 rollouts is already working well for MCTS on our problems, thus we also applied this for the RHEA. In order to use these 100 rollouts well, we employ a population of only 10 individuals, using only cloning + mutation (no crossover) and a  $(10 + 1)$  truncation selection (the worst individual from 10 parents and 1 offspring is removed). The mutation rate is set to 0.2 per action in the sequence. However, parameters are not sensitive, except rollouts. RHEA already works with 50 rollouts, albeit worse than with 100. As our rollouts always reach the end of the game, we usually get back  $Q_i(as) = \{1, -1\}$  for the  $i$ -th rollout for the action sequence  $as$ , meaning we win or lose. Counting the number of steps until this happens  $h$ , we compute the fitness of an individual to  $Q(as) = \frac{\sum_{i=1}^n Q_i(as)/h}{n}$  over multiple rollouts, thereby rewarding quick wins and slow losses. We choose  $n = 2$  (rollouts per individual) as it seems to perform a bit more stable than  $n = 1$ . We thus evaluate 50 individuals per run.

In our comparison experiment, we pit a random player, MCTS, RAVE (both without neural network support but a standard random rollout), and RHEA against each other with 500 repetitions over all three games, with 100 rollouts per run for all methods. The results are shown in Table 1.

The results indicate that in nearly all cases, RAVE is better than MCTS is better than RHEA is better than random, according to a binomial test at a significance level of 5%. Only for Othello, RHEA does not convincingly beat the random player. We can conclude from these results that RHEA is no suitable alternative in our case. The reason for this may be that the games are rather short so that we always reach the end, providing good conditions for MCTS and even more so for RAVE that more aggressively summarizes rollout information.



**Table 1.** Comparison of random player, MCTS, Rave, and RHEA on the three games, win rates in percent (column vs. row), 500 repetitions each.

adv	Gobang				Connect Four				Othello			
	rand	mcts	rave	rhea	rand	mcts	rave	rhea	rand	mcts	rave	rhea
random		97.0	100.0	90.0		99.6	100.0	80.0		98.50	98.0	48.0
mcts	3.0		89.4	34.0	0.4		73.0	3.0	1.4		46.0	1.0
rave	0.0	10.6		17.0	0.0	27.0		4.0	2.0	54.0		5.0
rhea	10.0	66.0	83.0		20.0	97.0	96.0		52.0	99.0	95.0	

Besides, start sequence planning is certainly harder for Othello where a single move can change large parts of the board.

## 6 Full Length Experiment

Taking into account the results of the comparison of standard MCTS/RAVE and RHEA at small scale, we now focus on the previously defined neural network based MCTS and its enhancements and run them over the full scale training.

### 6.1 Experiment Setup

For all 3 tested games and all experimental training runs based on Algorithm 1, we set parameters values in Table 2. Since tuning  $I'$  requires enormous computation resources, we set the value to 5 based on an initial experiment test, which means that for each self-play training, only the first 5 iterations will use one of the warm-start enhancements, after that, there will be only the MCTS in Algorithm 2. Other parameter values are set based on [43, 44].

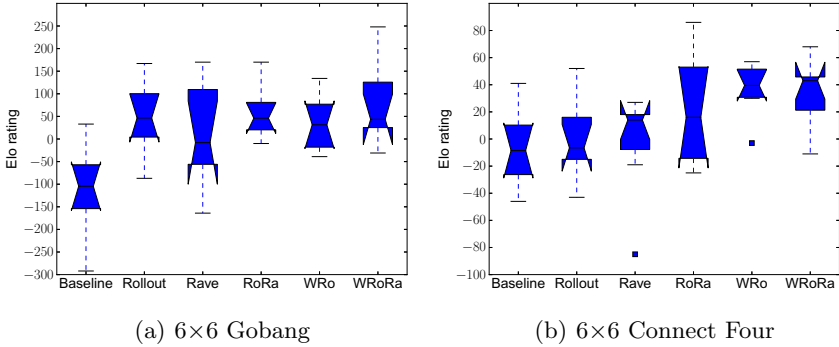
Our experiments are run on a GPU-machine with 2x Xeon Gold 6128 CPU at 2.6 GHz, 12 core, 384 GB RAM and 4x NVIDIA PNY GeForce RTX 2080TI. We use small versions of games ( $6 \times 6$ ) in order to perform a sufficiently high number of computationally demanding experiments. Shown are graphs with errorbars of 8 runs, of 100 iterations of self-play. Each single run takes 1 to 2 days.

**Table 2.** Default parameter setting

Para	Description	Value	Para	Description	Value
$I$	Number of iteration	100	$rs$	Number of retrain iteration	20
$I'$	Iteration threshold	5	$ep$	Number of epoch	10
$E$	Number of episode	50	$bs$	Batch size	64
$T'$	Step threshold	15	$lr$	Learning rate	0.005
$m$	MCTS simulation times	100	$d$	Dropout probability	0.3
$c$	Weight in UCT	1.0	$n$	Number of comparison games	40
$u$	Update threshold	0.6			

## 6.2 Results

After training, we collect 8 repetitions for all 6 categories players. Therefore we obtain 49 players in total (a Random player is included for comparison). In a full round robin tournament, every 2 of these 49 players are set to pit against each other for 20 matches on 3 different board games (Gobang, Connect Four and Othello). The Elo ratings are calculated based on the competition results using the same Bayesian Elo computation [45] as AlphaGo papers.

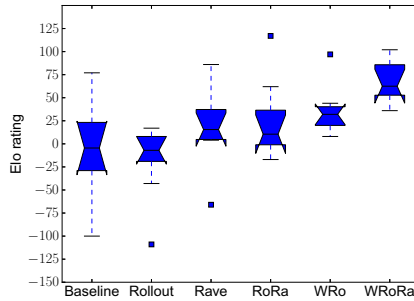


**Fig. 2.** Tournament results for  $6 \times 6$  Gobang and  $6 \times 6$  Connect Four among *Baseline*, *Rollout*, *Rave*, *RoRa*, *WRo* and *WRoRa*. Training with enhancements tends to be better than baseline MCTS.

Figure 2(a) displays results for training to play the  $6 \times 6$  Gobang game. We can clearly see that all players with the enhancement achieve higher Elo ratings than the Baseline player. For the Baseline player, the average Elo rating is about  $-100$ . For enhancement players, the average Elo ratings are about  $50$ , except for Rave, whose variance is larger. Rollout players and its combinations are better than the single Rave enhancement players in terms of the average Elo. In addition, the combination of Rollout and RAVE does not achieve significant improvement of Rollout, but is better than RAVE. This indicates that the contribution of the Rollout enhancement is larger than RAVE in Gobang game.

Figure 2(b) shows that all players with warm-start enhancement achieve higher Elo ratings in training to play the  $6 \times 6$  Connect Four game. In addition, we find that comparing Rollout with WRo, a weighted sum of rollout value and neural network value achieves higher performance. Comparing Rave and WRoRa, we see the same. We conclude that in 5 iterations, for Connect Four, enhancements that combine the value derived from the neural network contribute more than the pure enhancement value. Interestingly, in Connect Four, the combination of Rollout and RAVE shows improvement, in contrast to Othello (next figure) where we do not see significant improvement. However, this does not apply to WRoRa, the weighted case.

In Fig 3 we see that in Othello, except for Rollout which holds the similar Elo rating as Baseline setting, all other investigated enhancements are better than the Baseline. Interestingly, the enhancement with weighted sum of RoRa and neural network value achieves significant highest Elo rating. The reason that Rollout does not show much improvement could be that the rollout number is not large enough for the game length ( $6 \times 6$  Othello needs 32 steps for every episode to reach the game end, other 2 games above may end up with vacant positions). In addition, Othello does not have many transposes as Gobang and Connect Four which means that RAVE can not contribute to a significant improvement. We can definitively state that the improvements of these enhancements are sensitive to the different games. In addition, for all 3 tested games, at least WRoRa achieves the best performance according to a binomial test at a significance level of 5%.



**Fig. 3.** Tournament results for  $6 \times 6$  Othello among *Baseline*, *Rollout*, *Rave*, *RoRa*, *WRo* and *WRoRa*. Training with enhancements is mostly better than the baseline setting.

## 7 Discussion and Conclusion

Self-play has achieved much interest due to the AlphaGo Zero results. However, self-play is currently computationally very demanding, which hinders reproducibility and experimenting for further improvements. In order to improve performance and speed up training, in this paper, we investigate the possibility of utilizing MCTS enhancements to improve AlphaZero-like self-play. We embed Rollout, RAVE and their possible combinations as enhancements at the start period of iterative self-play training. The hypothesis is, that self-play suffers from a cold-start problem, as the neural network and the MCTS statistics are initialized to random weights and zero, and that this can be cured by prepending it with running MCTS enhancements or similar methods alone in order to train the neural network before “switching it on” for playing.

We introduce Rollout, RAVE, and combinations with network values, in order to quickly improve MCTS tree statistics before we switch to Baseline-like self-play training, and test these enhancements on  $6 \times 6$  versions of Gobang, Connect Four, and Othello. We find that, after 100 self-play iterations, we still see the

effects of the warm-start enhancements as playing strength has improved in many cases. For different games, different methods work best; there is at least one combination that performs better. It is hardly possible to explain the performance coming from the warm-start enhancements and especially to predict for which games they perform well, but there seems to be a pattern: Games that enable good static opening plans probably benefit more. For human players, it is a common strategy in Connect Four to play a middle column first as this enables many good follow-up moves. In Gobang, the situation is similar, only in 2D. It is thus harder to counter a good plan because there are so many possibilities. This could be the reason why the warm-start enhancements work so well here. For Othello, the situation is different, static openings are hardly possible, and are thus seemingly not detected. One could hypothesize that the warm-start enhancements recover human expert knowledge in a generic way. Recently, we have seen that human knowledge is essential for mastering complex games as StarCraft [46], whereas others as Go [2] can be learned from scratch. Re-generating human knowledge may still be an advantage, even in the latter case.

We also find that often, a single enhancement may not lead to significant improvement. There is a tendency for the enhancements that work in combination with the value of the neural network to be stronger, but that also depends on the game. Concluding, we can state that we find moderate performance improvements when applying warm-start enhancements and that we expect there is untapped potential for more performance gains here.

## 8 Outlook

We are not aware of other studies on warm-start enhancements of AlphaZero-like self-play. Thus, a number of interesting problems remain to be investigated.

- Which enhancements will work best on which games? Does the above hypothesis hold that games with more consistent opening plans benefit more from the warm-start?
- When (parameter  $I'$ ) and how do we lead over from the start methods to the full AlphaZero scheme including MCTS and neural networks? If we use a weighting, how shall the weight be changed when we lead over? Linearly?
- There are more parameters that are critical and that could not really be explored yet due to computational cost, but this exploration may reveal important performance gains.
- Other warm-start enhancements, e.g. built on variants of RHEA's or hybrids of it, shall be explored.
- All our current test cases are relatively small games. How does this transfer to larger games or completely different applications?

In consequence, we would like to encourage other researchers to help exploring this approach and enable using its potential in future investigations.

**Acknowledgments.** Hui Wang acknowledges financial support from the China Scholarship Council (CSC), CSC No.201706990015.

## References

1. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. *Nature* **529**(7587), 484 (2016)
2. Silver, D., et al.: Mastering the game of go without human knowledge. *Nature* **550**(7676), 354 (2017)
3. Silver, D., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* **362**(6419), 1140–1144 (2018)
4. Tao, J., Lin, W., Xiaofeng, H.: Principle analysis on AlphaGo and perspective in military application of artificial intelligence. *J. Command Control* **2**(2), 114–120 (2016)
5. Zhang, Z.: When doctors meet with AlphaGo: potential application of machine learning to clinical medicine. *Ann. Transl. Med.* **4**(6) (2016)
6. Browne, C., et al.: A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intell. AI Games* **4**(1), 1–43 (2012)
7. Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
8. Plaatt, A.: Learning to play—reinforcement learning and games (2020)
9. Gelly, S., Silver, D.: Combining online and offline knowledge in UCT. In: Proceedings of the 24th International Conference on Machine Learning, pp. 273–280 (2007)
10. Nair, S.: AlphaZero general. <https://github.com/suragnair/alpha-zero-general> (2018). Accessed May 2018
11. Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In: van den Herik, H.J., Ciancarini, P., Donkers, H.H.L.M.J. (eds.) CG 2006. LNCS, vol. 4630, pp. 72–83. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-75538-8\\_7](https://doi.org/10.1007/978-3-540-75538-8_7)
12. Ruijl, B., Vermaseren, J., Plaatt, A., van den Herik, J.: Combining simulated annealing and Monte Carlo tree search for expression simplification. In: Proceedings of the 6th International Conference on Agents and Artificial Intelligence-Volume 1, pp. 724–731. SCITEPRESS-Science and Technology Publications, Lda (2014)
13. Chaslot, G., Bakkes, S., Szita, I., Spronck, P.: Monte-Carlo tree search: a new framework for game AI. In: AIIDE (2008)
14. Gelly, S., Silver, D.: Monte-Carlo tree search and rapid action value estimation in computer go. *Artif. Intell.* **175**(11), 1856–1875 (2011)
15. Schmidhuber, J.: Deep learning in neural networks: an overview. *Neural Netw.* **61**, 85–117 (2015)
16. Clark, C., Storkey, A.: Training deep convolutional neural networks to play go. In: International Conference on Machine Learning, pp. 1766–1774 (2015)
17. Tesauro, G.: Temporal difference learning and TD-Gammon. *Commun. ACM* **38**(3), 58–68 (1995)
18. Heinz, E.A.: New self-play results in computer chess. In: Marsland, T., Frank, I. (eds.) CG 2000. LNCS, vol. 2063, pp. 262–276. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-45579-5\\_18](https://doi.org/10.1007/3-540-45579-5_18)
19. Wiering, M.A., et al.: Self-play and using an expert to learn to play backgammon with temporal difference learning. *J. Intell. Learn. Syst. Appl.* **2**(02), 57 (2010)

20. Van Der Ree, M., Wiering, M.: Reinforcement learning in the game of Othello: learning against a fixed opponent and learning from self-play. In: IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), pp. 108–115. IEEE (2013)
21. Runarsson, T.P., Lucas, S.M.: Coevolution versus self-play temporal difference learning for acquiring position evaluation in small-board go. *IEEE Trans. Evol. Comput.* **9**(6), 628–640 (2005)
22. Wang, H., Emmerich, M., Plaat, A.: Monte Carlo Q-learning for general game playing. arXiv preprint [arXiv:1802.05944](https://arxiv.org/abs/1802.05944) (2018)
23. Wu, D.J.: Accelerating self-play learning in go. arXiv preprint [arXiv:1902.10565](https://arxiv.org/abs/1902.10565) (2019)
24. Iwata, S., Kasai, T.: The Othello game on an  $n * n$  board is PSPACE-complete. *Theor. Comput. Sci.* **123**(2), 329–340 (1994)
25. Allis, L.V.: A knowledge-based approach of connect-four. *ICGA J.* **11**(4), 165 (1988)
26. Reisch, S.: Gobang ist pspace-vollständig. *Acta Informatica* **13**(1), 59–66 (1980)
27. Buro, M.: The Othello match of the year: Takeshi Murakami vs. Logistello. *ICGA J.* **20**(3), 189–193 (1997)
28. Chong, S.Y., Tan, M.K., White, J.D.: Observing the evolution of neural networks learning to play the game of Othello. *IEEE Trans. Evol. Comput.* **9**(3), 240–251 (2005)
29. Thill, M., Bagheri, S., Koch, P., Konen, W.: Temporal difference learning with eligibility traces for the game connect four. In: IEEE Conference on Computational Intelligence and Games, pp. 1–8. IEEE (2014)
30. Zhang, M.L., Wu, J., Li, F.Z.: Design of evaluation-function for computer gobang game system. *J. Comput. Appl.* **7**, 051 (2012)
31. Banerjee, B., Stone, P.: General game learning using knowledge transfer. In: IJCAI, pp. 672–677 (2007)
32. Wang, H., Emmerich, M., Plaat, A.: Assessing the potential of classical Q-learning in general game playing. In: Atzmueller, M., Duivesteyn, W. (eds.) BNAIC 2018. CCIS, vol. 1021, pp. 138–150. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-31978-6\\_11](https://doi.org/10.1007/978-3-030-31978-6_11)
33. Wang, H., Emmerich, M., Preuss, M., Plaat, A.: Alternative loss functions in alphazero-like self-play. In: IEEE Symposium Series on Computational Intelligence (SSCI), pp. 155–162. IEEE (2019)
34. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint [arXiv:1502.03167](https://arxiv.org/abs/1502.03167) (2015)
35. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
36. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014)
37. Rosin, C.D.: Multi-armed bandits with episode context. *Ann. Mathe. Artif. Intell.* **61**(3), 203–230 (2011). <https://doi.org/10.1007/s10472-011-9258-6>
38. Justesen, N., Mahlmann, T., Risi, S., Togelius, J.: Playing multi-action adversarial games: online evolutionary planning versus tree search. *IEEE Trans. Comput. Intell. AI Games* **10**, 281–291 (2017)
39. Perez, D., Samothrakis, S., Lucas, S., Rohlfschagen, P.: Rolling horizon evolution versus tree search for navigation in single-player real-time games. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO 2013, pp. 351–358. New York (2013). Association for Computing Machinery

40. Liu, J., Liebana, D.P., Lucas, S.M.: Rolling horizon coevolutionary planning for two-player video games. In: 8th Computer Science and Electronic Engineering Conference, CEEC 2016, Colchester, UK, 28–30 September 2016, pp. 174–179. IEEE (2016)
41. Gaina, R.D., et al.: The 2016 two-player GVGAI competition. *IEEE Trans. Games* **10**(2), 209–220 (2018)
42. Gaina, R.D., Devlin, S., Lucas, S.M., Perez-Liebana, D.: Rolling horizon evolutionary algorithms for general video game playing (2020)
43. Wang, H., Emmerich, M., Preuss, M., Plaat, A.: Hyper-parameter sweep on AlphaZero general. arXiv preprint [arXiv:1903.08129](https://arxiv.org/abs/1903.08129) (2019)
44. Wang, H., Emmerich, M., Preuss, M., Plaat, A.: Analysis of hyper-parameters for small games: iterations or epochs in self-play? arXiv preprint [arXiv:2003.05988](https://arxiv.org/abs/2003.05988) (2020)
45. Coulom, R.: Whole-history rating: a Bayesian rating system for players of time-varying strength. In: van den Herik, H.J., Xu, X., Ma, Z., Winands, M.H.M. (eds.) CG 2008. LNCS, vol. 5131, pp. 113–124. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-87608-3\\_11](https://doi.org/10.1007/978-3-540-87608-3_11)
46. Vinyals, O., et al.: Grandmaster level in StarCraft ii using multi-agent reinforcement learning. *Nature* **575**(7782), 350–354 (2019)

# **Theoretical Aspects of Nature-Inspired Optimization**





# Runtime Analysis of a Heavy-Tailed $(1 + (\lambda, \lambda))$ Genetic Algorithm on Jump Functions

Denis Antipov<sup>1,2</sup>(✉) and Benjamin Doerr<sup>2</sup>

<sup>1</sup> ITMO University, St. Petersburg, Russia  
antipovden@yandex.ru

<sup>2</sup> Laboratoire d'Informatique (LIX), CNRS, École Polytechnique,  
Institut Polytechnique de Paris, Palaiseau, France

**Abstract.** It was recently observed that the  $(1 + (\lambda, \lambda))$  genetic algorithm can comparably easily escape the local optimum of the jump functions benchmark. Consequently, this algorithm can optimize the jump function with jump size  $k$  in an expected runtime of only  $n^{(k+1)/2}k^{-k/2}e^{O(k)}$  fitness evaluations (Antipov, Doerr, Karavaev (GECCO 2020)). This performance, however, was obtained with non-standard parameter setting depending on the jump size  $k$ .

To overcome this difficulty, we propose to choose two parameters of the  $(1 + (\lambda, \lambda))$  genetic algorithm randomly from a power-law distribution. Via a mathematical runtime analysis, we show that this algorithm with natural instance-independent choices of the power-law parameters on all jump functions with jump size at most  $n/4$  has a performance close to what the best instance-specific parameters in the previous work obtained. This price for instance-independence can be made as small as an  $O(n \log(n))$  factor. Given the difficulty of the jump problem and the runtime losses from using mildly suboptimal fixed parameters (also discussed in this work), this appears to be a fair price.

**Keywords:** Theory · Runtime analysis · Crossover · Fast mutation

## 1 Introduction

The  $(1 + (\lambda, \lambda))$  genetic algorithm ( $(1 + (\lambda, \lambda))$  GA) is a still fairly simple evolutionary algorithm proposed at GECCO 2013 [14] (journal version [15]). Through a combination of mutation with a high mutation rate and crossover with the parent as repair mechanism, it tries to increase the speed of exploration without compromising in terms of exploitation. The mathematical analyses on ONEMAX [12, 15] and easy random satisfiability instances [6] showed that the new algorithm has a moderate advantage over classic evolutionary algorithms (EAs). Some experimental results [26, 31] also suggested that this algorithm is promising.

More recently, a mathematical analysis on jump functions showed that here the  $(1 + (\lambda, \lambda))$  GA with the right parameter setting outperforms the classic algorithms by a much wider margin than on the simpler problems regarded before [5]. One drawback of this result is that the choice of the parameters is non-trivial. In particular, (i) one needed to deviate from the previous recommendation to connect the mutation rate  $p$  and the crossover bias  $c$  to the population size  $\lambda$  via  $p = \frac{\lambda}{n}$  and  $c = \frac{1}{\lambda}$ , and (ii) the optimal parameters depended heavily on the difficulty parameter  $k$  of the jump functions class. While also many sub-optimal parameter values gave an improvement over classic algorithms, the non-trivial influence of the parameters on the algorithm performance still raises the question if one can (at least partially) relieve the algorithm designer from the task of choosing the parameters.

In this work, we make a big step forward in this direction. We deduce from previous works that taking mutation an equal rate  $p$  and crossover bias  $c$  can be a good idea when progress is difficult. This relation was found suitable in the last stages of the ONEMAX optimization process and to cross the fitness valley of jump functions. Parameterizing  $p = c = \sqrt{s/n}$ , we obtain that an offspring after mutation and crossover has an expected Hamming distance of  $s$  from the parent. Hence the parameter  $s$ , in a similar manner as the mutation rate in a traditional mutation-based algorithm, quantifies the typical search radius of the  $(1 + (\lambda, \lambda))$  GA. With this (heuristic) reduction of the parameter space, it remains to choose suitable values for the search radius  $s$  and for the offspring population size  $\lambda$ . Also with this reduction, this remains a non-trivial task—the optimal parameters determined in [5] are  $\lambda = \sqrt{\frac{n}{k}}$  and  $s = k$ .

One way to circumvent the parameter choice problem is letting the EA optimize its parameters itself. The last years have seen a decent number of self-adjusting or self-adapting parameter choices (e.g., [9, 16–18, 20, 29, 30], see also the survey [13]), including a self-adjusting choice of  $\lambda$  for the  $(1 + (\lambda, \lambda))$  GA optimizing ONEMAX [12, 15] and easy random SAT instance [6]. In all these successful applications of dynamic parameter settings, the characteristic of the optimization process changed only slowly over time. This enabled the algorithm to adjust to the changing environment. We are therefore sceptical that such ideas work well on problems like jump functions, which show a sudden change from an easy ONEMAX-style landscape to a difficult-to-cross fitness valley.

For this reason, we preferred another recently successful approach, namely a random choice of the parameters. Such a random choice (from a heavy-tailed distribution, independently in each iteration) of the mutation rate was shown to give good results for the  $(1 + 1)$  EA optimizing jump functions [19] (see [1, 2, 22, 23, 25, 31, 35, 36] for other successful uses of this idea). Hence trying this idea for our parameter  $s$  is very natural. There is less a-priori evidence that a random choice of the value for  $\lambda$  is a good idea, but we have tried this nevertheless. We note that the recent work [1] showed that the  $(1 + (\lambda, \lambda))$  GA with a heavy-tailed choice of  $\lambda$  and the previous recommendation  $p = \frac{\lambda}{n}$  and  $c = \frac{1}{\lambda}$  has a good performance on ONEMAX, but it is not clear why this should indicate also

a good performance on jump functions, in particular, with our different choice of  $p$  and  $c$ .

We conduct a mathematical runtime analysis of the  $(1 + (\lambda, \lambda))$  GA with heavy-tailed choices of  $s$  and  $\lambda$ , the heavy-tailed  $(1 + (\lambda, \lambda))$  GA for short, from a broad range of power-law distributions. It shows that a power-law exponent  $\beta_s > 1$  for the choice of  $s$  and a power-law exponent  $\beta_\lambda$  equal to or slightly above two for the choice of  $\lambda$  gives a very good performance. The resulting runtimes are slightly higher than those stemming from the best instance-specific static parameters, but are still much below the runtimes of classic evolutionary algorithms.

While undoubtedly we have obtained parameters that work uniformly well over all jump functions, we also feel that our choices of the power-law exponent are quite natural, so that the name parameterless  $(1 + (\lambda, \lambda))$  GA might be justified. There is not much to say on the choice of  $s$ , where apparently all power-laws (with exponent greater than one, which is a very natural assumption for any use of a power-law) give good results. For the choice of  $\lambda$ , we note that the cost of one iteration of the  $(1 + (\lambda, \lambda))$  GA is  $2\lambda$  fitness evaluations. Hence  $2E[\lambda]$  is the cost of an iteration with a random choice of  $\lambda$ . Now it is exactly the power-law exponents  $\beta_\lambda > 2$  that give a constant value for  $E[\lambda]$ . The larger  $\beta_\lambda$  is, the more the power-law distribution is concentrated on constant values. For constant  $\lambda$ , however, the  $(1 + (\lambda, \lambda))$  GA cannot profit a lot from the intermediate selection step, and thus shows a behavior similar to classic mutation-based algorithms. For this reason, choosing a power-law exponent close to two appears to be a natural choice. Based both on this informal argument and our mathematical results, for a practical application of our algorithm we recommend to use  $\beta_s$  slightly above one, say 1.1, and  $\beta_\lambda$  slightly above two, say 2.1.

The asymptotically best choice of  $\beta_\lambda$  (in the sense that the worst-case price for being instance-independent is lowest) is obtained from taking  $\beta_\lambda = 2$ . Since this alone would give an infinite value for  $E[\lambda]$ , one needs to restrict the range of values this distribution is defined on. To obtain an  $O(nk^{\beta_s-1})$  price of instance-independence, already a generous upper bound of  $2^n$  is sufficient. To obtain our best price of instance-independence of  $O(n \log n)$ , a similar trick is necessary for the choice of  $s$ , namely taking  $\beta_s = 1$  and capping the range at the (trivial) upper bound  $s \leq n$ . While we think that these considerations are interesting from the theoretical perspective as they explore the limits of our approach, we do not expect these hyperparameter choices to be useful in many practical applications. We note the runtime of the  $(1 + 1)$  EA with heavy-tailed mutation rate was shown [19] to exceed the instance-specific best runtime of the  $(1 + 1)$  EA by a factor of  $\Theta(n^{\beta-0.5})$ . Hence a power-law exponent  $\beta$  as low as possible (but larger than one) looks best from the theoretical perspective. In contrast, in the experiments in [19], no improvement was seen from lowering  $\beta$  below 1.5.

The remainder of this paper is structured as follows. In the following preliminaries section, we introduce the jump functions benchmark and the heavy-tailed  $(1 + (\lambda, \lambda))$  GA along with some relevant previous works. Section 3 contains the heart of this work, a mathematical runtime analysis of the heavy-tailed

$(1 + (\lambda, \lambda))$  GA on jump functions. In Sect. 4, we show via an elementary computational analysis that the  $(1 + (\lambda, \lambda))$  GA with fixed parameters is very sensitive to missing the optimal parameter values. This suggests that the small polynomial price of our one-size-fits-all solution is well invested compared to the performance losses stemming from missing the optimal static parameter values.

## 2 Preliminaries

In this section we collect all necessary definitions and tools, which we use in the paper. We only use standard notation such as the following. By  $\mathbb{N}$  we denote the set of positive integers. We use notations  $[a..b]$  for integer intervals and  $[a, b]$  for real-valued intervals. For  $a, b \in \mathbb{R}$  the notion  $[a..b]$  means  $[[a]..[b]]$ . For any probability distribution  $\mathcal{L}$  and random variable  $X$ , we write  $X \sim \mathcal{L}$  to indicate that  $X$  follows the law  $\mathcal{L}$ . We denote the binomial law with parameters  $n \in \mathbb{N}$  and  $p \in [0, 1]$  by  $\text{Bin}(n, p)$ .

### 2.1 Jump Functions

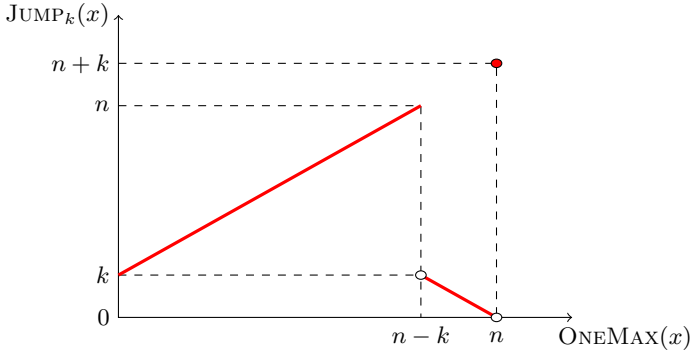
The family of jump functions is a class of model functions based on the classic ONEMAX benchmark function. ONEMAX is a pseudo-Boolean function defined on the space of bit-strings of length  $n$ , which returns the number of one-bits in its argument. More formally,

$$\text{ONEMAX}(x) = \text{OM}(x) = \sum_{i=1}^n x_i.$$

The  $\text{JUMP}_k$  function with jump size  $k$  is then defined as follows.

$$\text{JUMP}_k(x) = \begin{cases} \text{OM}(x) + k, & \text{if } \text{OM}(x) \in [0..n - k] \cup \{n\}, \\ n - \text{OM}(x), & \text{if } \text{OM}(x) \in [n - k + 1..n - 1]. \end{cases}$$

A plot of  $\text{JUMP}_k$  is shown in Fig. 1. Different from ONEMAX, this function has a fitness valley which is hard to cross for the many EAs. For example, the  $(\mu + \lambda)$  EA and  $(\mu, \lambda)$  EA for all values of  $\mu$  and  $\lambda$  need an expected time of  $\Omega(n^k)$  to optimize  $\text{JUMP}_k$  [11, 21]. With a heavy-tailed mutation operator, the runtime of the  $(1 + 1)$  EA can be lowered by a  $k^{\Theta(k)}$  factor, so it remains  $\Theta(n^k)$  for  $k$  constant. Better runtimes have been shown for algorithms using crossover and other mechanisms [7, 8, 24, 28, 32, 34] and for estimation-of-distribution algorithms [10, 27], though in our view only the  $O(n^{k-1})$  runtime in [8] stems from a classic algorithm with natural parameters.



**Fig. 1.** Plot of the  $JUMP_k$  function. As a function of unitation, the function value of a search point  $x$  depends only on the number  $ONEMAX(x)$  of one-bits in  $x$ .

### 2.2 Power-Law Distributions

We say that a random variable  $X \in \mathbb{N}$  follows a power-law distribution with parameters  $\beta$  and  $u$  if

$$\Pr[X = i] = \begin{cases} C_{\beta,u} i^{-\beta}, & \text{if } i \in [1..u], \\ 0, & \text{else,} \end{cases}$$

where  $C_{\beta,u} = (\sum_{j=1}^u j^{-\beta})^{-1}$  is the normalization coefficient. We write  $X \sim \text{pow}(\beta, u)$  and call  $u$  the upper limit and  $\beta$  the power-law exponent. We note that if  $\beta > 1$ , then  $\Pr[X = i] = \Theta(1)$  for any integer  $i = \Theta(1)$ . At the same time the distribution is *heavy-tailed*, which means that we have a decent (only inverse polynomial instead of negative-exponential) probability that  $X = i$  for any super-constant  $i \leq u$ . If  $\beta > 2$ , then we also have  $E[X] = \Theta(1)$ . These properties are easily seen from the following estimates of the partial sums of the generalized harmonic series, which we will frequently need in this work.

**Lemma 1.** For all positive integers  $a$  and  $b$  such that  $b \geq a$  and for all  $\beta > 0$ , the sum  $\sum_{i=a}^b i^{-\beta}$  is

- $\Theta((b + 1)^{1-\beta} - a^{1-\beta})$ , if  $\beta \in [0, 1)$ ,
- $\Theta(\log(\frac{b+1}{a}))$ , if  $\beta = 1$ , and
- $\Theta(a^{1-\beta} - (b + 1)^{1-\beta})$ , if  $\beta > 1$ ,

where  $\Theta$  notation is used for  $a$  and  $b$  tending to  $+\infty$ .

This lemma is easily shown by approximating the sums via integrals. It gives the following estimates for the normalization coefficient  $C_{\beta,u}$  of the power-law distribution and for the expected value of  $X \sim \text{pow}(\beta, u)$ .

**Lemma 2.** The normalization coefficient  $C_{\beta,u} = (\sum_{j=1}^u i^{-\beta})^{-1}$  of the power-law distribution with parameters  $\beta$  and  $u$  is

- $\Theta(u^{\beta-1})$ , if  $\beta \in [0, 1)$ ,
- $\Theta(1/\log(u+1))$ , if  $\beta = 1$ , and
- $\Theta(1)$ , if  $\beta > 1$ .

**Lemma 3.** *The expected value of  $X \sim \text{pow}(\beta, u)$  is*

- $\Theta(u)$ , if  $\beta \leq 1$ ,
- $\Theta(u^{2-\beta})$ , if  $\beta \in (1, 2)$ ,
- $\Theta(\log(u+1))$ , if  $\beta = 2$ , and
- $\Theta(1)$ , if  $\beta > 2$ .

In both Lemmas we use  $\Theta$  notation for  $u \rightarrow +\infty$ .

### 2.3 The Heavy-Tailed $(1 + (\lambda, \lambda))$ GA

We now define a variant of the  $(1 + (\lambda, \lambda))$  GA, which we call *heavy-tailed*  $(1 + (\lambda, \lambda))$  GA. The main difference from the standard  $(1 + (\lambda, \lambda))$  GA is that at the start of each iteration the mutation rate  $p$ , the crossover bias  $c$ , and the population sizes  $\lambda_m$  and  $\lambda_c$  for the mutation and crossover phases are randomly chosen as follows. We sample  $s \sim \text{pow}(\beta_s, u_s)$  and take  $p = c = (\frac{s}{n})^{1/2}$ . The population sizes are chosen via  $\lambda_m = \lambda_c = \lambda \sim \text{pow}(\beta_\lambda, u_\lambda)$ . Here the upper limits  $u_\lambda$  and  $u_s$  can be any positive integers and the power-law exponents  $\beta_\lambda$  and  $\beta_s$  can be any non-negative real numbers. We call these parameters of the power-law distribution the *hyperparameters of the heavy-tailed  $(1 + (\lambda, \lambda))$  GA* and we give recommendations on how to choose them in Sect. 3.1. The pseudocode of this algorithm is shown in Algorithm 1. We note that it is not necessary to store the whole offspring populations, since only the best individual has a chance to be selected as mutation or crossover winner. Hence also large values for  $\lambda$  are algorithmically feasible.

The few existing results for the  $(1 + (\lambda, \lambda))$  GA with static parameters show the following: With optimal static parameters, the algorithm optimizes ONEMAX in time roughly  $\Theta(n\sqrt{\frac{\log(n)\log\log\log(n)}{\log\log(n)}})$  [12]. With a suitable fitness dependent parameter choice or a self-adjusting parameter choice building on the one-fifth rule, this runtime can be lowered to  $\Theta(n)$ . Due to the weaker fitness-distance correlation, slightly inferior results have been shown in [6] for sufficiently dense random satisfiability instances in the planted solution model (and the experiments in [6] suggest that indeed the algorithm suffers from the weaker fitness-distance correlation). A runtime analysis [4] on LEADINGONES gave no better runtimes than the classic  $\Theta(n^2)$  bound, but at least it showed that also in the absence of a good fitness-distance correlation the  $(1 + (\lambda, \lambda))$  GA can be efficient by falling back to the optimization behavior of the  $(1 + 1)$  EA.

We use the following language (also for the standard  $(1 + (\lambda, \lambda))$  GA with fixed values for  $p, c, \lambda_m, \lambda_c$ ). We denote by  $T_I$  and  $T_F$  the number of iterations and the number of fitness evaluations performed until some event holds (which is always specified in the text). If the algorithm has already reached the local optimum, then we call the mutation phase *successful* if all the  $k$  zero-bits of

---

**Algorithm 1:** The heavy-tailed  $(1 + (\lambda, \lambda))$  GA maximizing a pseudo-Boolean function  $f$ .

---

```

1  $x \leftarrow$  random bit string of length  $n$ ;
2 while not terminated do
3   Choose  $s \sim \text{pow}(\beta_s, u_s)$ ;
4    $p \leftarrow (\frac{s}{n})^{1/2}$ ;
5    $c \leftarrow (\frac{s}{n})^{1/2}$ ;
6   Choose  $\lambda \sim \text{pow}(\beta_\lambda, u_\lambda)$ ;
7   Mutation phase:
8   Choose  $\ell \sim \text{Bin}(n, p)$ ;
9   for  $i \in [1..\lambda]$  do
10     $x^{(i)} \leftarrow$  a copy of  $x$ ;
11    Flip  $\ell$  bits in  $x^{(i)}$  chosen uniformly at random;
12  end
13   $x' \leftarrow \arg \max_{z \in \{x^{(1)}, \dots, x^{(\lambda)}\}} f(z)$ ;
14  Crossover phase:
15  for  $i \in [1..\lambda]$  do
16    Create  $y^{(i)}$  by taking each bit from  $x'$  with probability  $c$  and from  $x$ 
    with probability  $(1 - c)$ ;
17  end
18   $y \leftarrow \arg \max_{z \in \{y^{(1)}, \dots, y^{(\lambda)}\}} f(z)$ ;
19  if  $f(y) \geq f(x)$  then
20     $x \leftarrow y$ ;
21  end
22 end

```

---

$x$  are flipped to ones in the mutation winner  $x'$ . We also call an offspring of the mutation phase *good* if it has all  $k$  zero-bits flipped. If the algorithm has not reached the local optimum, then we call the mutation phase *successful* if  $x'$  contains a one-bit not present in  $x$ . In this case we call an offspring *good* if it has at least one zero-bit flipped to one and does not lie in the fitness valley of  $\text{JUMP}_k$ . We call the crossover phase *successful* if the crossover winner has a greater fitness than  $x$ . The *good* offspring in the crossover phase are those which have a better fitness than  $x$ .

To estimate the probability of a true progress in one iteration we use the following lemma, which can easily be deduced from Lemmas 3.1 and 3.2 in [5].

**Lemma 4.** *Let  $\lambda_m = \lambda_c = \lambda$  and  $p = c = (\frac{s}{n})^{1/2}$  with  $s \in [k..2k]$ . If the current individual  $x$  of the  $(1 + (\lambda, \lambda))$  GA is in the local optimum of  $\text{JUMP}_k$ , then the probability that the algorithm finds the global optimum in one iteration is at least  $e^{-\Theta(k)} \min\{1, (\frac{k}{n})^k \lambda^2\}$ .*

### 2.4 Wald’s Equation

Since not only the number of iterations until the optimum is found is a random variable, but also the number of fitness evaluations in each iteration, we shall use the following version of Wald’s equation [33] to estimate the number of fitness evaluations until the optimum is found.

**Lemma 5.** *Let  $(X_t)_{t \in \mathbb{N}}$  be a sequence of non-negative real-valued random variables with identical finite expectation. Let  $T$  be a positive integer random variable with finite expectation. If for all  $i \in \mathbb{N}$  the event  $\{T \geq i\}$  is independent of  $(X_t)_{t=i}^{+\infty}$ , then*

$$E \left[ \sum_{t=1}^T X_t \right] = E[T]E[X_1].$$

## 3 Heavy-Tailed Parameters

In this section we conduct a rigorous runtime analysis of the heavy-tailed  $(1 + (\lambda, \lambda))$  GA optimizing jump functions with jump size  $k \in [2.. \frac{n}{4}]$ . We cover the full spectrum of the algorithm’s hyperparameters  $\beta_s, u_s, \beta_\lambda, u_\lambda$ . For large ranges of the hyperparameters, in particular, for natural values like  $\beta_s = \beta_\lambda = 2 + \varepsilon$  and  $u_s = u_\lambda = \infty$ , we observe a performance that is only a little worse than the one with the best instance-specific static parameters. This price of instance-independence can be brought down to an  $O(n \log(n))$  factor. Taking into account the detrimental effect of missing the optimal fixed parameters shown in Sect. 4, this is a fair price for a one-size-fits-all algorithm.

Since a typical optimization process on jump functions consists of two very different regimes, we analyze separately the difficult regime of going from the local optimum to the global one (Sect. 3.1) and the easy ONEMAX-style regime encountered before that (Sect. 3.2).

### 3.1 Escaping the Local Optimum

The time to leave the local optimum (necessarily to the global one) is described in the following theorem and Table 1. We will see later that unless  $\beta_\lambda < 2$ , and this is not among our recommended choices, or  $k = 2$ , the time to reach the local optimum is not larger than the time to go from the local to the global optimum. Hence for  $\beta_\lambda \geq 2$  and for  $k \geq 3$ , the table also gives valid runtime estimates for the complete runtime.

**Theorem 6.** *Let  $k \in [2.. \frac{n}{4}]$ . Assume that we use the heavy-tailed  $(1 + (\lambda, \lambda))$  GA (Algorithm 1) to optimize  $\text{JUMP}_k$ , starting already in the local optimum. Then the expected number of the fitness evaluations until the optimum is found is shown in Table 1, where  $p_s$  denotes the probability that  $s \in [k..2k]$ . If  $u_s \geq 2k$ , then  $p_s$  is*



- $\Theta\left(\left(\frac{k}{u_s}\right)^{1-\beta_s}\right)$ , if  $\beta_s \in [0, 1)$ ,
- $\Theta\left(\frac{1}{\ln(u_s)}\right)$ , if  $\beta_s = 1$ , and
- $\Theta(k^{\beta_s-1})$ , if  $\beta_s > 1$ ,

where  $\Theta$  notation is used for  $n \rightarrow +\infty$ .

**Table 1.** Influence of the four hyperparameters  $\beta_s, u_s, \beta_\lambda, u_\lambda$  on the expected number  $E[T_F]$  of fitness evaluations the heavy-tailed  $(1 + (\lambda, \lambda))$  GA starting in the local optimum takes to optimize  $\text{JUMP}_k$ . Since all runtime bounds are of type  $E[T_F] = F(\beta_\lambda, u_\lambda)/p_s$ , where  $p_s = \Pr[s \in [k..2k]]$ , to ease reading we only state  $F(\beta_\lambda, u_\lambda) = E[T_F]p_s$ . By taking  $\beta_s = 2 + \varepsilon$  or  $\beta_s = 2 \wedge u_s = n$ , one obtains  $p_s = k^\varepsilon$  or  $p_s = O(\log n)$ . Using  $\beta_\lambda = 2$  and an exponential  $u_\lambda$  gives the lowest price of an  $O(n \log n)$  factor for being independent of the instance parameter  $k$ . We also advertise the slightly inferior combination  $\beta_\lambda = 2 + \varepsilon$  and  $u_\lambda = +\infty$  as for  $\beta_\lambda > 2$  each iteration has a constant expected cost and  $u_\lambda$  has no influence on the runtime (if chosen large enough). If  $\beta_\lambda \geq 2$  and  $k \geq 3$ , then the times stated are also the complete runtimes starting from a random initial solution.

$\beta_\lambda$	$E[T_F]p_s$ if $u_\lambda < \left(\frac{n}{k}\right)^{k/2}$	$E[T_F]p_s$ if $u_\lambda \geq \left(\frac{n}{k}\right)^{k/2}$
$[0, 1)$	$e^{\Theta(k)} \frac{1}{u_\lambda} \left(\frac{n}{k}\right)^k$	$u_\lambda e^{\Theta(k)}$
$= 1$		$u_\lambda e^{\Theta(k)} / \left(1 + \ln\left(u_\lambda \left(\frac{n}{k}\right)^{k/2}\right)\right)$
$(1, 2)$		$e^{\Theta(k)} u_\lambda^{2-\beta} \left(\frac{n}{k}\right)^{k/2(\beta-1)}$
$= 2$		$e^{\Theta(k)} \ln(u_\lambda) \left(\frac{n}{k}\right)^{k/2}$
$(2, 3)$	$e^{\Theta(k)} \frac{1}{u_\lambda^{3-\beta}} \left(\frac{n}{k}\right)^k$	$e^{\Theta(k)} \left(\frac{n}{k}\right)^{k/2(\beta-1)}$
$= 3$	$e^{\Theta(k)} \frac{1}{\ln(u_\lambda+1)} \left(\frac{n}{k}\right)^k$	$e^{\Theta(k)} \left(\frac{n}{k}\right)^k / \ln\left(\left(\frac{n}{k}\right)^k\right)$
$> 3$	$e^{\Theta(k)} \left(\frac{n}{k}\right)^k$	

From Theorem 6, we distill the following how to set the parameters of the power-law distributions.

*Distribution of  $\lambda$ :* When guessing  $u_\lambda$  right (depending on  $k$ ), and only then, then good runtimes can be obtained for  $\beta_\lambda < 2$ . Since we aim at a (mostly) parameterless approach, this is not very interesting. When  $\beta_\lambda > 3$ , we observe a slow runtime behavior similar to the one of the  $(1 + 1)$  EA with heavy-tailed mutation rate [19]. This is not surprising since with this distribution of  $\lambda$  typically only small values of  $\lambda$  are sampled. We profit most from the strength of the heavy-tailed  $(1 + (\lambda, \lambda))$  GA when  $\beta_\lambda$  is close to two. If  $\beta_\lambda$  is larger than two, then each iteration has an expected constant cost, so we can conveniently choose  $u_\lambda = \infty$  without that this has a negative effect on the runtime. This is a hyperparameter setting we would recommend as a first, low-risk attempt to use this algorithm. Slightly better results are obtained from using  $\beta_\lambda = 2$ . Now a finite value for  $u_\lambda$  is necessary, but the logarithmic influence of  $u_\lambda$  on the runtime allows to be generous, e.g., taking  $u_\lambda$  exponential in  $n$ . Smaller values lead to

minimally better runtimes as long as one stays above the boundary  $(\frac{n}{k})^{k/2}$ , so optimizing here is risky.

*Distribution of  $s$ :* The distribution of  $s$  is less critical as long as  $u_s \geq 2k$ . Aiming at an algorithm free from critical parameter choices, we therefore recommend to take  $u_s = n$  unless there is a clear indication that only short moves in the search space are necessary. Once we decided on  $u_s = n$ , a  $\beta_s$  value below one is not interesting (apart from very particular situations). Depending on what jump sizes we expect to encounter, taking  $\beta_s = 1$  leading to an  $O(\log n)$ -factor contribution of  $s$  to the runtime or taking  $\beta_s = 1 + \varepsilon$ ,  $\varepsilon > 0$  but small, leading to an  $O(k^\varepsilon)$ -factor contribution to the runtime are both reasonable choices.

For reasons of space we only sketch the proof of Theorem 6.<sup>1</sup> We first estimate the probability  $P$  to make a jump into the global optimum in one iteration via Lemma 4. Then we estimate  $E[T_I] = P^{-1}$  and use Wald's equation (Lemma 5) to show that the expected number of fitness evaluations is  $E[T_F] = E[T_I]E[2\lambda]$ . Finally, we estimate  $E[\lambda]$  via Lemma 3.

### 3.2 Reaching the Local Optimum

In this section we show that the heavy-tailed choice of the parameters lets the  $(1 + (\lambda, \lambda))$  GA reach the local optimum relatively fast. Without proof, we note that if  $\beta_\lambda \geq 2$  and  $k \geq 3$ , then the time to reach the local optimum is not larger than the time to go from the local to the global optimum. For a set of hyperparameters giving the best price for instance-independence, we now show an  $O(n^2 \log^2(n))$  time bound for reaching the local optimum.

**Theorem 7.** *Let  $u_\lambda = 2^{\Theta(n)}$ ,  $\beta_\lambda = 2$ ,  $u_s = \Theta(n)$ , and  $\beta_s = 1$ . Then the expected runtime until the heavy-tailed  $(1 + (\lambda, \lambda))$  GA reaches the local optimum of  $\text{JUMP}_k$  starting in a random string is at most  $O(n^2 \log^2(n))$  fitness evaluations. For larger  $\beta_\lambda$  and any  $u_\lambda$  this runtime is at most  $O(n \log^2(n))$ . In both cases with  $\beta_s > 1$  and any  $u_s \in \mathbb{N}$  the runtime is reduced by a  $\Theta(\log(n))$  factor.*

For reasons of space we only sketch the proof for the hyperparameters  $\beta_\lambda = 2$  and  $\beta_s = 1$  (for other hyperparameters the arguments are similar). Via Lemma 2 we show that the probability to choose  $\lambda = s = 1$  is  $\Theta(1/\log(n))$ . In this case the  $(1 + (\lambda, \lambda))$  GA behaves as the  $(1 + 1)$  EA and finds the local optimum in  $O(n \log(n))$  iterations with this parameter choice. Taking into account the expected cost of one iteration, which is  $\Theta(n)$  by Lemma 3, we obtain a total runtime of  $\Theta(n^2 \log^2(n))$ .

## 4 Static Parameters

In [5] it was shown that the  $(1 + (\lambda, \lambda))$  GA can solve  $\text{JUMP}_k$  in  $(\frac{n}{k})^{k/2} e^{O(k)}$  fitness evaluations when it starts in the local optimum. This is, if we ignore  $e^{O(k)}$

<sup>1</sup> The omitted proofs can be found in the preprint [3].

factors, the square root of the runtime of the best mutation-based algorithms [19]. However, such an upper bound was obtained only by setting the parameters of the algorithm to values which depend on the jump size  $k$ . In this section we show that a deviation from these instance-specific optimal parameter settings significantly increases the runtime. The consequence is that when the parameter  $k$  is unknown, we are not likely to choose a good static parameter setting.

To analyze the negative effect of a wrong parameter choice we use the precise expression of the probability  $P$  to go from the local to the global optimum in one iteration, which is

$$P = \sum_{\ell=0}^n p_{\ell} p_m(\ell) p_c(\ell), \tag{1}$$

where  $p_{\ell}$  is the probability to choose  $\ell$  bits to flip,  $p_m(\ell)$  is the probability of a successful mutation phase conditional on the chosen  $\ell$ , and  $p_c(\ell)$  is the probability of a successful crossover phase conditional on the chosen  $\ell$  and on the mutation being successful.

Since  $\ell \sim \text{Bin}(n, p)$ , we have  $p_{\ell} = \binom{n}{\ell} p^{\ell} (1-p)^{n-\ell}$ . The probability of a successful mutation depends on the chosen  $\ell$ . If  $\ell < k$ , then it is impossible to flip all  $k$  zero-bits, hence  $p_m(\ell) = 0$ . For larger  $\ell$  the probability to create a good offspring in a single application of the mutation operator is  $q_m(\ell) = \binom{n-k}{\ell-k} / \binom{n}{\ell}$ . If  $\ell \in [k + 1, 2k - 1]$  then any good offspring occurs in the fitness valley and has a worse fitness than any other offspring that is not good. Hence, in order to have a successful mutation we need all  $\lambda_m$  offspring to be good. Therefore, the probability of a successful mutation is  $(q_m(\ell))^{\lambda_m}$ . For  $\ell = k$  and  $\ell \geq 2k$  we are guaranteed to choose a good offspring as the winner of the mutation phase if there is at least one. Therefore, the mutation phase is successful with probability  $p_m(\ell) = 1 - (1 - q_m(\ell))^{\lambda_m}$ .

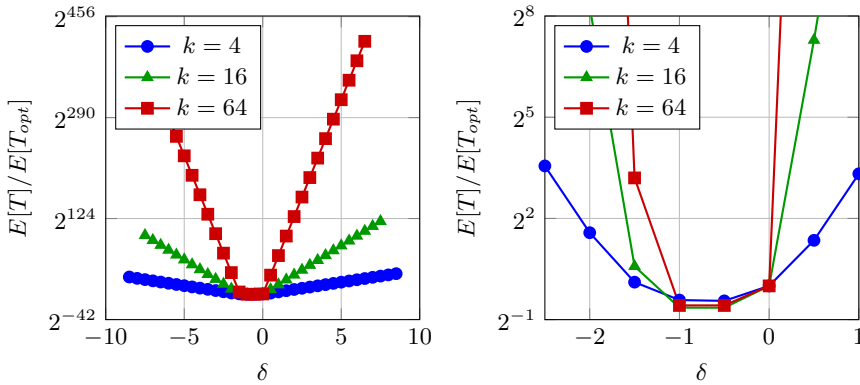
In the crossover phase we can create a good offspring only if  $\ell \geq k$ , and then we need to take all  $k$  bits which are zero in  $x$  from  $x'$  and take all  $\ell - k$  one-bits which were flipped from  $x$ . The probability to do so in one offspring is  $q_c(\ell) = c^k (1-c)^{\ell-k}$ . Since we create  $\lambda_c$  offspring and at least one of them must be superior to  $x$ , the probability of a successful crossover phase is  $p_c(\ell) = 1 - (1 - c^k (1-c)^{\ell-k})^{\lambda_c}$ .

Putting these probabilities into (1) we obtain

$$\begin{aligned} P &= \binom{n}{k} p^k (1-p)^{n-k} \left( 1 - \left( 1 - \binom{n}{k}^{-1} \right)^{\lambda_m} \right) (1 - (1 - c^k)^{\lambda_c}) \\ &+ \sum_{\ell=k+1}^{2k-1} \binom{n}{\ell} p^{\ell} (1-p)^{n-\ell} \left( \frac{\binom{n-k}{\ell-k}}{\binom{n}{\ell}} \right)^{\lambda_m} (1 - (1 - c^k (1-c)^{\ell-k})^{\lambda_c}) \\ &+ \sum_{\ell=2k}^n \binom{n}{\ell} p^{\ell} (1-p)^{n-\ell} \left( 1 - \left( 1 - \frac{\binom{n-k}{\ell-k}}{\binom{n}{\ell}} \right)^{\lambda_m} \right) (1 - (1 - c^k (1-c)^{\ell-k})^{\lambda_c}). \end{aligned}$$

Via this expression for  $P$  we compute the expected runtime in terms of iterations as  $E[T_I] = P^{-1}$  and the expected runtime in terms of fitness evaluations as  $E[T_F] = (\lambda_m + \lambda_c)P^{-1}$ . It is hard estimate precisely the probability  $P$  and thus the expected runtime. Therefore, to show the critical influence of the parameters on the runtime, we compute  $E[T_F]$  precisely for  $n = 2^{20}$  and  $k \in \{2^2, 2^4, 2^6\}$  using different parameter values. We fix  $\lambda_m = \lambda_c = \sqrt{\frac{n}{k}}$  and take  $p = 2^\delta \sqrt{\frac{k}{n}}$  and  $c = 2^{-\delta} \sqrt{\frac{k}{n}}$  for all  $\delta \in [-\log_2(\sqrt{\frac{n}{k}}) .. \log_2(\sqrt{\frac{n}{k}})]$ ; this range for  $\delta$  guarantees that both  $p$  and  $c$  do not exceed 1. Note that we preserve the invariant  $pcn = k$ , since otherwise the expected Hamming distance between  $x$  and any crossover offspring (the search radius) is not  $k$ , which would make it even harder to find the global optimum. For  $\delta = 0$  these values were suggested in [5] (based on an asymptotic analysis, so constant factors were ignored). The results of this computation are shown in Fig. 2.

As one can see, there is a relatively small interval around  $\delta = 0$  in which the runtime is close to the one for  $\delta = 0$  (for  $\delta = -1$  the runtime is even slightly better), but generally the runtime increases by a  $\Theta(2^{|\delta|k})$  factor. Therefore, in order to solve  $JUMP_k$  effectively with the  $(1 + (\lambda, \lambda))$  GA, one has to guess the value of  $k$  relatively precisely to obtain a good performance from the static parameters suggested in [5]. In practice when we optimize a problem with local optima we usually cannot tell in advance the size of jump needed to escape the local optima. Therefore, the heavy-tailed parameter choice suggested in this work is likely to give better results than a static parameter choice.



**Fig. 2.** The ratio of the runtime with disturbed parameters to the runtime with the parameters suggested in [5]. The left plot shows the full picture for all considered values of  $\delta$ . The right plot shows in more detail a smaller interval around the best values.

## 5 Conclusion

In this work, we proposed a variant of the  $(1 + (\lambda, \lambda))$  GA with a heavy-tailed choice of both the population size  $\lambda$  and the search radius  $s$ . To the best of our knowledge, this is the first time that two parameters of an EA are chosen in this manner. Our mathematical runtime analysis showed that this algorithm with suitable, but natural choices of the distribution parameters can optimize all jump functions in a time that is only mildly higher than the runtime of the  $(1 + (\lambda, \lambda))$  GA with the best known instance-specific parameter values.

We are optimistic that the insights gained on the jump functions benchmark extend, at least to some degree, also to other non-unimodal problems. Clearly, supporting this hope with rigorous results is an interesting direction for future research. From a broader perspective, this work suggests to try to use heavy-tailed parameter choices for more than one parameter simultaneously. Our rigorous results indicate that the prices for ignorant (heavy-tailed) choices of parameters simply multiply. For a small number of parameters with critical influence on the performance, this might be a good deal.

**Acknowledgements.** This study was funded by RFBR and CNRS, project number 20-51-15009.

## References

1. Antipov, D., Buzdalov, M., Doerr, B.: Fast mutation in crossover-based algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 1268–1276. ACM (2020)
2. Antipov, D., Buzdalov, M., Doerr, B.: First steps towards a runtime analysis when starting with a good solution. In: Bäck, T., et al. (eds.) PPSN 2020. LNCS, vol. 12270, pp. 560–573. Springer, Switzerland (2020). [https://doi.org/10.1007/978-3-030-58115-2\\_39](https://doi.org/10.1007/978-3-030-58115-2_39)
3. Antipov, D., Doerr, B.: Runtime analysis of a heavy-tailed  $(1+(\lambda, \lambda))$  genetic algorithm on jump functions. CoRR abs/2006.03523 (2020). <https://arxiv.org/abs/2006.03523>
4. Antipov, D., Doerr, B., Karavaev, V.: A tight runtime analysis for the  $(1 + (\lambda, \lambda))$  GA on LeadingOnes. In: Foundations of Genetic Algorithms, FOGA 2019, pp. 169–182. ACM (2019)
5. Antipov, D., Doerr, B., Karavaev, V.: The  $(1 + (\lambda, \lambda))$  GA is even faster on multi-modal problems. In: Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 1259–1267. ACM (2020)
6. Buzdalov, M., Doerr, B.: Runtime analysis of the  $(1 + (\lambda, \lambda))$  genetic algorithm on random satisfiable 3-CNF formulas. In: Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 1343–1350. ACM (2017)
7. Dang, D.-C., Friedrich, T., Kötzing, T., Krejca, M.S., Lehre, P.K., Oliveto, P.S., Sudholt, D., Sutton, A.M.: Escaping local optima with diversity mechanisms and crossover. In: Genetic and Evolutionary Computation Conference, GECCO 2016, pp. 645–652. ACM (2016)

8. Dang, D.-C., Friedrich, T., Kötzing, T., Krejca, M.S., Lehre, P.K., Oliveto, P.S., Sudholt, D., Sutton, A.M.: Escaping local optima using crossover with emergent diversity. *IEEE Trans. Evol. Comput.* **22**, 484–497 (2018)
9. 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)
10. Doerr, B.: A tight runtime analysis for the cGA on jump functions: EDAs can cross fitness valleys at no extra cost. In: Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 1488–1496. ACM (2019)
11. Doerr, B.: Does comma selection help to cope with local optima? In: Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 1304–1313. ACM (2020)
12. Doerr, B., Doerr, C.: Optimal static and self-adjusting parameter choices for the  $(1 + (\lambda, \lambda))$  genetic algorithm. *Algorithmica* **80**, 1658–1709 (2018)
13. Doerr, B., Doerr, C.: Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices. In: Doerr, B., Neumann, F. (eds.) *Theory of Evolutionary Computation*. NCS, pp. 271–321. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-29414-4\\_6](https://doi.org/10.1007/978-3-030-29414-4_6). <https://arxiv.org/abs/1804.05650>
14. Doerr, B., Doerr, C., Ebel, F.: Lessons from the black-box: fast crossover-based genetic algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2013, pp. 781–788. ACM (2013)
15. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. *Theoret. Comput. Sci.* **567**, 87–104 (2015)
16. Doerr, B., Doerr, C., Kötzing, T.: Static and self-adjusting mutation strengths for multi-valued decision variables. *Algorithmica* **80**, 1732–1768 (2018)
17. Doerr, B., Doerr, C., Yang, J.:  $k$ -bit mutation with self-adjusting  $k$  outperforms standard bit mutation. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 824–834. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45823-6\\_77](https://doi.org/10.1007/978-3-319-45823-6_77)
18. Doerr, B., Gießen, C., Witt, C., Yang, J.: The  $(1 + \lambda)$  evolutionary algorithm with self-adjusting mutation rate. *Algorithmica* **81**, 593–631 (2019)
19. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 777–784. ACM (2017)
20. Doerr, B., Witt, C., Yang, J.: Runtime analysis for self-adaptive mutation rates. In: Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 1475–1482. ACM (2018)
21. Droste, S., Jansen, T., Wegener, I.: On the analysis of the  $(1+1)$  evolutionary algorithm. *Theoret. Comput. Sci.* **276**, 51–81 (2002)
22. Friedrich, T., Göbel, A., Quinzan, F., Wagner, M.: Evolutionary algorithms and submodular functions: benefits of heavy-tailed mutations. *CoRR* abs/1805.10902 (2018)
23. Friedrich, T., Göbel, A., Quinzan, F., Wagner, M.: Heavy-tailed mutation operators in single-objective combinatorial optimization. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11101, pp. 134–145. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99253-2\\_11](https://doi.org/10.1007/978-3-319-99253-2_11)
24. Friedrich, T., Kötzing, T., Krejca, M.S., Nallaperuma, S., Neumann, F., Schirneck, M.: Fast building block assembly by majority vote crossover. In: Genetic and Evolutionary Computation Conference, GECCO 2016, pp. 661–668. ACM (2016)

25. Friedrich, T., Quinzan, F., Wagner, M.: Escaping large deceptive basins of attraction with heavy-tailed mutation operators. In: Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 293–300. ACM (2018)
26. Goldman, B.W., Punch, W.F.: Parameter-less population pyramid. In: Genetic and Evolutionary Computation Conference, GECCO 2014, pp. 785–792. ACM (2014)
27. Hasenöhr, V., Sutton, A.M.: On the runtime dynamics of the compact genetic algorithm on jump functions. In: Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 967–974. ACM (2018)
28. Jansen, T., Wegener, I.: The analysis of evolutionary algorithms - a proof that crossover really can help. *Algorithmica* **34**, 47–66 (2002)
29. Lässig, J., Sudholt, D.: Adaptive population models for offspring populations and parallel evolutionary algorithms. In: Foundations of Genetic Algorithms, FOGA 2011, pp. 181–192. ACM (2011)
30. Mambrini, A., Sudholt, D.: Design and analysis of schemes for adapting migration intervals in parallel evolutionary algorithms. *Evol. Comput.* **23**, 559–582 (2015)
31. Mironovich, V., Buzdalov, M.: Evaluation of heavy-tailed mutation operator on maximum flow test generation problem. In: Genetic and Evolutionary Computation Conference, GECCO 2017. Companion Material, pp. 1423–1426. ACM (2017)
32. Rowe, J.E., Aishwaryaprajna: the benefits and limitations of voting mechanisms in evolutionary optimisation. In: Foundations of Genetic Algorithms, FOGA 2019, pp. 34–42. ACM (2019)
33. Wald, A.: Some generalizations of the theory of cumulative sums of random variables. *Ann. Math. Stat.* **16**, 287–293 (1945)
34. Whitley, D., Varadarajan, S., Hirsch, R., Mukhopadhyay, A.: Exploration and exploitation without mutation: solving the jump function in  $\Theta(n)$  time. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 55–66. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_5](https://doi.org/10.1007/978-3-319-99259-4_5)
35. Wu, M., Qian, C., Tang, K.: Dynamic mutation based pareto optimization for subset selection. In: Huang, D.-S., Gromiha, M.M., Han, K., Hussain, A. (eds.) ICIC 2018. LNCS (LNAI), vol. 10956, pp. 25–35. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-95957-3\\_4](https://doi.org/10.1007/978-3-319-95957-3_4)
36. Ye, F., Wang, H., Doerr, C., Bäck, T.: Benchmarking a  $(\mu + \lambda)$  genetic algorithm with configurable crossover probability. *CoRR* abs/2006.05889 (2020)



# First Steps Towards a Runtime Analysis When Starting with a Good Solution

Denis Antipov<sup>1,2(✉)</sup>, Maxim Buzdalov<sup>1</sup>, and Benjamin Doerr<sup>2</sup>

<sup>1</sup> ITMO University, St. Petersburg, Russia  
[antipovden@yandex.ru](mailto:antipovden@yandex.ru)

<sup>2</sup> Laboratoire d'Informatique (LIX), CNRS, École Polytechnique,  
Institut Polytechnique de Paris, Palaiseau, France

**Abstract.** The mathematical runtime analysis of evolutionary algorithms traditionally regards the time an algorithm needs to find a solution of a certain quality when initialized with a random population. In practical applications it may be possible to guess solutions that are better than random ones. We start a mathematical runtime analysis for such situations. We observe that different algorithms profit to a very different degree from a better initialization. We also show that the optimal parameterization of the algorithm can depend strongly on the quality of the initial solutions. To overcome this difficulty, self-adjusting and randomized heavy-tailed parameter choices can be profitable. Finally, we observe a larger gap between the performance of the best evolutionary algorithm we found and the corresponding black-box complexity. This could suggest that evolutionary algorithms better exploiting good initial solutions are still to be found. These first findings stem from analyzing the performance of the  $(1 + 1)$  evolutionary algorithm and the static, self-adjusting, and heavy-tailed  $(1 + (\lambda, \lambda))$  GA on the OneMax benchmark, but we are optimistic that the question how to profit from good initial solutions is interesting beyond these first examples.

**Keywords:** Theory · Runtime analysis · Initialization of evolutionary algorithms · Crossover · Fast mutation

## 1 Introduction

The mathematical runtime analysis (see, e.g., [4, 15, 20, 28]) has contributed to our understanding of evolutionary algorithms (EAs) via rigorous analyses how long an EA takes to optimize a particular problem. The overwhelming majority of these results considers a random or worst-case initialization of the algorithm. In this work, we argue that it also makes sense to analyze the runtime of algorithms starting already with good solutions. This is justified because such situations arise in practice and because, as we observe in this work, different algorithms show a different runtime behavior when started with such good solutions. In particular, we observe that the  $(1 + (\lambda, \lambda))$  genetic algorithm ( $(1 + (\lambda, \lambda))$  GA)



profits from good initial solutions by much more than, e.g., the  $(1+1)$  EA. From a broader perspective, this work suggests that the recently proposed fine-grained runtime notions like fixed budget analysis [22] and fixed target analysis [6], which consider optimization up to a certain solution quality, should be extended to also take into account different initial solution qualities.

### 1.1 Starting with Good Solutions

As just said, the vast majority of the runtime analyses assume a random initialization of the algorithm or they prove performance guarantees that hold for all initializations (worst-case view). This is justified for two reasons. (i) When optimizing a novel problem for which little problem-specific understanding is available, starting with random initial solutions is a recommended approach. This avoids that a wrong understanding of the problem leads to an unfavorable initialization. Also, with independent runs of the algorithm automatically reasonably diverse initializations are employed. (ii) For many optimizations processes analyzed with mathematical means it turned out that there is not much advantage of starting with a good solution. For this reason, such results are not stated explicitly, but can often be derived from the proofs. For example, when optimizing the simple ONEMAX benchmark via the equally simple  $(1+1)$  EA, then results like [10, 11, 16, 26] show a very limited advantage from a good initialization. When starting with a solution having already 99% of the maximal fitness, the expected runtime has the same  $en \ln(n) \pm O(n)$  order of magnitude. Hence the gain from starting with the good solution is bounded by an  $O(n)$  lower order term. Even when starting with a solution of fitness  $n - \sqrt{n}$ , that is, with fitness distance  $\sqrt{n}$  to the optimum of fitness  $n$ , then only a runtime reduction by asymptotically a factor of a half results. Clearly, a factor-two runtime improvement is interesting in practice, but the assumption that an initial solution can be found that differs from the optimum in only  $\sqrt{n}$  of the  $n$  bit positions, is very optimistic.

Besides this justification for random initializations, we see a number of situations in which better-than-random solutions are available (and this is the motivation of this work). The obvious one is that a problem is to be solved for which some, at least intuitive, understanding is available. This is a realistic assumption in scenarios where similar problems are to be solved over a longer time period or where problems are solved by combining a human understanding of the problem with randomized heuristics. A second situation in which we expect to start with a good solution is reoptimization. Reoptimization [30, 34] means that we had already solved a problem, then a mild change of the problem data arises (due to a change in the environment, a customer being unhappy with a particular aspect of the solution, etc.), and we react to this change not by optimizing the new problem from scratch, but by initializing the EA with solutions that were good in the original problem. While there is a decent amount of runtime analysis literature on how EAs cope with dynamic optimization problems, see [27], almost all of them regard the situation that a dynamic change of the instance happens frequently and the question is how well the EA adjusts to

these changes. The only mathematical runtime analysis of a true reoptimization problem we are aware of is [9]. The focus there, however, is to modify an existing algorithm so that it better copes with the situation that the algorithm is started with a solution that is structurally close to the optimum, but has a low fitness obscuring to the algorithm that the current solution is already structurally good.

We note that using a known good solution to initialize a randomized search heuristic is again a heuristic approach. It is intuitive that an iterative optimization heuristic can profit from such an initialization, but there is no guarantee and, clearly, there are also situations where using such initializations is detrimental. As one example, assume that we obtain good initial solutions from running a simple hill-climber. Then these initial solutions could be local optima which are very hard to leave. An evolutionary algorithm initialized with random solutions might find it easier to generate a sufficient diversity that allows to reach the basin of attraction of the optimum. So obviously some care is necessary when initializing a search heuristic with good solutions. Several practical applications of evolutionary algorithms have shown advantages of initializations with good solutions, e.g., [24] on the open shop scheduling problem.

While there are no explicit mathematical runtime analyses for EAs starting with a good solution, it is clear that many of the classic results in their proofs reveal much information also on runtimes starting from a good solution. This is immediately clear for the fitness level method [32], but also for drift arguments like [12, 19, 23, 25] when as potential function the fitness or a similar function is used, and for many other results. By not making these results explicit, however, it is hard to see the full picture and to draw the right conclusions.

## 1.2 The $(1 + (\lambda, \lambda))$ GA Starting with Good Solutions

In this work, we make explicit how the  $(1 + (\lambda, \lambda))$  GA optimizes ONEMAX when starting from a solution with fitness distance  $D$  from the optimum. We observe that the  $(1 + (\lambda, \lambda))$  GA profits in a much stronger way from such a good initialization than other known algorithms. For example, when starting in fitness distance  $D = \sqrt{n}$ , the expected time to find the optimum is only  $\tilde{O}(n^{3/4})$  when using optimal parameters. We recall that this algorithm has a runtime of roughly  $n\sqrt{\log n}$  when starting with a random solution [7, 8]. We recall further that the  $(1+1)$  EA has an expected runtime of  $(1 \pm o(1))\frac{1}{2}en \ln(n)$  when starting in fitness distance  $\sqrt{n}$  and an expected runtime of  $(1 \pm o(1))en \ln n$  when starting with a random solution. So clearly, the  $(1 + (\lambda, \lambda))$  GA profit to a much higher degree from a good initialization than the  $(1+1)$  EA. We made this precise for the  $(1+1)$  EA, but it is clear from other works such as [3, 13, 21, 33] that similar statements hold as well for many other  $(\mu + \lambda)$  EAs optimizing ONEMAX, at least for some ranges of the parameters.

The runtime stated above for the  $(1 + (\lambda, \lambda))$  GA assumes that the algorithm is used with the optimal parameter setting, more precisely, with the optimal setting for starting with a solution of fitness-distance  $D$ . Besides that we usually do not expect the algorithm user to guess the optimal parameter values, it is also not very realistic to assume that the user has a clear picture on how far the initial

solution is from the optimum. For that reason, we also regard two parameterless variants of the  $(1 + (\lambda, \lambda))$  GA (where parameterless means that parameters with a crucial influence on the performance are replaced by hyperparameters for which the influence is less critical or for which we can give reasonable general rules of thumb).

Already in [8], a self-adjusting choice based on the one-fifth success rule of the parameters of the  $(1 + (\lambda, \lambda))$  GA was proposed. This was shown to give a linear runtime on ONEMAX in [7]. We note that this is, essentially, a parameterless algorithm since the target success rate (the “one-fifth”) and the update factor had only a small influence on the result provided that they were chosen not too large (where the algorithm badly fails). See [7, Sect. 6.4] for more details. For this algorithm, we show that it optimizes ONEMAX in time  $O(\sqrt{nD})$  when starting in distance  $D$ . Again, this is a parameterless approach (when taking the previous recommendations on how to set the hyperparameters).

A second parameterless approach for the  $(1 + (\lambda, \lambda))$  GA was recently analyzed in [1], namely to choose the parameter  $\lambda$  randomly from a power-law distribution. Such a heavy-tailed parameter choice was shown to give a performance only slightly below the one obtainable from the best instance-specific values for the  $(1 + 1)$  EA optimizing jump functions [14]. Surprisingly, the  $(1 + (\lambda, \lambda))$  GA with heavy-tailed parameter choice could not only overcome the need to specify parameter values, it even outperformed any static parameter choice and had the same  $O(n)$  runtime that the self-adjusting  $(1 + (\lambda, \lambda))$  GA had [1]. When starting with a solution in fitness distance  $D$ , this algorithm with any power-law exponent equal to or slightly above two gives a performance which is only by a small factor slower than  $O(\sqrt{nD})$ .

### 1.3 Experimental Results

We support our theoretical findings with an experimental validation, which shows that both the self-adjusting and the heavy-tailed version of the  $(1 + (\lambda, \lambda))$  GA indeed show the desired asymptotic behavior and this with only moderate implicit constants. In particular, the one-fifth self-adjusting version can be seen as a very confident winner in all cases, and the heavy-tailed versions with different power-law exponents follow it with the accordingly distributed runtimes. Interestingly enough, the logarithmically-capped self-adjusting version, which has been shown to be beneficial for certain problems other than ONEMAX [5] and just a tiny bit worse than the basic one-fifth version on ONEMAX, starts losing ground to the heavy-tailed versions at distances just slightly smaller than  $\sqrt{n}$ .

### 1.4 Black-Box Complexity and Lower Bounds

The results above show that some algorithms can profit considerably from good initial solutions (but many do not). This raises the question of how far we can go in this direction, or formulated inversely, what lower bounds on this runtime problem we can provide. We shall not go much into detail on this question, but

note here that one can define a black-box complexity notion for this problem. Informally speaking, the input to this problem is an objective function from a given class of functions and a search point in Hamming distance  $D$  from the optimum. The unrestricted black-box complexity is the smallest expected number of fitness evaluations that an otherwise unrestricted black-box algorithm performs to find the optimum (of a worst-case input).

If the class of functions consists of all ONEMAX-type functions, that is, ONEMAX and all functions with an isomorphic fitness landscape, then the classic argument via randomized search trees and Yao's minimax principle from [17]<sup>1</sup> shows that the black-box complexity is at least  $\Omega(\frac{D \log(n/D)}{\log n})$ . A matching upper bound follows from evaluating random search points until all evaluation results leave only one solution (out of the originally  $\binom{n}{D}$  ones) fitting to the evaluations results (this is the classic random guessing strategy of [18]). For small  $D$ , this black-box complexity of order  $\Theta(\frac{D \log(n/D)}{\log n})$  is considerably lower than our upper bounds. Also, this shows a much larger gap between black-box complexity and EA performance than in the case of random initialization, where the black-box complexity is  $\Theta(\frac{n}{\log n})$  and simple EAs have an  $O(n \log n)$  performance.

## 1.5 Synopsis and Structure of the Paper

Overall, our results show that the question of how EAs work when started with a good initial solution is far from trivial. Some algorithms profit more from this than others, the question of how to set the parameters might be influenced by the starting level  $D$  and this may make parameterless approaches more important, and the larger gap to the black-box complexity could suggest that there is room for further improvements.

The rest of the paper is organized as follows. In Sect. 2 we formally define the considered algorithms and the problem and collect some useful analysis tools. In Sect. 3 we prove the upper bounds on the runtime of the algorithms and deliver general recommendations on how to use each algorithm. In Sect. 4 we check how our recommendations work in experiments.

## 2 Preliminaries

### 2.1 The $(1 + (\lambda, \lambda))$ GA and Its Modifiactions

We consider the  $(1 + (\lambda, \lambda))$  GA, which is a genetic algorithm for the optimization of  $n$ -dimensional pseudo-Boolean functions, first proposed in [8]. This algorithm has three parameters, which are the mutation rate  $p$ , the crossover bias  $c$ , and the population size  $\lambda$ .

<sup>1</sup> This argument can be seen as a formalization of the intuitive argument that there are  $\binom{n}{D}$  different solution candidates, each fitness evaluation has up to  $n + 1$  different answers, hence if the runtime is less than  $\log_{n+1} \binom{n}{D}$  then there are two solution candidates that receive the same sequence of answers and hence are indistinguishable.

The  $(1 + (\lambda, \lambda))$  GA stores the current individual  $x$ , which is initialized with a random bit string. Each iteration of the algorithm consists of a mutation phase and a crossover phase. In the mutation phase we first choose a number  $\ell$  from the binomial distribution with parameters  $n$  and  $p$ . Then we create  $\lambda$  offsprings by flipping  $\ell$  random bits in  $x$ , independently for each offspring. An offspring with the best fitness is chosen as the mutation winner  $x'$  (all ties are broken uniformly at random). Note that  $x'$  can and often will have a worse fitness than  $x$ .

In the crossover phase we create  $\lambda$  offspring by applying a biased crossover to  $x$  and  $x'$  (independently for each offspring). This biased crossover takes each bit from  $x$  with probability  $(1 - c)$  and from  $x'$  with probability  $c$ . A crossover offspring with best fitness is selected as the crossover winner  $y$  (all ties are broken uniformly at random). If  $y$  is not worse than  $x$ , it replaces the current individual. The pseudocode of the  $(1 + (\lambda, \lambda))$  GA is shown in Algorithm 1.

---

**Algorithm 1:** The  $(1 + (\lambda, \lambda))$  GA maximizing a pseudo-Boolean function  $f$ .

---

```

1  $x \leftarrow$  random bit string of length  $n$ ;
2 while not terminated do
3   Mutation phase:
4   Choose  $\ell \sim \text{Bin}(n, p)$ ;
5   for  $i \in [1.. \lambda]$  do
6      $x^{(i)} \leftarrow$  a copy of  $x$ ;
7     Flip  $\ell$  bits in  $x^{(i)}$  chosen uniformly at random;
8   end
9    $x' \leftarrow \arg \max_{z \in \{x^{(1)}, \dots, x^{(\lambda)}\}} f(z)$ ;
10  Crossover phase:
11  for  $i \in [1.. \lambda]$  do
12    Create  $y^{(i)}$  by taking each bit from  $x'$  with probability  $c$  and from  $x$ 
    with probability  $(1 - c)$ ;
13  end
14   $y \leftarrow \arg \max_{z \in \{y^{(1)}, \dots, y^{(\lambda)}\}} f(z)$ ;
15  if  $f(y) \geq f(x)$  then
16     $x \leftarrow y$ ;
17  end
18 end

```

---

Based on intuitive considerations and rigorous runtime analyses, a standard parameter settings was proposed in which the mutation rate and crossover bias are defined via the population size, namely,  $p = \frac{\lambda}{n}$  and  $c = \frac{1}{\lambda}$ .

It was shown in [8] that with a suitable **static parameter** value for  $\lambda$ , this algorithm can solve the ONEMAX function in  $O(n\sqrt{\log(n)})$  fitness evaluations (this bound was minimally reduced and complemented with a matching lower bound in [7]). The authors of [8] noticed that with the **fitness-dependent parameter**  $\lambda = \sqrt{\frac{n}{d}}$  the algorithm solves ONEMAX in only  $\Theta(n)$  iterations.

The fitness-depending parameter setting was not satisfying, since it is too problem-specific and most probably does not work on practical problems. For this reason, also a **self-adjusting parameter choice** for  $\lambda$  was proposed in [8] and analyzed rigorously in [7]. It uses a simple one-fifth rule, multiplying the parameter  $\lambda$  by some constant  $A > 1$  at the end of the iteration when  $f(y) \leq f(x)$ , and dividing  $\lambda$  by  $A^4$  otherwise (the fourth power ensures the desired property that the parameter does not change in the long run when in average one fifth of the iterations are successful). This simple rule was shown to keep the parameter  $\lambda$  close to the optimal fitness-dependent value during the whole optimization process, leading to a  $\Theta(n)$  runtime on ONEMAX. However, this method of parameter control was not efficient on the MAX-3SAT problem, which has a lower fitness-distance correlation than ONEMAX [5]. Therefore, capping the maximal value of  $\lambda$  at  $2 \ln(n + 1)$  was needed to obtain a good performance on this problem.

Inspired by [14], the recent paper [1] proposed use a heavy-tailed random  $\lambda$ , which gave a birth to the **fast**  $(1 + (\lambda, \lambda))$  **GA**. In this algorithm the parameter  $\lambda$  is chosen from the power-law distribution with exponent  $\beta$  and with upper limit  $u$ . Here for all  $i \in \mathbb{N}$  we have

$$\Pr[\lambda = i] = \begin{cases} C_{\beta,u} i^{-\beta}, & \text{if } i \in [1..u], \\ 0, & \text{otherwise,} \end{cases}$$

where  $C_{\beta,u} = (\sum_{j=1}^u j^{-\beta})^{-1}$  is the normalization coefficient. It was proven that the fast  $(1 + (\lambda, \lambda))$  GA finds the optimum of ONEMAX in  $\Theta(n)$  fitness evaluations if  $\beta \in (2, 3)$  and  $u$  is large enough. Also it was empirically shown that this algorithm without further capping of  $\lambda$  is quite efficient on MAX-3SAT.

When talking about the runtime of the  $(1 + (\lambda, \lambda))$  GA, we denote the number of iterations until the optimum is found by  $T_I$  and the number of fitness evaluations until the optimum is found by  $T_F$ . We denote the distance of the current individual to the optimum by  $d$ .

### 2.2 Problem Statement

The main object of this paper is the runtime of the algorithms discussed in Sect. 2.1 when they start in distance  $D$  from the optimum, where  $D$  should be smaller than the distance of a random solution. For this purpose we consider the classic ONEMAX function, which is defined on the space of bit strings of length  $n$  by

$$\text{ONEMAX}(x) = \text{OM}(x) = \sum_{i=1}^n x_i.$$

### 2.3 Probability for Progress

To prove our upper bounds on the runtimes we use the following estimate for the probability that the  $(1 + (\lambda, \lambda))$  GA finds a better solution in one iteration.

**Lemma 1.** *The probability that  $OM(y) > OM(x)$  is  $\Omega(\min\{1, \frac{d\lambda^2}{n}\})$ .*

To prove this lemma we use the following auxiliary result from [1], a slight adaptation of [29, Lemma 8].

**Lemma 2 (Lemma 2.2 in [1]).** *For all  $p \in [0, 1]$  and all  $\lambda > 0$  we have*

$$1 - (1 - p)^\lambda \geq \frac{\lambda p}{1 + \lambda p}.$$

*Proof (of Lemma 1).* By Lemma 7 in [8] the probability to have a true progress in one iteration is  $\Omega(1 - (\frac{n-d}{n})^{\frac{\lambda^2}{2}})$ . By Lemma 2 this is at least  $\Omega(\min\{1, \frac{d\lambda^2}{n}\})$ .

### 3 Runtime Analysis

In this section we conduct a rigorous runtime analysis for the different variants of the  $(1 + (\lambda, \lambda))$  GA and prove upper bounds on their runtime when they start in distance  $D$  from the optimum. We start with the standard algorithm with static parameters.

**Theorem 3.** *The expected runtime of the  $(1 + (\lambda, \lambda))$  GA with static parameter  $\lambda$  (and mutation rate  $p = \frac{\lambda}{n}$  and crossover bias  $c = \frac{1}{\lambda}$  as recommended in [8]) on ONEMAX with initialization in distance  $D$  from the optimum is*

$$E[T_F] = O\left(\frac{n}{\lambda} \ln\left(\frac{n}{\lambda^2}\right) + D\lambda\right)$$

*fitness evaluations. This is minimized by  $\lambda = \sqrt{\frac{n \ln(D)}{D}}$ , which gives a runtime guarantee of  $E[T_F] = O(\sqrt{nD \ln(D)})$ .*

We omit the proof for reasons of space<sup>2</sup>. We move on to the  $(1 + (\lambda, \lambda))$  GA with optimal fitness-dependent parameters.

**Theorem 4.** *The expected runtime of the  $(1 + (\lambda, \lambda))$  GA with fitness-dependent  $\lambda = \lambda(d) = \sqrt{\frac{n}{d}}$  on ONEMAX with initialization in distance  $D$  from the optimum is  $E[T_F] = O(\sqrt{nD})$ .*

We omit the proof for reasons of space and since it trivially follows from Lemma 1.

The one-fifth rule was shown to be to keep the value of  $\lambda$  close to its optimal fitness-dependent value, when starting in the random bit string. The algorithm is initialized with  $\lambda = 2$ , which is close-to-optimal when starting in a random bit string. In the following theorem we show that even when we start in a smaller distance  $D$ , the one-fifth rule is capable to quickly increase  $\lambda$  to its optimal value and keep it there.

---

<sup>2</sup> All the omitted proofs can be found in preprint [2].

**Theorem 5.** *The expected runtime of the  $(1 + (\lambda, \lambda))$  GA with self-adjusting  $\lambda$  (according to the one-fifth rule) on ONEMAX with initialization in distance  $D$  from the optimum is  $E[T_F] = O(\sqrt{nD})$ .*

We only sketch the proof for reasons of space. We first show that there is some distance  $d \leq D$  at which the algorithm reaches the optimal fitness-dependent value of  $\lambda$  for the first time. This happens in a relatively short time after the start of the algorithm. In a similar manner as in [7] we show that from that moment on the value of  $\lambda$  always stays close to the optimal fitness-dependent one, yielding asymptotically the same runtime.

For the fast  $(1 + (\lambda, \lambda))$  GA with different parameters of the power-law distribution, we show the following runtimes.

**Theorem 6.** *The expected runtime of the fast  $(1 + (\lambda, \lambda))$  GA on ONEMAX with initialization in distance  $D$  from the optimum is as shown in Table 1. The runtimes for  $\beta > 2$  hold also for all  $u \geq \sqrt{n}$ .*

We omit the proof for reasons of space, but sketch the main arguments. We deliver the upper bounds for the fast  $(1 + (\lambda, \lambda))$  GA in two steps. First we find an upper bound on the expected number of iterations  $E[T_I]$  of the algorithm in the same way as in Theorem 3.1 in [1]. Then we use Lemma 3.5 in the same paper to find the expected cost of one iteration, which is  $2E[\lambda]$ . Finally, by the Wald’s equation [31] we compute the expected number of iterations  $E[T_F] = 2E[\lambda]E[T_I]$ .

**Table 1.** Runtime of the heavy-tailed  $(1 + (\lambda, \lambda))$  GA for different ranges of  $\beta$  and for two variants of choosing  $u$ . The best possible fitness dependent choice of  $u = \sqrt{\frac{n}{d}}$  is given rather for reasons of comparison. The best fitness-independent choice is  $u = \sqrt{n}$ , but larger values of  $u$  are not harmful when  $\beta > 2$  (for  $\beta = 2$ , the  $\log n$  is actually a  $\log u$ , so the influence of  $u$  is small). Our recommendation when  $D$  is not known is to use  $\beta = 2$  and  $u = \sqrt{n}$ .

$\beta$	$E[T_F]$ with $u = \sqrt{\frac{n}{d}}$	$E[T_F]$ with $u = \sqrt{n}$
(0, 1)	$O(\sqrt{nD}\sqrt{\frac{n}{D}}^{1-\beta})$	$O(\sqrt{nD}\sqrt{D}^\beta n^{1-\beta})$
= 1	$O(\sqrt{nD}\log(\frac{n}{D}))$	$O(\sqrt{nD}\log(n))$
(1, 2)	$O(\sqrt{nD})$	$O(\sqrt{nD}\sqrt{D}^{2-\beta})$
= 2	$O(\sqrt{nD}\log(\frac{n}{D}))$	$O(\sqrt{nD}\log(n))$
(2, 3)	$O(\sqrt{nD}\sqrt{\frac{n}{D}}^{\beta-2})$	
= 3	$O(n\frac{\log(D)}{\log(n)})$	
> 3	$O(n\log(D))$	

From Table 1 we see that choosing  $\beta = 2$  and  $u = \sqrt{n}$  is the most universal option. The empirical results in [14] let us assume that different values of  $\beta$ , but



close to two might also be effective in practice. The results of our experiments provided in the Sect. 4 confirm this and show that using  $\beta < 2$  with  $u = \sqrt{n}$  can be beneficial when starting from a small distance.

## 4 Experiments

To highlight that the theoretically proven behavior of the algorithms is not strongly affected by the constants hidden in the asymptotic notation, we conducted experiments with the following settings:

- fast  $(1 + (\lambda, \lambda))$  GA with  $\beta \in \{2.1, 2.3, 2.5, 2.7, 2.9\}$  and the upper limit  $u = n/2$ ;
- self-adjusting  $(1 + (\lambda, \lambda))$  GA, both in its original uncapped form and with  $\lambda$  capped from above by  $2 \log(n + 1)$  as proposed in [5];
- the mutation-only algorithms  $(1 + 1)$  EA and RLS.

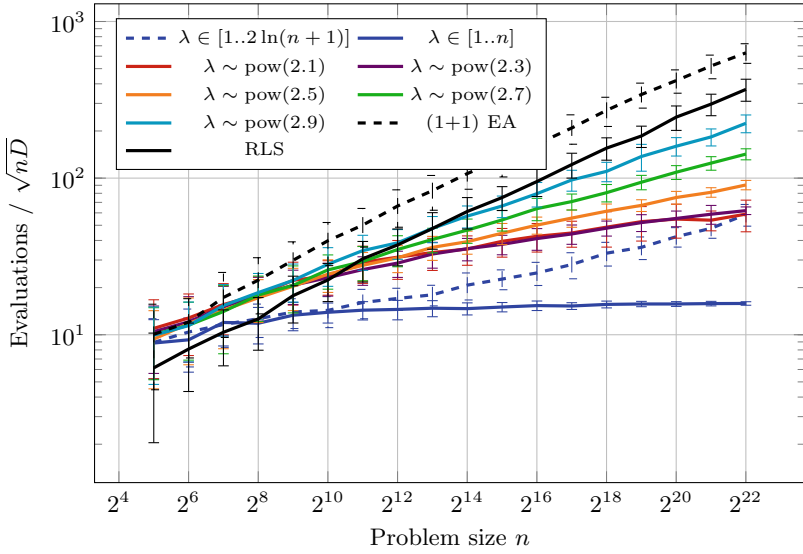
In all our experiments, the runtimes are averaged over 100 runs, unless said otherwise.

In Fig. 1 we show the mean running times of these algorithms when they start in Hamming distance roughly  $\sqrt{n}$  from the optimum. For this experiment, to avoid possible strange effects from particular numbers, we used a different initialization for all algorithms, namely that in the initial individual every bit was set to 0 with probability  $\frac{1}{\sqrt{n}}$  and it was set to 1 otherwise. As the figure shows, all algorithms with a heavy-tailed choice of  $\lambda$  outperformed the mutation-based algorithms, which struggled from the coupon-collector effect.

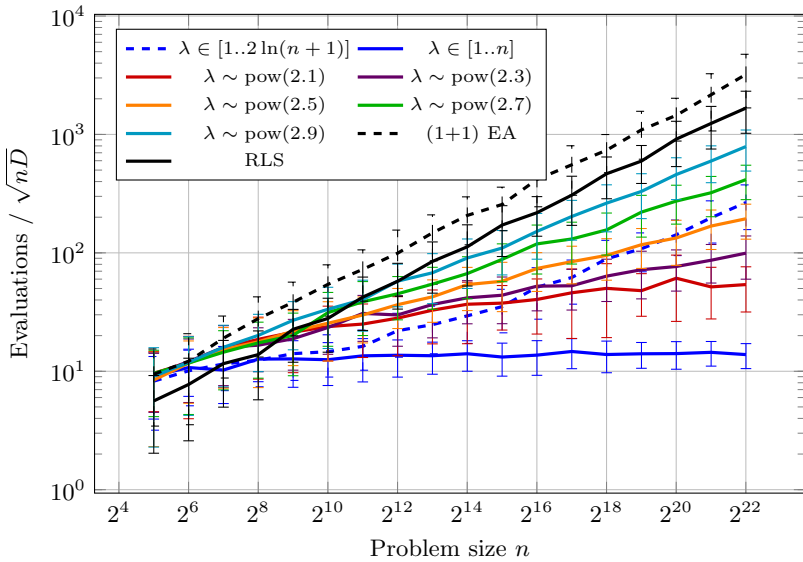
We can also see that the logarithmically capped self-adjusting version, although initially looking well, starts to lose ground when the problem size grows. For  $n = 2^{22}$  it has roughly the same running time as the  $(1 + (\lambda, \lambda))$  GA with  $\beta \leq 2.3$ . To see whether this effect is stronger when the algorithm starts closer to the optimum, we also conducted the series of experiments when the initial distance to the optimum being only logarithmic. The results are presented in Fig. 2. The logarithmically capped version loses already to  $\beta = 2.5$  this time, indicating that the fast  $(1 + (\lambda, \lambda))$  GA is faster close to the optimum than that.

In order to understand better how different choices for  $\beta$  behave in practice when the starting point also varies, we conducted additional experiments with problem size  $n = 2^{22}$ , but with expected initial distances  $D$  equal to  $2^i$  for  $i \in [0..21]$ . We also normalize all the expected running times by  $\sqrt{nD}$ , but this time we vary  $D$ . The results are presented in Fig. 3, where the results are averaged over 10 runs for distances between  $2^9$  and  $2^{20}$  due to the lack of computational budget. At distances smaller than  $2^{12}$  the smaller  $\beta > 2$  perform noticeably better, as specified in Table 1, however for larger distances the constant factors start to influence the picture: for instance,  $\beta = 2.1$  is outperformed by  $\beta = 2.3$  at distances greater than  $2^{13}$ .

We also included in this figure a few algorithms with  $\beta < 2$ , namely  $\beta \in \{1.5, 1.7, 1.9\}$ , which have a distribution upper bound of  $\sqrt{n}$ , for which running times are averaged over 100 runs. From Fig. 3 we can see that the running time of

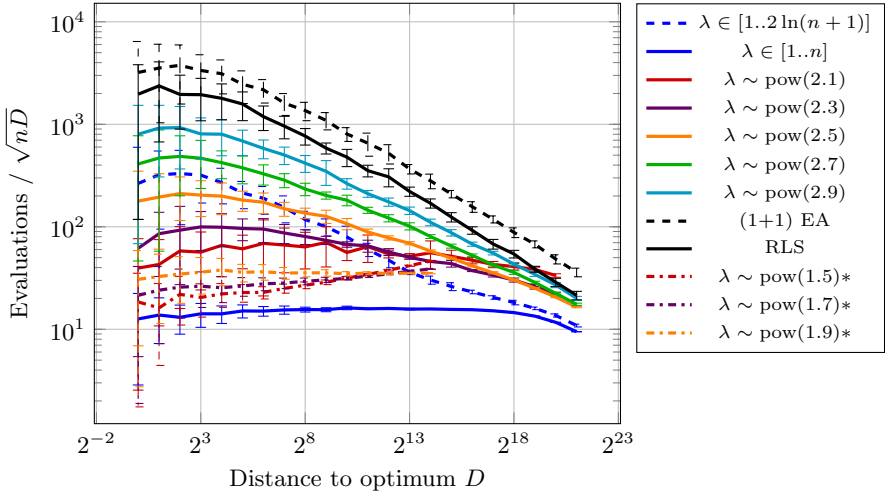


**Fig. 1.** Mean runtimes and their standard deviation of different algorithms on ONEMAX with initial Hamming distance  $D$  from the optimum equal to  $\sqrt{n}$  in expectation. By  $\lambda \in [1..u]$  we denote the self-adjusting parameter choice via the one-fifth rule in the interval  $[1..u]$ . The indicated confidence interval for each value  $X$  is  $[E[X] - \sigma(X), E[X] + \sigma(X)]$ , where  $\sigma(X)$  is the standard deviation of  $X$ . The runtime is normalized by  $\sqrt{nD}$ , so that the plot of the self-adjusting  $(1 + (\lambda, \lambda))$  GA is a horizontal line.



**Fig. 2.** Mean runtimes and their standard deviation of different algorithms on ONEMAX with initial Hamming distance  $D$  from the optimum equal to  $\log(n + 1)$  in expectation.

these algorithms increases with decreasing  $\beta$  just as in Table 1 for comparatively large distances ( $2^{12}$  and up), however for smaller distances their order is reversed, which shows that constant factors still play a significant role.



**Fig. 3.** Mean runtimes and their standard deviation of different algorithms on ONEMAX with problem size  $n = 2^{22}$  and with initial Hamming distances of the form  $D = 2^i$  for  $0 \leq i \leq 21$ . The starred versions of the fast  $(1 + (\lambda, \lambda))$  GA have a distribution upper bound of  $\sqrt{n}$ .

## 5 Conclusion

In this paper we proposed a new notion of the fixed-start runtime analysis, which in some sense complements the fixed-target notion. Among the first results in this direction we observed that different algorithms profit differently from having an access to a solution close to the optimum.

The performance of all observed algorithms, however, is far from the theoretical lower bound. Hence, we are still either to find the EAs which can benefit from good initial solutions or to prove a stronger lower bounds for unary and binary algorithms.

**Acknowledgements.** This work was supported by the Government of Russian Federation, grant number 08-08, and by a public grant as part of the Investissement d’avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences.

## References

1. Antipov, D., Buzdalov, M., Doerr, B.: Fast mutation in crossover-based algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 1268–1276. ACM (2020)
2. Antipov, D., Buzdalov, M., Doerr, B.: First steps towards a runtime analysis when starting with a good solution. CoRR abs/2006.12161 (2020)
3. Antipov, D., Doerr, B., Fang, J., Hetet, T.: Runtime analysis for the  $(\mu + \lambda)$  EA optimizing OneMax. In: Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 1459–1466. ACM (2018)
4. Auger, A., Doerr, B. (eds.): Theory of Randomized Search Heuristics. World Scientific Publishing, Singapore (2011)
5. Buzdalov, M., Doerr, B.: Runtime analysis of the  $(1 + (\lambda, \lambda))$  genetic algorithm on random satisfiable 3-CNF formulas. In: Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 1343–1350. ACM (2017). <http://arxiv.org/abs/1704.04366>
6. Buzdalov, M., Doerr, B., Doerr, C., Vinokurov, D.: Fixed-target runtime analysis. In: Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 1295–1303. ACM (2020)
7. Doerr, B., Doerr, C.: Optimal static and self-adjusting parameter choices for the  $(1 + (\lambda, \lambda))$  genetic algorithm. *Algorithmica* **80**, 1658–1709 (2018)
8. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. *Theoret. Comput. Sci.* **567**, 87–104 (2015)
9. Doerr, B., Doerr, C., Neumann, F.: Fast re-optimization via structural diversity. In: Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 233–241. ACM (2019)
10. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. *Theoret. Comput. Sci.* **801**, 1–34 (2020)
11. Doerr, B., Fouz, M., Witt, C.: Sharp bounds by probability-generating functions and variable drift. In: Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 2083–2090. ACM (2011)
12. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. *Algorithmica* **64**, 673–697 (2012)
13. Doerr, B., Künnemann, M.: Optimizing linear functions with the  $(1 + \lambda)$  evolutionary algorithm—Different asymptotic runtimes for different instances. *Theoret. Comput. Sci.* **561**, 3–23 (2015)
14. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 777–784. ACM (2017)
15. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation—Recent Developments in Discrete Optimization. Springer, Heidelberg (2020). <https://doi.org/10.1007/978-3-030-29414-4>
16. Droste, S., Jansen, T., Wegener, I.: On the analysis of the  $(1+1)$  evolutionary algorithm. *Theoret. Comput. Sci.* **276**, 51–81 (2002)
17. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory Comput. Syst.* **39**, 525–544 (2006)
18. Erdős, P., Rényi, A.: On two problems of information theory. *Magyar Tudományos Akad. Mat. Kutató Intézet Közleményei* **8**, 229–243 (1963)
19. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. *Artif. Intell.* **127**, 51–81 (2001)

20. Jansen, T.: *Analyzing Evolutionary Algorithms - The Computer Science Perspective*. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-17339-4>
21. Jansen, T., Jong, K.A.D., Wegener, I.: On the choice of the offspring population size in evolutionary algorithms. *Evol. Comput.* **13**, 413–440 (2005)
22. Jansen, T., Zarges, C.: Performance analysis of randomised search heuristics operating with a fixed budget. *Theoret. Comput. Sci.* **545**, 39–58 (2014)
23. Johannsen, D.: *Random combinatorial structures and randomized search heuristics*. Ph.D. thesis, Universität des Saarlandes (2010)
24. Liaw, C.: A hybrid genetic algorithm for the open shop scheduling problem. *Eur. J. Oper. Res.* **124**, 28–42 (2000)
25. Mitavskiy, B., Rowe, J.E., Cannings, C.: Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. *Int. J. Intell. Comput. Cybern.* **2**, 243–284 (2009)
26. Mühlenbein, H.: How genetic algorithms really work: mutation and hillclimbing. In: *Parallel Problem Solving from Nature, PPSN 1992*, pp. 15–26. Elsevier (1992)
27. Neumann, F., Pourhassan, M., Roostapour, V.: Analysis of evolutionary algorithms in dynamic and stochastic environments. In: Doerr, B., Neumann, F. (eds.) *Theory of Evolutionary Computation*. NCS, pp. 323–357. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-29414-4\\_7](https://doi.org/10.1007/978-3-030-29414-4_7)
28. Neumann, F., Witt, C.: *Bioinspired Computation in Combinatorial Optimization - Algorithms and Their Computational Complexity*. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-16544-3>
29. Rowe, J.E., Sudholt, D.: The choice of the offspring population size in the  $(1, \lambda)$  evolutionary algorithm. *Theoret. Comput. Sci.* **545**, 20–38 (2014)
30. Schieber, B., Shachnai, H., Tamir, G., Tamir, T.: A theory and algorithms for combinatorial reoptimization. *Algorithmica* **80**, 576–607 (2018)
31. Wald, A.: Some generalizations of the theory of cumulative sums of random variables. *Ann. Math. Stat.* **16**, 287–293 (1945)
32. 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)
33. Witt, C.: Runtime analysis of the  $(\mu + 1)$  EA on simple pseudo-Boolean functions. *Evol. Comput.* **14**, 65–86 (2006)
34. Zych-Pawlewicz, A.: Reoptimization of NP-hard problems. In: Böckenhauer, H.-J., Komm, D., Unger, W. (eds.) *Adventures Between Lower Bounds and Higher Altitudes*. LNCS, vol. 11011, pp. 477–494. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-98355-4\\_28](https://doi.org/10.1007/978-3-319-98355-4_28)



# Optimal Mutation Rates for the $(1 + \lambda)$ EA on OneMax

Maxim Buzdalov<sup>1</sup>(✉) and Carola Doerr<sup>2</sup>

<sup>1</sup> ITMO University, Saint Petersburg, Russia  
mbuzdalov@gmail.com

<sup>2</sup> Sorbonne Université, CNRS, LIP6, Paris, France  
Carola.Doerr@lip6.fr

**Abstract.** The OneMax problem, alternatively known as the Hamming distance problem, is often referred to as the “drosophila of evolutionary computation (EC)”, because of its high relevance in theoretical and empirical analyses of EC approaches. It is therefore surprising that even for the simplest of all mutation-based algorithms, Randomized Local Search and the  $(1 + 1)$  EA, the optimal mutation rates were determined only very recently, in a GECCO 2019 poster.

In this work, we extend the analysis of optimal mutation rates to two variants of the  $(1 + \lambda)$  EA and to the  $(1 + \lambda)$  RLS. To do this, we use dynamic programming and, for the  $(1 + \lambda)$  EA, numeric optimization, both requiring  $\Theta(n^3)$  time for problem dimension  $n$ . With this in hand, we compute for all population sizes  $\lambda \in \{2^i \mid 0 \leq i \leq 18\}$  and for problem dimension  $n \in \{1000, 2000, 5000\}$  which mutation rates minimize the expected running time and which ones maximize the expected progress. Our results do not only provide a lower bound against which we can measure common evolutionary approaches, but we also obtain insight into the structure of these optimal parameter choices. For example, we show that, for large population sizes, the best number of bits to flip is not monotone in the distance to the optimum. We also observe that the expected remaining running times are not necessarily unimodal for the  $(1 + \lambda)$  EA<sub>0→1</sub> with shifted mutation.

**Keywords:** Parameter control · Optimal mutation rates · Population-based algorithms · OneMax

## 1 Introduction

Evolutionary algorithms (EAs) are particularly useful for the optimization of problems for which algorithms with proven performance guarantee are not known; e.g., due to a lack of knowledge, time, computational power, or access to problem data. It is therefore not surprising that we observe a considerable gap between the problems on which EAs are applied, and those for which rigorously proven analyses are available [12].

If there is a single problem that stands out in the EA theory literature, this is the ONEMAX problem, which is considered to be “the drosophila of evolutionary computation” [15]. The ONEMAX problem asks to maximize the simple linear function that counts the number of ones in a bit string, i.e.,  $OM(x) = \sum_{i=1}^n x_i$ . This function is, of course, easily optimized by sampling the unique optimum  $(1, \dots, 1)$ . However, most EAs show identical performance on ONEMAX as on any problem asking to minimize the Hamming distance  $H(z, \cdot)$  to an unknown string  $z$ , i.e.,  $f_z(x) = n - H(z, x)$ , which is a classical problem studied in various fields of Computer Science, starting in the early 60s [14]. In the analysis of EAs, ONEMAX typically plays the role of a benchmark problem that is easy to understand, and on which one can easily test the hill-climbing capabilities of the considered algorithm; very similar to the role of the sphere function in derivative-free numerical optimization [1, 17].

Despite its popularity, and numerous deep results on the ONEMAX problem (see [12] for examples), there are still a number of open questions, and this even for the simplest settings in which the problem is static and noise-free, and the algorithms under consideration can be described in a few lines of pseudo-code. One of these questions concerns the optimal mutation rates of the  $(1 + \lambda)$  EA, i.e., the algorithm which always keeps in memory a best-so-far solution  $x$ , and which samples in each iteration  $\lambda$  “offspring” by applying standard bit mutation to  $x$ . By optimal mutation rates we refer to the values that minimize the expected optimization time, i.e., the average number of function evaluations needed until the algorithm evaluates for the first time an optimal solution. It is not very difficult to see that the optimal mutation rate of this algorithm as well as of its Randomized Local Search (RLS) analog (i.e., the algorithm applying a deterministic mutation strength rather than a randomly sampled one) depend only on the function value  $OM(x)$  of the current incumbent [2, 3, 8]. However, even for  $\lambda = 1$  the optimal mutation rates were numerically computed only in the recent work [4]. Prior to [4], only the rates that maximize the expected progress and those that yield asymptotically optimal running times (in terms of big-Oh notation) were known, see discussion below. It was shown in [4] that the optimal mutation rates are not identical to those maximizing the expected progress, and that the differences can be significant when the current Hamming distance to the optimum is large. In terms of running time, however, the *drift-maximizing* mutation rates are known to yield almost optimal performance, which is another result that was proven only recently [8] (more precisely, it was proven there for Randomized Local Search (RLS), but the result is likely to extend to the  $(1+1)$  EA and its  $(1 + \lambda)$  variants).

**Our Contribution.** We extend in this work the results from [4] to the case  $\lambda \in \{2^i \mid i \in [0..18]\}$ . As in [4] we do not only focus on the standard  $(1 + \lambda)$  EA, but we also consider the  $(1 + \lambda)$  equivalent of RLS and we consider the  $(1 + \lambda)$  EA with the “shift” mutation operator suggested in [22]. The shift mutation operator  $0 \rightarrow 1$  flips exactly one randomly chosen bit when the sampled mutation strength of the standard bit mutation operator equals zero.

Differently from [4] we do not only store the optimal and the drift-maximizing parameter settings for the three different algorithms, but we also store the expected remaining running time of the algorithm that always applies the same fixed mutation rate as long as the incumbent has distance  $d$  to the optimum and that applies the optimal mutation rate at all distances  $d' < d$ . With these values at hand, we can compute the *regret* of each mutation rate, and summing these regrets for a given  $(1 + \lambda)$ -type algorithm gives the exact expected running time, as well as the cumulative regret, which is the expected performance loss of the considered algorithm against the optimal strategy.

Our results extend the main observation shared in [4], which states that, for the  $(1 + 1)$  EA, the drift-maximizing mutation rates are not always also optimal, to the  $(1 + \lambda)$  RLS and to both considered  $(1 + \lambda)$  EAs. We also show that the drift-maximizing and the optimal mutation rates are almost identical across different dimensions, when compared against the normalized distance  $d/n$ .

We also show that, for large population sizes, the optimal number of bits to flip is not monotone in the distance to the optimum. Moreover, we observe that the expected remaining running time is not necessarily unimodal for the  $(1 + \lambda)$  EA $_{0 \rightarrow 1}$  with shifted mutation. Another interesting finding is that some of the drift-maximizing mutation strengths of the  $(1 + \lambda)$  RLS with  $\lambda > 1$  are even, whereas it was proven in [8] that for the  $(1 + 1)$  EA the drift-maximizing mutation strength must always be uneven. The distance  $d$  at which we observe even drift-maximizing mutation strengths decreases with  $\lambda$ , whereas its frequency increases with  $\lambda$ .

**Applications of Our Results in the Analysis of Parameter Control Mechanisms.** Apart from providing several data-driven conjectures about the formal relationship between the optimal and the drift-maximizing parameter settings of the investigated  $(1 + \lambda)$  algorithms, our results have immediate impact on the analysis of parameter control techniques. Not only do we provide an accurate lower bound against which we can measure the performance of other algorithms, but we can also very easily identify where potential performance losses originate from. We demonstrate such an example in Sect. 6, and recall here only that, despite its discussed simplicity, ONEMAX is a very commonly used test case for all types of parameter control mechanisms – not only for theoretical studies [9], but also in purely empirical works [21, 25].

**OneMax Does Not Require Offspring Population.** It is well known that, for the optimization of ONEMAX, the  $(1 + 1)$  EA is the most efficient among the  $(1 + \lambda)$  EAs [20] when measuring performance by fitness evaluations. In practice, however, the  $\lambda$  offspring can be evaluated in parallel, so that – apart from mathematical curiosity – the influence of the population size, the problem size, and the distance to the optimum on the optimal (and on the drift-maximizing) mutation rates also has practical relevance.

**Related Work.** Tight running time bounds for the  $(1 + \lambda)$  EA with *static* mutation rate  $p = c/n$  are proven in [18]. For constant  $\lambda$ , these bounds were



---

**Algorithm 1:** Blueprint of an elitist  $(1 + \lambda)$  unbiased black-box algorithm maximizing a function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ .

---

```

1 Initialization: Sample  $x \in \{0, 1\}^n$  uniformly at random and evaluate  $f(x)$ ;
2 Optimization: for  $t = 1, 2, 3, \dots$  do
3   for  $i = 1, \dots, \lambda$  do
4     Sample  $k(i) \sim D(n, f(x))$ ;
5      $y^{(i)} \leftarrow \text{flip}_{k(i)}(x)$ ;
6     evaluate  $f(y^{(i)})$ ;
7    $y \leftarrow \text{select} \left( \arg \max \{f(y^{(i)}) \mid i \in [\lambda]\} \right)$ ;
8   if  $f(y) \geq f(x)$  then  $x \leftarrow y$ ;
```

---

further refined in [19]. The latter also presents optimal static mutation rates for selected combinations of population size  $\lambda$  and problem size  $n$ .

For the here-considered *dynamic* mutation rates, the following works are most relevant to ours. Bäck [2] studied, by numerical means, the drift-maximizing mutation rates of the classic  $(1 + \lambda)$  EA with standard bit mutation, for problem size  $n = 100$  and for  $\lambda \in \{1, 5, 10, 20\}$ . Mutation rates which minimize the expected optimization time in big-Oh terms were derived in [3, Theorem 4]. More precisely, it was shown there that the  $(1 + \lambda)$  EA using mutation rate  $p(\lambda, n, d) = \max\{1/n, \ln(\lambda)/(n \ln(en/d))\}$  needs  $O\left(\frac{n}{\ln \lambda} + \frac{n \log n}{\lambda}\right)$  function evaluations, on average, to find an optimal solution. This is asymptotically optimal among all  $\lambda$ -parallel mutation-only black-box algorithms [3, Theorem 3]. Self-adjusting and self-adaptive  $(1 + \lambda)$  EAs achieving this running time were presented in [10] and [11], respectively.

## 2 OneMax and $(1 + \lambda)$ Mutation-Only Algorithms

As mentioned, the classical ONEMAX function OM simply counts the number of ones in the string, i.e.,  $\text{OM} : \{0, 1\}^n \rightarrow \mathbb{R}, x \mapsto \sum_{i=1}^n x_i$ . For all algorithms discussed in this work, the behavior on OM is identical to that on any of the problems  $\text{OM}_z : \{0, 1\}^n \rightarrow \mathbb{R}, x \mapsto n - H(z, x) := |\{i \in [n] \mid x_i \neq z_i\}|$ . We study the maximization of these problems.

Algorithm 1 summarizes the structure of the algorithms studied in this work. All algorithms start by sampling a uniformly chosen point  $x$ . In each iteration,  $\lambda$  offspring  $y^{(1)}, \dots, y^{(\lambda)}$  are sampled from  $x$ , independently of each other. Each  $y^{(i)}$  is created from the incumbent  $x$  by flipping some  $k(i)$  bits, which are pairwise different, independently and uniformly chosen (this is the operator `flip` in line 4). The best of these  $\lambda$  offspring replaces the incumbent if it is at least as good as it (line 8). When  $\arg \max \{f(y^{(i)}) \mid i \in [\lambda]\}$  contains more than one point, the selection operator `select` chooses one of them, e.g., uniformly at random or via some other rule. As a consequence of the symmetry of ONEMAX, all results shown in this work apply regardless of the chosen tie-breaking rule.

What is left to be specified is the distribution  $D(n, f(x))$  from which the *mutation strengths*  $k(i)$  are chosen in line 3. This is the only difference between the algorithms studied in this work.

**Deterministic vs. Random Sampling:** The Randomized Local Search variants (RLS) use a deterministic mutation strength  $k(i)$ , i.e., the distributions  $D(n, f(x))$  are one-point Dirac distributions. We distinguish two EA variants: the one using standard bit mutation, denoted  $(1 + \lambda)$  EA<sub>sbm</sub>, and the one using the shift mutation suggested in [22], which we refer to as  $(1 + \lambda)$  EA<sub>0→1</sub>. *Standard bit mutation* uses the binomial distribution  $\text{Bin}(n, p)$  with  $n$  trials and success probability  $p$ . The *shift mutation* operator uses  $\text{Bin}_{0 \rightarrow 1}(n, p)$ , which differs from  $\text{Bin}(n, p)$  only in that all probability mass for  $k = 0$  is moved to  $k = 1$ . That is, with shift mutation we are guaranteed to flip at least one bit, and the probability to flip exactly one bit equals  $(1 - p)^n + np(1 - p)^{n-1}$ . In both cases we refer to  $p$  as the *mutation rate*.

**Optimal vs. Drift-maximizing Rates:** Our main interest is in the *optimal* mutation rates, which minimize the expected time needed to optimize ONEMAX. Much easier to compute than the optimal mutation rates are the *drift-maximizing ones*, i.e., the values which maximize the expected gain  $\mathbb{E}[f(y) - f(x) \mid y \leftarrow \text{flip}_k(x), k \sim D(n, f(x))]$ , see Sect. 3.

**Notational Convention.** We omit the explicit mention of  $(1 + \lambda)$  when the value of  $\lambda$  is clear from the context. Also, formally, we should distinguish between the *mutation rate* (used by the EAs, see above) and the *mutation strengths* (i.e., the number of bits that are flipped). However, to ease presentation, we will just speak of *mutation rates* even when referring to the parameter setting for RLS.

### 3 Computation of Optimal Parameter Configurations

We compute the optimal parameters using the similar flavor of dynamic programming that has already been exploited in [4]. Namely, we compute the optimal parameters and the corresponding remaining time expectations for Hamming distance  $d$  to the optimum after we have computed them for all smaller distances  $d' < d$ . We denote by  $T_{D,O}^*(n, \lambda, d)$  the minimal expected remaining time of a  $(1 + \lambda)$  algorithm with mutation rate distribution  $D \in \{\text{RLS}, \text{sbm}, 0 \rightarrow 1\}$ , optimality criterion  $O \in \{\text{opt}, \text{drift}\}$ , and population size  $\lambda$  on a problem size  $n \in \mathbb{N}$  when at distance  $d \in [0..n]$ . We also denote the distribution parameter (mutation strength or rate) by  $\rho$ , and the optimal distribution parameter for the current context as  $\rho_{D,O}^*(n, \lambda, d)$ .

Let  $P_{n,D}(d, d', \rho)$  be the probability of sampling an offspring at distance  $d'$  to the optimum, provided the parent is at distance  $d$ , the problem size is  $n$ , the distribution function is  $D$ , and the distribution parameter is  $\rho$ . The expected remaining time  $T_{D,O}(n, \lambda, d, \rho)$ , which assumes that at distance  $d$  the algorithm consistently uses parameter  $\rho$  and at all smaller distances it uses the optimal (time-minimizing or drift-maximizing, respectively) parameter for that distance, is then computed as follows:

$$T_{D,O}(n, \lambda, d, \rho) = \frac{1}{(P_{n,D}(d, d, \rho))^\lambda} + \sum_{d'=1}^{d-1} T_{D,O}^*(n, \lambda, d') \cdot P_{n,D}^\lambda(d, d', \rho), \quad (1)$$

where  $P_{n,D}^\lambda(d, d', \rho) = \left(\sum_{t=d'}^d P_{n,D}(d, t, \rho)\right)^\lambda - \left(\sum_{t=d'+1}^d P_{n,D}(d, t, \rho)\right)^\lambda$ .

To compute  $T_{D,O}^*(n, \lambda, d)$ , Eq. (1) is used, where direct minimization of  $\rho$  is performed when  $O = \text{opt}$ , and the following drift-maximizing value of  $\rho$  is substituted when  $O = \text{drift}$ :  $\rho_{n,D}(d) = \arg \max_{\rho} \sum_{d'=0}^{d-1} (d - d') \cdot P_{n,D}^\lambda(d, d', \rho)$ .

Another difference to the work of [4] is in that we do not only compute the expected remaining running times  $T_{D,O}^*(n, \lambda, d)$  when using the optimal mutation rates  $\rho_{D,O}^*(n, \lambda, d)$ , but we also compute and store  $T_{D,O}(n, \lambda, d, \rho)$  for suboptimal values of  $\rho$ . For RLS we do that for all possible values of  $\rho$ , which are integers not exceeding  $n$ , while for the  $(1 + \lambda)$  EA we consider  $\rho = 2^{i/5-10}/n$  for all  $i \in [0; 150]$ . We do this not only because it gives us additional insight into the sensitivity of  $T_{D,O}(n, \lambda, d, \rho)$  with respect to  $\rho$ , but it also offers a convenient way to detect deficits of parameter control mechanism; see Sect. 6 for an illustrated example. Since our data base is hence much more detailed than that of [4], we also re-consider the case  $\lambda = 1$ .

Our code has the  $\Theta(n^3)$  runtime and  $\Theta(n^2)$  memory complexity. The code is available on GitHub [6], whereas the generated data is available on Zenodo [5].

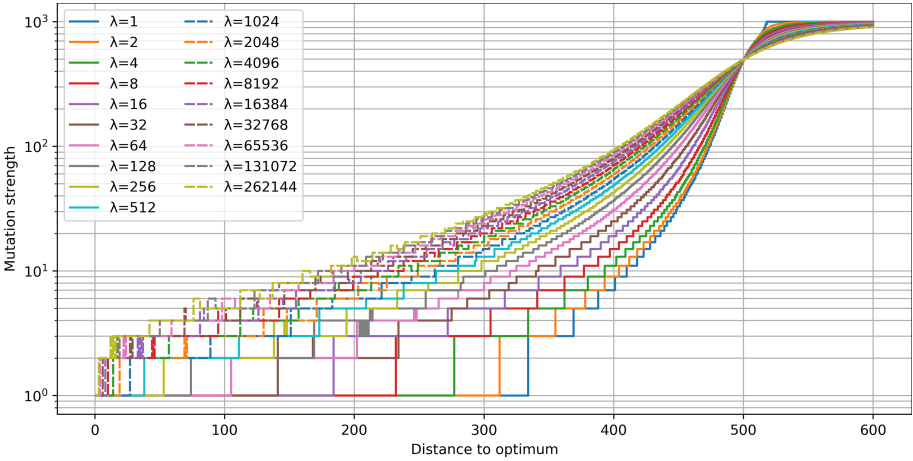
## 4 Optimal Mutation Rates and Optimal Running Times

Figure 1 plots the optimal parameter settings  $\rho_{\text{RLS,opt}}^*(n, \lambda, d)$  for fixed dimension  $n = 10^3$  and for different values of  $\lambda$ , in dependence of the Hamming distance  $d$  to the optimum. We observe that the mutation strengths  $\rho_{\text{RLS,opt}}^*(n, \lambda, d)$  are nearly monotonically increasing in  $\lambda$ , as a result of having more trials to generate an offspring with large fitness gain. We also see that, for some values of  $\lambda$ , the curves are not monotonically decreasing in  $d$ , but show small ‘‘bumps’’. Similar non-monotonic behavior can also be observed for drift-maximizing mutation strengths  $\rho_{\text{RLS,drift}}^*(n, \lambda, d)$ , as can be seen in Fig. 2.

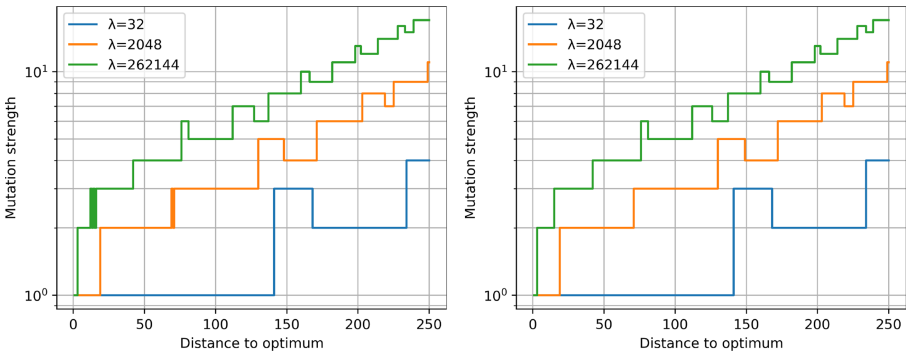
**Table 1.** Drifts for  $n = 30$ ,  $\lambda = 512$ ,  $d = 7, 8$ ,  $\rho \in [1..10]$ .

$d$	$\rho = 1$	2	3	4	5	6	7	8	9	10
7	0.5000	2.0000	2.9762	2.9604	<b>3.0434</b>	2.7009	2.5766	2.2292	1.7457	1.3854
8	0.5000	2.0000	2.9984	<b>3.4601</b>	3.3583	3.3737	3.2292	2.9124	2.7323	2.3445

We show now that these ‘‘bumps’’ are not just numeric precision artifacts, but rather a (quite surprising) feature of the parameter landscape. For a small example that can be computed by a human we consider  $n = 30$  and  $\lambda = 512$ . For  $d = 7$  and 8, we compute the drifts for mutation strengths in  $[1..10]$ .



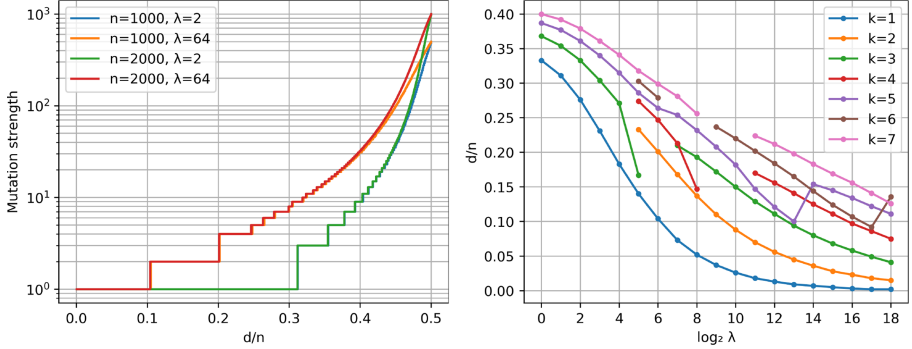
**Fig. 1.** Optimal parameters  $\rho_{\text{RLS,opt}}^*(n, \lambda, d)$  for different values of  $\lambda$  and  $n = 1000$  as a function of  $d$ , the distance to the optimum



**Fig. 2.** Non-monotonicity in optimal (left) and drift-optimal (right) mutation strengths for  $n = 1000$  and selected  $\lambda$

These values are summarized in Table 1. Here we see that the drift-maximizing mutation for  $d = 7$  is 5, whereas for  $d = 8$  it is 4. This example, in fact, serves two purposes: first, it shows that even the drift-maximizing strengths can be non-monotone, and second, that the drift-maximizing strengths can be even for non-trivial problem sizes, which – as mentioned in the introduction – cannot be the case when  $\lambda = 1$  [8].

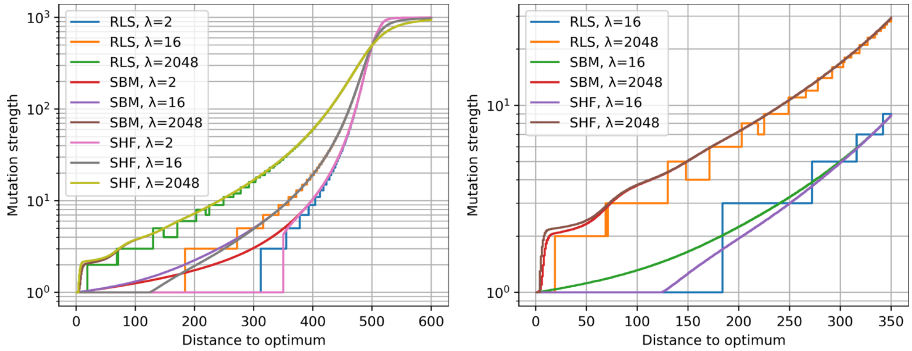
In the left chart of Fig. 3 we show that at least small  $\rho_{\text{RLS,opt}}^*(n, \lambda, d)$  are quite robust with respect to the problem dimension  $n \in \{1, 2\} \cdot 10^3$ , if the Hamming distance  $d$  to the optimum is appropriately scaled as  $d/n$ . The chart plots the curves for  $\lambda \in \{2, 64\}$  only, but the observation applies to all tested values of  $\lambda$ . In accordance to our previous notes, we also see that for  $\lambda = 64$  there is a regime



**Fig. 3.** Left:  $\rho_{\text{RLS,opt}}^*(n, \lambda, d)$  for  $\lambda \in \{2, 64\}$  and  $n \in \{1k, 2k\}$ , in dependence of  $d/n$ . Right: normalized maximal distance  $d/n$  at which flipping  $k \in [1..7]$  bits is optimal for RLS, for  $n = 10^3$  and  $\lambda \in \{2^i \mid 0 \leq i \leq 18\}$ .

for which flipping two bits is optimal. For small population sizes  $\lambda$ , we also obtain even numbers for certain regimes, but only for much larger distances.

The maximal distances at which flipping  $k$  bits is optimal are summarized in the chart on the right of Fig. 3. Note here that the curves are less smooth than one might have expected. For instance, for  $n = 10^3$ , flipping three bits is never optimal for  $\lambda = 64$ , and flipping seven bits is never optimal for  $\lambda = 2^9$  and  $2^{10}$ .



**Fig. 4.** Left: expected mutation strengths of the time-minimizing parameter settings for the  $(1 + \lambda)$  RLS and two  $(1 + \lambda)$  EAs,  $\lambda \in \{2, 16, 2048\}$ , using standard bit mutation (SBM) and shift mutation (SHF), respectively. Values are for  $n = 1000$  and plotted against the Hamming distance to the optimal solution. Right: same for  $\lambda \in \{16, 2048\}$  with an emphasis on small distances.

In Fig. 4 we compare the optimal (i.e., time-minimizing) parameter settings of the  $(1 + \lambda)$  variants of RLS, the  $\text{EA}_{0 \rightarrow 1}$ , and the  $\text{EA}_{\text{sbm}}$ . To obtain a proper comparison, we compare the mutation strength  $\rho_{\text{RLS,opt}}^*(n, \lambda, d)$  with the expected

number of bits that flip in the two EA variants, i.e.,  $n\rho_{\text{sbm,opt}}^*(n, \lambda, d)$  for the EA using standard bit mutation and  $n\rho_{\text{sbm,opt}}^*(n, \lambda, d) + (1 - \rho_{\text{sbm,opt}}^*(n, \lambda, d))^n$  for the EA using the shift mutation operator. We show here only values for  $\lambda \in \{2, 16, 1024\}$ , but the picture is similar for all evaluated  $\lambda$ .

We observe that, for each  $\lambda$ , the curves are close together. While for  $\lambda = 1$  the curves for standard bit mutation were always below that of RLS, we see here that this picture changes with increasing  $\lambda$ . We also see a sudden decrease in the expected mutation strength of the shift operator when  $\lambda$  is small. In fact, it is surprising to see that, for  $\lambda = 2$ , the value drops from around 5.9 at distance 373 to 1 at distance 372. This is particularly interesting in light of a common tendency in state-of-the-art parameter control mechanisms to allow only for small parameter updates between two consecutive iterations. This is the case, for example, in the well-known one-fifth success rule [7, 23, 24]. Parameter control techniques building on the family of reinforcement learning algorithms (see [16] for examples) might catch such drastic changes more efficiently.

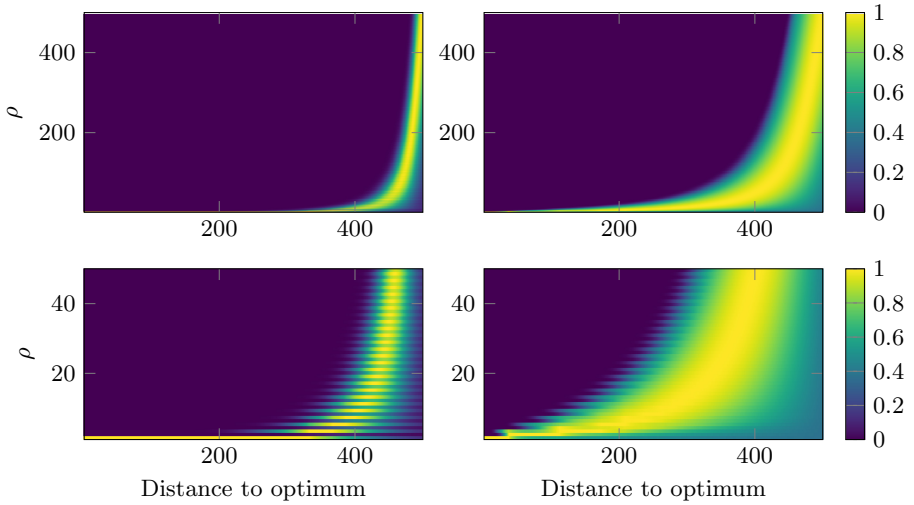
Non-surprisingly, the expected mutation strengths of the optimal standard bit mutation rate and the optimal shift mutation rate converge as the distance to the optimum increases.

## 5 Sensitivity of the Optimization Time w.r.t the Parameter Settings

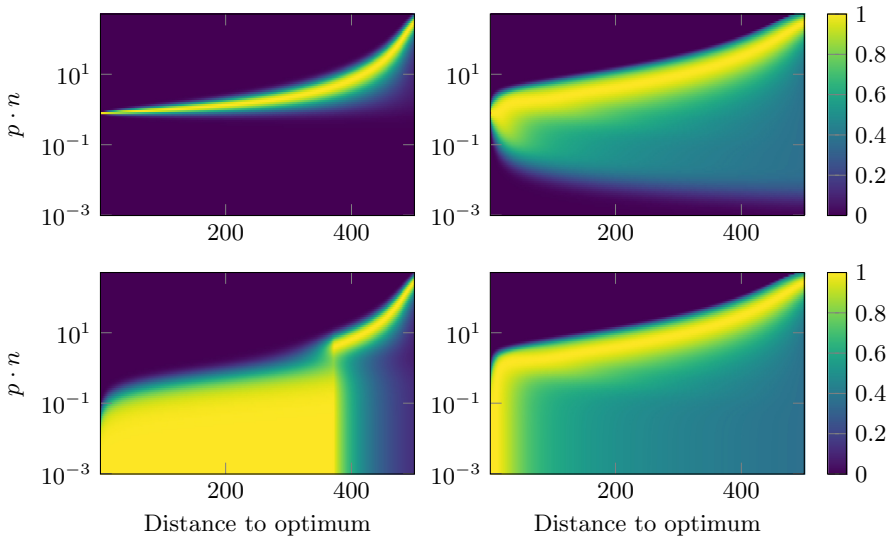
In this section, we present our findings on the sensitivity of the considered  $(1 + \lambda)$  algorithms to their mutation parameters. To do this, we use not only the expected remaining times  $T_{D,O}^*(n, \lambda, d)$  that correspond to optimal parameter values, but also  $T_{D,O}(n, \lambda, d, \rho)$  for various parameter values  $\rho$ , which correspond to the situation when an algorithm uses the parameter  $\rho$  while it remains at distance  $d$ , and switches to using the optimal parameter values (time-minimizing for  $O = \text{opt}$  and drift-maximizing for  $O = \text{drift}$ , respectively) once the distance is improved. For reasons of space we focus on  $O = \text{opt}$ .

We use distance-versus-parameter heatmaps as a means to show which parameter values are efficient. As the non-optimality regret  $\delta_{D,O}(n, \lambda, d, \rho) = T_{D,O}(n, \lambda, d, \rho) - T_{D,O}^*(n, \lambda, d)$  is asymptotically smaller than the remaining time, we derive the color from the value  $\tau(\rho) = \exp(-\delta_{D,O}(n, \lambda, d, \rho))$ . Note that  $\tau(\rho) \in (0; 1]$ , and the values close to one represent parameters that are almost optimal by their effect. The parameters where  $\tau(\rho) \approx 0.5$ , on the other hand, correspond to a regret of roughly 0.7, that is, if the parameters satisfy  $\tau(\rho) \geq 0.5$  throughout the entire optimization, the total expected running time is greater by at most  $0.7n/2$  than the optimal time for this type of algorithms.

Figure 5 depicts these regrets for  $\text{RLS}_{\text{opt}}$  on  $n = 10^3$  and  $\lambda \in \{1, 512\}$ . The stripes on the fine-grained plot for  $\lambda = 1$  expectedly indicate, as in [4], that flipping an even number of bits is generally non-optimal when the distance to the optimum is small, which is the most pronounced for  $\rho = 2$ . This also indicates that the parameter landscape of RLS is multimodal, posing another difficulty to

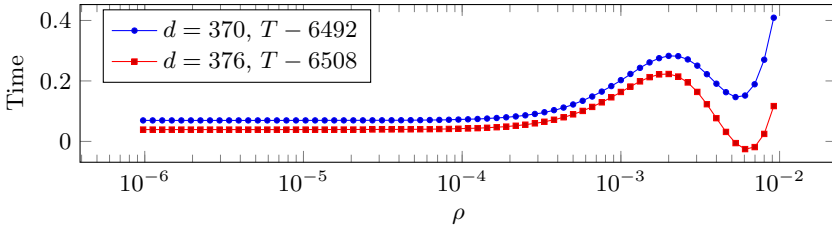


**Fig. 5.** Relative expected remaining optimization times for the  $(1 + \lambda)$   $\text{RLS}_{\text{opt}}$  with parameters  $n = 10^3$ ,  $\lambda = 1$  (left) and  $\lambda = 512$  (right). The first row displays the general picture, the second row focuses on small mutation strengths



**Fig. 6.** Relative expected remaining optimization times for the  $(1 + \lambda)$   $\text{EA}_{\text{sbm,opt}}$  (top) and the  $(1 + \lambda)$   $\text{EA}_{0 \rightarrow 1,\text{opt}}$  (bottom) with  $\lambda = 1$  (left) and  $\lambda = 512$  (right)

parameter control methods. The parameter-time landscape remains multimodal for  $\lambda = 512$ , but the picture is now much smoother around the optimal parameter values.



**Fig. 7.** Expected remaining optimization time of the  $(1 + \lambda)$   $EA_{0 \rightarrow 1, \text{opt}}$  as a function of the mutation probability  $\rho$

Figure 6 plots the regret for the  $(1 + \lambda)$   $EA_{\text{sbm}}$  (top) and the  $(1 + \lambda)$   $EA_{0 \rightarrow 1}$  (bottom) with  $\lambda = 1$  (left) and  $\lambda = 512$  (right). The pictures for standard and shift mutations are very similar until the distance is so small that one-bit flips become nearly optimal. We also see that bigger population sizes result in a lower sensitivity of the expected remaining optimization time with respect to the mutation rate. In fact, we see that, even for standard bit mutation, parameter settings that are much smaller than the typically recommended mutation rates (e.g.,  $\rho = 1/(10n)$ ) are also good enough when the distance is  $\Omega(n)$ , as the probability to flip at least one bit at least once is still quite large.

The plot for the  $(1+1)$   $EA_{0 \rightarrow 1}$  deserves separate attention. Unlike other plots in Fig. 6, it shows a bimodal behavior with respect to the mutation probability  $\rho$  even for quite large distances  $d < n/2$ . We zoom into this effect by displaying in Fig. 7 the expected remaining optimization times for  $d \in \{370, 376\}$ .

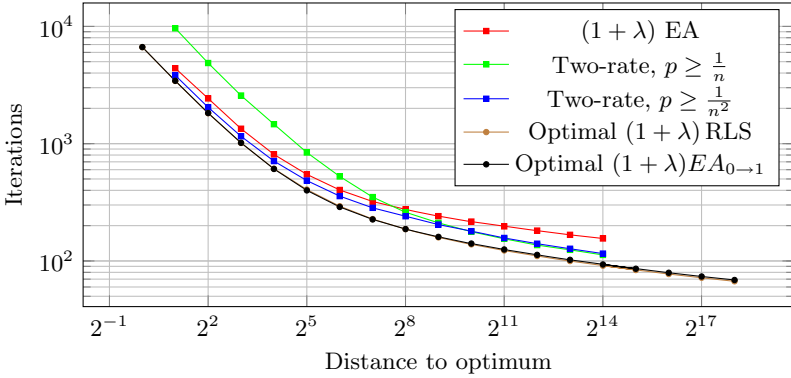
**Drift-Maximization vs. Time-Minimization.** We note, without diving into the details, that the observation that the optimal mutation parameters are not identical to the drift maximizing ones, made in [4] for  $(1 + 1)$  algorithms, extends to  $(1 + \lambda)$ -type algorithms with  $\lambda > 1$ . More precisely, it applies to all tested dimensions and population sizes  $\lambda$ . We note, though, that the disadvantage  $T_{\text{RLS,drift}}^*(n, \lambda, d) - T_{\text{RLS,opt}}^*(n, \lambda, d)$  decreases with increasing  $\lambda$ . Since the difference is already quite small for the case  $\lambda = 1$  (e.g., for  $n = 1000$ , it is 0.242), we conclude that this difference, albeit interesting from a mathematical perspective, has very limited relevance in empirical evaluations. This is good news for automated algorithm configuration techniques, as it implies that simple regret (e.g., in the terms of one-step progress) is sufficient to derive reasonable parameter values – as opposed to requiring cumulative regret, which, as Sect. 3 shows, is much more difficult to track.

## 6 Applications in Parameter Control

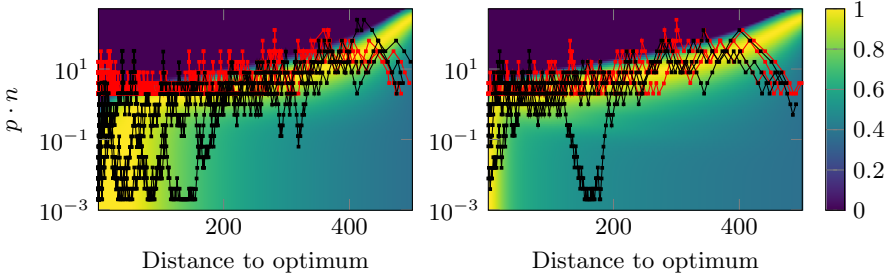
Figure 8 displays the experimentally measured mean optimization times, averaged over 100 runs, of (1) the standard  $(1 + \lambda)$  EA with static mutation rate  $\rho = 1/n$ , (2)  $RLS_{\text{opt}}$ , (3) the  $(1 + \lambda)$   $EA_{0 \rightarrow 1, \text{opt}}$ , and of (4–5) the “two-rate” parameter control mechanism suggested in [10], superposed here to the



$(1 + \lambda)$  EA $_{0 \rightarrow 1}$  with two different lower bounds  $\rho_{\min}$  at which the mutation rate is capped.



**Fig. 8.** Mean number of iterations of different  $(1 + \lambda)$  EAs vs. the expected number of iterations of RLS $_{\text{opt}}$  and EA $_{\text{opt},0 \rightarrow 1}$  for  $n = 10^3$ , as a function of the population size  $\lambda$



**Fig. 9.** Parameter control plots of the two-rate method atop parameter efficiency heatmaps,  $n = 10^3$ ,  $\lambda = 64$  (left) and  $\lambda = 2048$  (right). Red traces are for the mutation rate lower bound of  $\rho_{\min} = 1/n$ , black traces are for the lower bound of  $\rho_{\min} = 1/n^2$  (Color figure online)

With such pictures, we can infer how far a certain algorithm is from an optimally tuned algorithm with the same structure, which can highlight its strengths and weaknesses. However, it is difficult to derive insights from just expected times. To get more information, one can record the parameter values produced by the investigated parameter control method and draw them atop the heatmaps produced as in Sect. 5. An example of this is shown in Fig. 9. An insight from this figure, that may be relevant to the analysis of strengths and weaknesses of this method, would be that the version using  $\rho_{\min} = 1/n$  cannot use very small probabilities and is thus suboptimal at distances close to the optimum, whereas the version using  $\rho_{\min} = 1/n^2$  falls down from the optimal parameter region too frequently and too deep.

## 7 Conclusions

Extending the work [4], we have presented in this work optimal and drift-maximizing mutation rates for two different  $(1 + \lambda)$  EAs and for the  $(1 + \lambda)$  RLS. We have demonstrated how our data can be used to detect weak spots of parameter control mechanisms. We have also described two unexpected effects of the dependency of the expected remaining optimization time on the mutation rates: non-monotonicity in  $d$  (Sect. 4) and non-unimodality (Sect. 5). We plan on exploring these effects in more detail, and with mathematical rigor. Likewise, we plan on analyzing the formal relationship of the optimal mutation rates with the normalized distance  $d/n$ . As a first step towards this goal, we will use the numerical data presented above to derive close-form expressions for the expected remaining optimization times  $T_{D,O}(n, \lambda, d, \rho)$  as well as for the optimal configurations  $\rho_{D,O}^*(n, \lambda, d)$ . Finally, we also plan on applying similar analyses to more sophisticated benchmark problems.

The extended version of the paper is available on arXiv [13].

**Acknowledgments.** The work was supported by RFBR and CNRS, project no. 20-51-15009, by the Paris Ile-de-France Region, and by ANR-11-LABX-0056-LMH.

## References

1. Auger, A., Hansen, N.: Theory of evolution strategies: a new perspective. In: Theory of Randomized Search Heuristics: Foundations and Recent Developments, pp. 289–325. World Scientific (2011). [https://doi.org/10.1142/9789814282673\\_0010](https://doi.org/10.1142/9789814282673_0010)
2. Bäck, T.: The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In: Proceedings of Parallel Problem Solving from Nature (PPSN II), pp. 87–96. Elsevier (1992)
3. Badkobeh, G., Lehre, P.K., Sudholt, D.: Unbiased black-box complexity of parallel search. In: Bartz-Beielstein, T., Branke, J., Filipič, B., Smith, J. (eds.) PPSN 2014. LNCS, vol. 8672, pp. 892–901. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10762-2\\_88](https://doi.org/10.1007/978-3-319-10762-2_88)
4. Buskalic, N., Doerr, C.: Maximizing drift is not optimal for solving OneMax. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2019), Companion Material, pp. 425–426. ACM (2019). <https://doi.org/10.1145/3319619.3321952>. <https://arxiv.org/abs/1904.07818>
5. Buzdalov, M.: Data for “Optimal mutation rates for the  $(1 + \lambda)$  EA on OneMax” (2020). <https://doi.org/10.5281/zenodo.3897351>
6. Buzdalov, M.: Repository with the code to compute the optimal rates and the expected remaining optimization times (2020). <https://github.com/mbuzdalov/one-plus-lambda-on-onemax/releases/tag/v1.0>
7. Devroye, L.: The compound random search. Ph.D. dissertation, Purdue Univ., West Lafayette, IN (1972)
8. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. Theor. Comput. Sci. **801**, 1–34 (2020). <https://doi.org/10.1016/j.tcs.2019.06.014>

9. Doerr, B., Doerr, C.: Theory of parameter control for discrete black-box optimization: provable performance gains through dynamic parameter choices. In: Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 271–321. Springer, Cham (2020). <https://arxiv.org/abs/1804.05650v2>
10. Doerr, B., Gießen, C., Witt, C., Yang, J.: The  $(1 + \lambda)$  evolutionary algorithm with self-adjusting mutation rate. *Algorithmica* **81**(2), 593–631 (2019)
11. Doerr, B., Witt, C., Yang, J.: Runtime analysis for self-adaptive mutation rates. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2018), pp. 1475–1482. ACM (2018). <https://doi.org/10.1145/3205455.3205569>
12. Doerr, B. and Neumann, F. (eds.) Theory of Evolutionary Computation—Recent Developments in Discrete Optimization, Springer, Cham (2020). <https://doi.org/10.1007/978-3-030-29414-4>
13. Doerr, C., Buzdalov, M.: Optimal mutation rates for the  $(1 + \lambda)$  EA on OneMax (2020). <https://arxiv.org/abs/2006.11457>
14. Erdős, P., Rényi, A.: On two problems of information theory. *Magyar Tudományos Akadémia Matematikai Kutató Intézet Közleményei* **8**, 229–243 (1963)
15. Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M.: Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In: Stützle, T. (ed.) LION 2009. LNCS, vol. 5851, pp. 176–190. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-11169-3\\_13](https://doi.org/10.1007/978-3-642-11169-3_13)
16. Fialho, Á., Costa, L.D., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. *Ann. Math. Artif. Intell.* **60**(1–2), 25–64 (2010). <https://doi.org/10.1007/s10472-010-9213-y>
17. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2010: presentation of the noiseless functions (2010). <http://coco.gforge.inria.fr/downloads/download16.00/bbobdocfunctions.pdf>
18. Gießen, C., Witt, C.: The interplay of population size and mutation probability in the  $(1 + \lambda)$  EA on OneMax. *Algorithmica* **78**(2), 587–609 (2016). <https://doi.org/10.1007/s00453-016-0214-z>
19. Gießen, C., Witt, C.: Optimal mutation rates for the  $(1 + \lambda)$  EA on OneMax through asymptotically tight drift analysis. *Algorithmica* **80**(5), 1710–1731 (2017). <https://doi.org/10.1007/s00453-017-0360-y>
20. Jansen, T., De Jong, K.A., Wegener, I.: On the choice of the offspring population size in evolutionary algorithms. *Evol. Comput.* **13**(4), 413–440 (2005)
21. Karafotias, G., Hoogendoorn, M., Eiben, A.: Parameter control in evolutionary algorithms: trends and challenges. *IEEE Trans. Evol. Comput.* **18**(2), 167–187 (2014)
22. Pinto, E.C., Doerr, C.: Towards a more practice-aware runtime analysis of evolutionary algorithms. [arxiv.org/abs/1812.00493](https://arxiv.org/abs/1812.00493) (2018)
23. Rechenberg, I.: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboorg Verlag, Stuttgart (1973)
24. Schumer, M.A., Steiglitz, K.: Adaptive step size random search. *IEEE Trans. Autom. Control* **13**, 270–276 (1968)
25. Thierens, D.: On benchmark properties for adaptive operator selection. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2009), Companion Material, pp. 2217–2218. ACM (2009). <https://doi.org/10.1145/1570256.1570306>



# Maximizing Submodular or Monotone Functions Under Partition Matroid Constraints by Multi-objective Evolutionary Algorithms

Anh Viet Do and Frank Neumann<sup>(✉)</sup>

The University of Adelaide, Adelaide, Australia  
{vietanh.do, frank.neumann}@adelaide.edu.au

**Abstract.** Many important problems can be regarded as maximizing submodular functions under some constraints. A simple multi-objective evolutionary algorithm called GSEMO has been shown to achieve good approximation for submodular functions efficiently. While there have been many studies on the subject, most of existing run-time analyses for GSEMO assume a single cardinality constraint. In this work, we extend the theoretical results to partition matroid constraints which generalize cardinality constraints, and show that GSEMO can generally guarantee good approximation performance within polynomial expected run time. Furthermore, we conducted experimental comparison against a baseline GREEDY algorithm in maximizing undirected graph cuts on random graphs, under various partition matroid constraints. The results show GSEMO tends to outperform GREEDY in quadratic run time.

**Keywords:** Evolutionary algorithms · Multi-objective evolutionary algorithms · Run-time analysis

## 1 Introduction

The area of runtime analysis has made significant contributions to the theory of evolutionary algorithms over the last 25 years [1, 20]. Important results have been obtained for a wide range of benchmark functions as well as for important combinatorial optimization problems [33]. This includes a wide range of evolutionary computing methods in a wide range of deterministic, stochastic and dynamic settings. We refer the reader to [12] for a presentation of important recent research results.

Many important real-world problems can be stated in terms of optimizing a submodular function and the analysis of evolutionary algorithms using multi-objective formulations has shown that they obtain in many cases the best possible performance guarantee (unless  $P = NP$ ). Important recent results on the use of evolutionary algorithms for submodular optimization are summarized

in [42]. The goal of this paper is to expand the investigations of evolutionary multi-objective optimization for submodular optimization. While previous investigations mainly concentrated on monotone submodular functions with a single constraint, we consider non-monotone submodular functions with a set of constraints.

## 1.1 Related Work

Submodular functions are considered the discrete counterparts of convex functions [31]. Submodularity captures the notion of diminishing marginal return, and is present in many important problems. While minimizing submodular functions can be done using a polynomial time combinatorial algorithm [19], submodular maximization encompasses many NP-hard combinatorial problems such as maximum coverage, maximum cut [18], maximum influence [22], and sensor placement problem [23, 24]. It is also applied in many problems in machine learning domain [28–30, 38, 41]. Considering the role of evolutionary algorithms in difficult optimization problems, we focus on submodular maximization.

Realistic optimization problems often impose constraints on the solutions. In applications of submodular maximization, Matroid and Knapsack constraints are among the most common [26]. In this work, we consider submodular maximization under partition matroid constraints, which are a generalization of cardinality constraints. This type of constraint has been considered in a variety of applications [6, 14, 21].

A greedy algorithm has been shown to achieve  $1/2$ -approximation ratio in maximizing monotone submodular functions under partition matroid constraints [9]. It was later proven that  $(1-1/e)$  is the best approximation ratio a polynomial time algorithm can guarantee. A more recent study [10] proposed a randomized algorithm that achieves this ratio. Another study [5] analyzes derandomizing search heuristics, leading to a deterministic  $0.5008$ -approximation ratio.

Additionally, more nuanced results have been reached when limiting objective functions to those with finite rate of marginal change, quantified by curvature  $\alpha$  as defined in [15]. The results in [7, 40] indicate that  $\frac{1}{\alpha}(1 - e^{-\alpha})$ -approximation ratio is achievable by the continuous greedy algorithm in maximizing monotone submodular functions under a matroid constraint. A more recent study [4] proved  $\frac{1}{\alpha}(1 - e^{-\gamma\alpha})$ -approximation ratio for the deterministic greedy algorithm in maximizing functions with submodularity ratio  $\gamma$ , under a cardinality constraint.

These results rely on the assumption of monotonicity of the objective functions,  $f(S) \leq f(T)$  for all  $S \subseteq T$ , which do not hold in many applications of submodular maximization. A study [17] derives approximation guarantees for GSEMO algorithm in maximizing monotone and symmetric submodular function under a matroid constraint, which suggest that non-monotone functions are harder to maximize. This is supported by another result [15] for a greedy algorithm in maximizing general submodular function under partition matroid constraints. A recent study [36] extends the results for a GSEMO variant to the problems of maximizing general submodular functions, but under a cardinality constraint.

## 1.2 Our Contribution

In this work, we contribute to the theoretical analysis of GSEMO by generalizing previous results to partition matroid constraints. Firstly, we provide an approximation guarantee for GSEMO in maximizing general submodular functions under partition matroid constraints (Theorem 1). Secondly, we derive another result for monotone and not necessarily submodular functions, under the same constraints (Theorem 2), to account for other important types of function like subadditive functions. Subadditivity encompasses submodularity, and is defined by the property where the whole is no greater than the sum of parts. Subadditive functions are commonly used to model items evaluations and social welfare in combinatorial auctions [2, 3, 15, 39]. Our results extend the existing ones [36] with more refined bounds.

We investigate GSEMO's performance against GREEDY's [15] in maximizing undirected cuts in random graphs under varying cardinality constraints and partition matroid constraints. Graph cut functions with respect to vertices sets are known to be submodular and non-monotone [13]. In particular, they are also symmetric for undirected graphs [37]. Our results suggest that GSEMO typically requires more evaluations to reach GREEDY's outputs quality. Nonetheless, GSEMO surpasses GREEDY shortly after the latter stops improving, indicating the former's capacity for exploring the search spaces. Predictably, GSEMO outperforms GREEDY more reliably in larger search spaces.

The paper is structured as follows. We formally define the problems and the algorithms in Sect. 2. In Sect. 3, we analyze GSEMO with respect to its approximation behaviour and runtime and report on our experimental investigations in Sect. 4. Finally, we finish with some conclusions.

## 2 Preliminaries

In this section, we provide a formal definition of the problem and some of its parameters relevant to our analyses. We also describe the simple GREEDY algorithm and the GSEMO algorithm considered in this work.

### 2.1 Problem Definition

We consider the following problem. Let  $f : 2^V \rightarrow \mathbb{R}^+$  be a non-negative function over a set  $V$  of size  $n$ ,  $B = \{B_i\}_{i=1,\dots,k}$  be a partition of  $V$  for some  $k \leq n$ ,  $D = \{d_i\}_{i=1,\dots,k}$  be integers such that  $d_i \in [1, |B_i|]$  for all  $i$ , the problem is finding  $X \subseteq V$  maximizing  $f(X)$ , subject to

$$|X \cap B_i| \leq d_i, \quad \forall i = 1, \dots, k.$$

These constraints are referred to as partition matroid constraints, which are equivalent to a cardinality constraint if  $k = 1$ . The objective function  $f$  of interest is submodular, meaning it satisfies the property as defined in [32]

**Definition 1.** A function  $f : 2^V \rightarrow \mathbb{R}^+$  is submodular if

$$f(X \cup \{v\}) - f(X) \geq f(Y \cup \{v\}) - f(Y), \quad \forall X \subseteq Y \subseteq V, v \in V \setminus Y.$$

We can assume that  $f$  is not non-increasing, and for monotone  $f$ , we can assume  $f(\emptyset) = 0$ . To perform analysis, we define the function’s monotonicity approximation term similar to [24], but only for subsets of a certain size.

**Definition 2.** For a function  $f : 2^V \rightarrow \mathbb{R}^+$ , its monotonicity approximation term with respect to a parameter  $j$  is

$$\epsilon_j = \max_{X, v: |X| < j} \{f(X \setminus \{v\}) - f(X)\},$$

for  $j > 0$  and  $\epsilon_0 = 0$ .

It is clear that  $\epsilon_j$  is non-negative, non-decreasing with increasing  $j$ , and  $f$  is monotone iff  $\epsilon_n = 0$ . Additionally, for monotone non-submodular  $f$ , we use submodularity ratio which quantifies how close  $f$  is to being modular. In particular, we simplify the definition [11] which measures the severity of the diminishing return effect.

**Definition 3.** For a monotone function  $f : 2^V \rightarrow \mathbb{R}^+$ , its submodularity ratio with respect to two parameters  $i, j \geq 1$  is

$$\gamma_{i,j} = \min_{|X| < i, |L| \leq j, X \cap L = \emptyset} \frac{\sum_{v \in L} [f(X \cup \{v\}) - f(X)]}{f(X \cup L) - f(X)},$$

for  $i > 0$  and  $\gamma_{0,j} = \gamma_{1,j}$ .

It can be seen that  $\gamma_{i,j}$  is non-negative, non-increasing with increasing  $i$  and  $j$ , and  $f$  is submodular iff  $\gamma_{i,j} \geq 1$  for all  $(i, j)$ .

For the purpose of analysis, we denote  $d = \sum_i d_i$ ,  $\bar{d} = \min_i \{d_i\}$ , and  $OPT$  the optimal solution; we have  $\bar{d} \leq d/k$  and  $|OPT| \leq d$ . We evaluate the algorithm’s performance via  $f(X^*)/f(OPT)$  where  $X^*$  is the algorithm’s output. Furthermore, we use the black-box oracle model to evaluate run time, hence our results are based on numbers of oracle calls.

## 2.2 Algorithms Descriptions

A popular baseline method to solve hard problems is greedy heuristics. A simple deterministic GREEDY variant has been studied for this problem [15]. It starts with the empty solution, and in each iteration adds the feasible remaining element in  $V$  that increases  $f$  value the most. It terminates when there is no remaining feasible elements that yield positive gains. This algorithm extends the GREEDY algorithms in [32] to partition matroids constraints. Note that at iteration  $k$ , GREEDY calls the oracle  $n - k + 1$  times, so its run time is  $\mathcal{O}(dn)$ . According to [15], it achieves  $(1 - e^{-\bar{d}/d})/\alpha$  approximation ratio when  $f$  is submodular, and  $(1 - e^{-\alpha(1-\alpha)\bar{d}/d})/\alpha$  approximation ratio when  $f$  is non-decreasing subadditive, with  $\alpha$  being the curvature of  $f$ .

---

**Algorithm 1.** GSEMO algorithm

---

**Input:** a problem instance:  $(f, B, D)$   
**Parameter:** the number of iterations  $T \geq 0$   
**Output:** a feasible solution  $x \in \{0, 1\}^n$   
 $x \leftarrow 0, P \leftarrow \{x\}$   
**while**  $t < T$  **do**  
    Randomly sample a solution  $y$  from  $P$   
    Generate  $y'$  by flipping each bit of  $y$  independently with probability  $1/n$   
    **if**  $\nexists x \in P, x \succ y'$  **then**  
         $P \leftarrow (P \setminus \{x \in P, y' \succeq x\}) \cup \{y'\}$   
    **end if**  
**end while**  
**return**  $\operatorname{argmax}_{x \in P} f_1(x)$

---

On the other hand, GSEMO [16, 17, 25], also known as POMC [34], is a well-known simple Pareto optimization approach for constrained single-objective optimization problems. It has been shown to outperform the generalized greedy algorithm in overcoming local optima [34]. To use GSEMO with partition matroid constraints, the problem is reformulated as a bi-objective problem

$$\operatorname{maximize}_{X \subseteq V} (f_1(X), f_2(X)),$$

where

$$f_1(X) = \begin{cases} -\infty, & \exists i, |B_i \cap X| > d_i \\ f(X), & \text{otherwise} \end{cases}, f_2(X) = -|X|.$$

GSEMO optimizes two objectives simultaneously, using the dominance relation between solutions, which is common in Pareto optimization approaches. By definition, solution  $X_1$  dominates  $X_2$  ( $X_1 \succeq X_2$ ) iff  $f_1(X_1) \geq f_1(X_2)$  and  $f_2(X_1) \geq f_2(X_2)$ . The dominance relation is strict ( $X_1 \succ X_2$ ) iff  $f_1(X_1) > f_1(X_2)$  or  $f_2(X_1) > f_2(X_2)$ . Intuitively, dominance relation formalizes the notion of “better” solution in multi-objective contexts. Solutions that don’t dominate any other present a trade-off between objectives to be optimized.

The second objective in GSEMO is typically formulated to promote solutions that are “further” from being infeasible. The intuition is that for those solutions, there is more room for feasible modification, thus having more potential of becoming very good solutions. For the problem of interest, one way of measuring “distance to infeasibility” for some solution  $X$  is counting the number of elements in  $V \setminus X$  that can be added to  $X$  before it is infeasible. The value then would be  $d - |X|$ , which is the same as  $f_2(X)$  in practice. Another way is counting the minimum number of elements in  $V \setminus X$  that need to be added to  $X$  before it is infeasible. The value would then be  $\min_i \{d_i - |B_i \cap X|\}$ . The former approach is chosen for simplicity and viability under weaker assumption about the oracle.



On the other hand, the first objective aims to present the canonical evolutionary pressure based on objective values. Additionally,  $f_1$  also discourages all infeasible solutions, which is different from the formulation in [34] that allows some degree of infeasibility. This is because for  $k > 1$ , there can be some infeasible solution  $Y$  where  $|Y| \leq d$ . If  $f_1(Y)$  is very high, it can dominate many good feasible solutions, and may prevent acceptance of global optimal solutions into the population. Furthermore, restricting to only feasible solutions decreases the maximum population size, which can improve convergence performance. It is clear the population size is at most  $d+1$ . These formulations of the two objective functions are identical to the ones in [17] when  $k = 1$ .

In practice, set solutions are represented in GSEMO as binary sequences, where with  $V = \{v_1, \dots, v_n\}$  the following bijective mapping is implicitly assumed

$$g : 2^V \rightarrow \{0, 1\}^n, \quad g(X)_i = \begin{cases} 0, & v_i \notin X, \\ 1, & v_i \in X \end{cases}.$$

This representation of set is useful in evolutionary algorithms since genetic bit operators are compatible. GSEMO operates on the bit sequences, and the fitness function is effectively a pseudo-Boolean function  $f \circ g^{-1}$ . It starts with initial population of a single empty solution. In each iteration, a new solution is generated by random parent selection and bit flip mutation. Then the elitist survival selection mechanism removes dominated solutions from the population, effectively maintaining a set of known Pareto-optimal solutions. The algorithm terminates when the number of iteration reaches some predetermined limit. The procedure is described in Algorithm 1. We choose empty set as the initial solution, similar to [34] and different from [17], to simplify the analysis and stabilize theoretical performance. Note that GSEMO calls the oracle once per iteration to evaluate a new solution, so its run time is identical to the number of iterations.

### 3 Approximation Guarantees

We derive an approximation guarantee for GSEMO on maximizing a general submodular function under partition matroid constraints. According to the analysis for GREEDY [15], we can assume there are  $d$  “dummy” elements with zero marginal contribution. For all feasible solutions  $X \subseteq V$  where  $|X| < \bar{d}$ , let  $v_X^* = \operatorname{argmax}_{v \in V \setminus X} f_1(X \cup \{v\})$  be the feasible greedy addition to  $X$ , we can derive the following result from Lemma 2 in [36], using  $f(OPT \cup X) \geq f(OPT) - j\epsilon_{d+j+1}$ .

**Lemma 1** ([36]). *Let  $f$  be a submodular function and  $\epsilon_d$  be defined in Definition 2, for all feasible solutions  $X \subseteq V$  such that  $|X| = j < \bar{d}$*

$$f(X \cup \{v_X^*\}) - f(X) \geq \frac{1}{d}[f(OPT) - f(X) - j\epsilon_{d+j+1}].$$

With this lemma, we can prove the following result where  $P_t$  denotes the population at iteration  $t$ .

**Theorem 1.** *For the problem of maximizing a submodular function  $f$  under partition matroid constraints, GSEMO generates in expected run time  $\mathcal{O}(d^2n/k)$  a solution  $X \subseteq V$  such that*

$$f(X) \geq \left(1 - e^{-\bar{d}/d}\right) [f(OPT) - (\bar{d} - 1)\epsilon_{d+\bar{d}}].$$

*Proof.* Let  $S(X, j)$  be a statement  $|X| \leq j \wedge f(X) \geq \left[1 - \left(1 - \frac{1}{d}\right)^j\right] [f(OPT) - (j - 1)\epsilon_{d+j}]$ , and  $J_t = \max\{i \in [0, \bar{d}] \mid \exists X \in P_t, S(X, i)\}$ , it is clear that  $S(\emptyset, 0)$  holds, so  $J_0 = 0$  and  $J_t$  is well-defined for all  $t \geq 0$  since the empty solution is never dominated.

Assuming  $J_t = i$  at some  $t$ , let  $\bar{X} \in P_t$  such that  $S(\bar{X}, i)$  holds. If  $\bar{X}$  is not dominated and removed from  $P_{t+1}$ , then  $J_{t+1} \geq J_t$ . Otherwise, there must be some  $Y \in P_{t+1}$  such that  $|Y| \leq |\bar{X}|$  and  $f(Y) \geq f(\bar{X})$ . This implies  $S(Y, i)$ , so  $J_{t+1} \geq J_t$ . Therefore,  $J_t$  is never decreased as  $t$  progresses. Let  $X' = \bar{X} \cup \{u_{\bar{X}}^*\}$ , Lemma 1 implies

$$\begin{aligned} f(X') &\geq \frac{1}{d}f(OPT) + \left(1 - \frac{1}{d}\right) \left[1 - \left(1 - \frac{1}{d}\right)^i\right] [f(OPT) - (i - 1)\epsilon_{d+i}] \\ &\quad - \frac{i}{d}\epsilon_{d+i+1} \\ &\geq \left[1 - \left(1 - \frac{1}{d}\right)^{i+1}\right] [f(OPT) - i\epsilon_{d+i+1}]. \end{aligned}$$

The second inequality uses  $0 \leq \epsilon_{d+i} \leq \epsilon_{d+i+1}$ . The probability that GSEMO selects  $\bar{X}$  is at least  $\frac{1}{d+1}$ , and the probability of generating  $X'$  by mutating  $\bar{X}$  is at least  $\frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en}$ . Furthermore,  $S(X', i + 1)$  holds as shown, so  $J_{t+1} \geq i + 1$  if  $X' \in P_{t+1}$ . Since  $i \leq \bar{d} - 1$  and  $t \geq 0$  are chosen arbitrarily, this means

$$E[J_{t+1} - J_t \mid J_t \in [0, \bar{d} - 1]] \geq \frac{1}{en(d + 1)}, \quad \forall t \geq 0.$$

Therefore, the Additive Drift Theorem [27] implies the expected number of iterations for  $J_t$  to reach  $\bar{d}$  from 0 is at most  $e\bar{d}n(d + 1)$ . When  $J_t = \bar{d}$ ,  $P_t$  must contain a feasible solution  $Z$  such that

$$f(Z) \geq \left(1 - e^{-\bar{d}/d}\right) [f(OPT) - (\bar{d} - 1)\epsilon_{d+\bar{d}}].$$

Therefore, GSEMO generates such a solution in expected run time at most  $e\bar{d}n(d + 1) = \mathcal{O}(d^2n/k)$ . □

In case of a single cardinality constraint ( $\bar{d} = d$ ), this approximation guarantee is at least as tight as the one for GSEMO-C in [36]. If monotonicity of  $f$  is further assumed, the result is equivalent to the one for GSEMO in [17]. Additionally,

the presence of  $\epsilon_d$  suggests that the non-monotonicity of  $f$  does not necessarily worsen the approximation guarantee when negative marginal gains are absent from all GSEMO's possible solutions (i.e. cannot decrease objective values by adding an element).

As an extension beyond submodularity instead of monotonicity, we provide another proof of the approximation guarantee for GSEMO on the problems of maximizing monotone functions under the same constraints. Without loss of generality, we can assume that  $f$  is normalized, meaning  $f(\emptyset) = 0$ . We make use of the following inequality, derived from Lemma 1 in [35].

**Lemma 2.** *Let  $f$  be a monotone function and  $\gamma_{i,j}$  be defined in Definition 3, for all feasible solutions  $X \subseteq V$  such that  $|X| = j < \bar{d}$*

$$f(X \cup \{v_X^*\}) - f(X) \geq \frac{\gamma_{j+1,d}}{d} [f(OPT) - f(X)].$$

Using this lemma, we similarly prove the following result.

**Theorem 2.** *For the problem of maximizing a monotone function under partition matroid constraints, GSEMO with expected run time  $\mathcal{O}(d^2n/k)$  generates a solution  $X \subseteq V$  such that*

$$f(X) \geq \left(1 - e^{-\gamma_{\bar{d},\bar{d}}\bar{d}/d}\right) f(OPT).$$

*Proof.* Let  $S(X, j)$  be a statement  $|X| \leq j \wedge f(X) \geq \left[1 - \left(1 - \frac{\gamma_{j,d}}{d}\right)^j\right] f(OPT)$ ,

and  $J_t = \max\{i \in [0, \bar{d}] | \exists X \in P_t, S(X, i)\}$ , it is clear that  $S(\emptyset, 0)$  holds, so  $J_0 = 0$  and  $J_t$  is well-defined for all  $t \geq 0$  since the empty solution is never dominated.

Assuming  $J_t = i$  at some  $t$ , there must be  $\bar{X} \in P_t$  such that  $S(\bar{X}, i)$  holds. If  $\bar{X}$  is not dominated and removed from  $P_{t+1}$ , then  $J_{t+1} \geq J_t$ . Otherwise, there must be some  $Y \in P_{t+1}$  such that  $|Y| \leq |\bar{X}|$  and  $f(Y) \geq f(\bar{X})$ . This implies  $S(Y, i)$ , so  $J_{t+1} \geq J_t$ . Therefore,  $J_t$  is never decreased as  $t$  progresses. Let  $X' = \bar{X} \cup \{v_{\bar{X}}^*\}$ , Lemma 2 implies

$$\begin{aligned} f(X') &\geq \frac{\gamma_{i+1,d}}{d} f(OPT) + \left(1 - \frac{\gamma_{i+1,d}}{d}\right) \left[1 - \left(1 - \frac{\gamma_{i,d}}{d}\right)^i\right] f(OPT) \\ &\geq \left[1 - \left(1 - \frac{\gamma_{i+1,d}}{d}\right)^{i+1}\right] f(OPT). \end{aligned}$$

The second inequality uses  $\gamma_{i,d} \geq \gamma_{i+1,d}$ . The probability that GSEMO selects  $\bar{X}$  is at least  $\frac{1}{\bar{d}+1}$ , and the probability of generating  $X'$  by mutating  $\bar{X}$  is at least  $\frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en}$ . Furthermore,  $S(X', i+1)$  holds as shown, so  $J_{t+1} \geq i+1$ . Since  $i \leq \bar{d} - 1$  and  $t \geq 0$  are chosen arbitrarily, this means

$$E[J_{t+1} - J_t | J_t \in [0, \bar{d} - 1]] \geq \frac{1}{en(d+1)}, \quad \forall t \geq 0.$$

Therefore, according to the Additive Drift Theorem [27], the expected number of iterations for  $J_t$  to reach  $\bar{d}$  from 0 is at most  $e\bar{d}n(d + 1)$ . When  $J_t = \bar{d}$ ,  $P_t$  must contain a feasible solution  $Z$  such that

$$f(Z) \geq \left[ 1 - \left( 1 - \frac{\gamma_{\bar{d},d}}{d} \right)^{\bar{d}} \right] f(OPT) \geq \left( 1 - e^{-\gamma_{\bar{d},d}\bar{d}/d} \right) f(OPT).$$

Therefore, GSEMO generates such a solution in expected run time at most  $e\bar{d}n(d + 1) = \mathcal{O}(d^2n/k)$ . □

Compared to the results in [34], it is reasonable to assume that restricting GSEMO’s population to only allow feasible solutions improves worst-case guarantees. However, it also eliminates the possibility of efficient improvement by modifying infeasible solutions that are very close to very good feasible ones. This might reduce its capacity to overcome local optima.

## 4 Experimental Investigations

We compare GSEMO and GREEDY on the symmetric submodular Cut maximization problems with randomly generated graphs under varying settings. The experiments are separated into two groups: cardinality constraints ( $k = 1$ ) and general partition matroid constraints ( $k > 1$ ).

### 4.1 Max Cut Problems Setup

Weighted graphs are generated for the experiments based on two parameters: number of vertices (which is  $n$ ) and density. There are 3 values for  $n$ : 50, 100, 200. There are 5 density values: 0.01, 0.02, 0.05, 0.1, 0.2. For each  $n$ -density pair, 30 different weighted graphs – each denoted as  $G = (V, E, c)$  – are generated with the following procedure:

1. Randomly sample  $E$  from  $V \times V$  without replacement, until  $|E| = \lfloor \text{density} \times n^2 \rfloor$ .
2. Assign to  $c(a, b)$  a uniformly random value in  $[0, 1]$  for each  $(a, b) \in E$ .
3. Assign  $c(a, b) = 0$  for all  $(a, b) \notin E$ .

Each graph is then paired with different sets of constraints, and each pairing constitutes a problem instance. This enables observations of changes in outputs on the same graphs under varying constraints. For cardinality constraints,  $d_1 = \{ \frac{n}{4}, \frac{n}{2}, \frac{3n}{4} \}$ , rounded to the nearest integer. Thus, there are 30 problem instances per  $n$ -density- $d_1$  triplet. For partition matroid constraints, the numbers of partitions are  $k = \{ 2, 5, 10 \}$ . The partitions are of the same size, and each element is randomly assigned to a partition. The thresholds  $d_i$  are all set to  $\lceil \frac{n}{2k} \rceil$  since the objective functions are symmetric. Likewise, there are 30 problem instances per  $n$ -density- $k$  triplet.

GSEMO is run on each instance 30 times, and the minimum, mean and maximum results are denoted by  $\text{GSEMO}^-$ ,  $\text{GSEMO}^*$  and  $\text{GSEMO}^+$ , respectively.

The GREEDY algorithm is run until satisfying its stopping condition, while GSEMO is run for  $T = 4n^2$  iterations. Their final achieved objective values are then recorded and analyzed. Note that the run time budget for GSEMO in every setting is smaller than the theoretical bound on average run time in Theorem 1, except for  $(n, k) = (50, 10)$  where it is only slightly larger.

## 4.2 Cut Maximization Under a Cardinality Constraint

The experimental results for cardinality constraint cases are shown in Table 1. Signed-rank U-tests [8] are applied to the outputs, with pairing based on instances. Furthermore, we count the numbers of instances where GSEMO outperforms, ties with, and is outperformed by GREEDY via separate U-tests on individual instances.

Overall, GSEMO on average outperforms GREEDY with statistical significance in most cases. The significance seems to increase, with some noises, with increasing  $n$ , density, and  $d_1$ . This indicates that GSEMO more reliably produces better solutions than GREEDY as the graph's size and density increase. Moreover, in few cases with large  $n$ , GSEMO<sup>-</sup> is higher than GREEDY's with statistical significance.

Additionally, it is indicated that GSEMO\* tend to be closer to GSEMO<sup>+</sup> than GSEMO<sup>-</sup>. This suggests skewed distribution of outputs in each instance toward high values. The implication is that GSEMO is more likely to produce outputs greater than average, than otherwise. It might be an indication that these results are close to the global optima for these instances.

Per instance analyses show high number of ties between GSEMO and GREEDY for small  $n$ , density, and to a lesser extent  $d_1$ . As these increase, the number of GSEMO's wins increases and ends up dominating at  $n = 200$ . This trend coincides with earlier observations, and suggests the difficulty of making improvements in sparse graphs faced by GSEMO where GREEDY heuristic seems more suited. On the other hand, large graph sizes seem to favour GSEMO over GREEDY despite high sparsity, likely due to more local optima present in larger search spaces.

## 4.3 Cut Maximization Under Partition Matroid Constraints

The experimental results for partition matroid constraints cases are shown in Table 2. Notations and statistical test procedure are the same as in Table 1.

Overall, the main trend in cardinality constraint cases is present: GSEMO on average outperforms GREEDY, with increased reliability at larger  $n$  and density. This can be observed in both the average performances and the frequency at which GSEMO beats GREEDY. It seems the effect of this phenomenon is less influenced by variations in  $k$  than it is by variations in  $d_1$  in cardinality constraint cases. Note that the analysis in Theorem 1 only considers bottom-up improvements up to  $|X| = \bar{d}$ . Experimental results likely suggest GSEMO can make similar improvements beyond that point up to  $|X| = d$ .

**Table 1.** Experimental results for cardinality constraints cases. Ranges of final objective values across 30 graphs are shown for each setting. The signed-rank U-tests are used to compare GREEDY’s with GSEMO<sup>-</sup>, GSEMO\*, GSEMO<sup>+</sup> for each setting, pairing by instances, with 95% confidence level. ‘+’ denotes values being significantly greater than GREEDY’s, ‘-’ denotes less, and ‘\*’ denotes no significant difference. Additionally, numbers of losses, wins and ties (L-W-T) GSEMO has over GREEDY are shown, which are determined by separate U-tests on individual instances.

n	Density	d <sub>1</sub>	GREEDY			GSEMO <sup>-</sup>			GSEMO*			GSEMO <sup>+</sup>			L-W-T
			Min	Max	Stat	Min	Max	Stat	Min	Max	Stat	Min	Max	Stat	
50	0.01	13	6.638	12.71	6.625	12.59	-	6.635	12.7	-	6.638	12.71	*	3-3-24	
		25	6.638	13.27	6.625	13.13	-	6.65	13.25	*	6.706	13.27	+	2-7-21	
		38	6.638	13.27	6.625	13.13	-	6.647	13.24	*	6.706	13.27	+	2-6-22	
	0.02	13	12.16	18.08	12.1	18.08	-	12.15	18.08	*	12.17	18.08	+	6-5-19	
		25	13.5	20.27	13.43	20.17	-	13.47	20.29	+	13.5	20.33	+	8-16-6	
		38	13.5	20.27	13.39	20.23	-	13.47	20.3	+	13.5	20.33	+	7-17-6	
	0.05	13	22.09	29.38	21.69	29.28	-	22.03	29.49	+	22.09	29.51	+	2-17-11	
		25	26.52	37.14	27.18	36.28	-	27.75	36.95	+	28.1	37.14	+	4-20-6	
		38	26.52	37.14	27.39	36.43	-	27.89	37.02	+	28.2	37.14	+	5-21-4	
	0.1	13	38.64	47.14	38.19	46.82	-	38.59	47.12	*	38.69	47.14	+	5-11-14	
		25	46.61	57.93	45.36	57.38	-	46.85	57.85	+	47.28	58.03	+	11-16-3	
		38	46.61	58.08	45.03	57.58	-	46.77	57.95	+	47.28	58.1	+	10-17-3	
	0.2	13	63.13	77.38	63.09	77.01	*	63.26	77.43	+	63.46	77.64	+	2-16-12	
		25	78.89	92.57	79.37	91.68	-	80.61	92.25	+	80.82	92.66	+	5-21-4	
		38	78.89	92.57	79.82	91.78	-	80.62	92.31	+	80.82	92.66	+	5-21-4	
100	0.01	25	24.93	31.88	24.9	31.86	-	25.11	31.89	*	25.36	31.89	+	7-9-14	
		50	27.87	37.79	27.56	37.87	-	28.11	38.07	*	28.67	38.26	+	12-13-5	
		75	27.87	37.79	27.24	37.71	-	28.07	38	+	28.67	38.26	+	11-13-6	
	0.02	25	42.95	53.66	42.95	53.56	-	43.38	53.64	*	43.4	53.66	+	7-10-13	
		50	51.93	66	51.95	65.27	-	52.62	66.14	+	52.78	66.4	+	9-15-6	
		75	51.93	66	51.69	64.31	-	52.6	66.08	+	52.8	66.4	+	9-17-4	
	0.05	25	78.13	94.98	78.09	95.3	-	78.38	95.45	+	78.78	95.49	+	8-17-5	
		50	100.6	120.7	100	119	-	100.9	120.3	+	101.7	120.7	+	7-18-5	
		75	100.7	120.7	99.23	118.9	-	100.8	120.4	+	101.9	120.7	+	7-17-6	
	0.1	25	138.9	155.3	138.5	155	-	139.1	156	+	139.6	156.2	+	2-20-8	
		50	178.4	197.8	177	198.3	*	178	199.9	+	179.6	200.6	+	7-17-6	
		75	178.4	197.8	176.6	198.6	*	178	199.9	+	179.4	200.6	+	6-17-7	
	0.2	25	224.1	249.2	222.8	249.4	*	224	250	+	225.6	250.2	+	4-22-4	
		50	297.6	325.1	297.8	323.9	*	300.9	325.8	+	302.7	326.4	+	6-20-4	
		75	297.6	325.1	298	323.9	-	300.4	325.8	+	303.2	326.4	+	6-19-5	
200	0.01	50	85.54	96.11	84.98	96.05	*	85.58	96.3	+	85.84	96.52	+	7-20-3	
		100	103.1	118.4	103.5	118.5	-	104.6	120.1	+	104.9	121.4	+	4-23-3	
		150	103.1	118.4	103.6	118.6	-	104.7	120	+	105.1	121.4	+	4-21-5	
	0.02	50	139.1	159.3	140.6	158.8	*	141.8	159.2	+	142.2	159.3	+	8-19-3	
		100	173.3	198.2	175.6	197.5	*	177.3	198.5	+	179.4	199.3	+	2-25-3	
		150	173.3	198.2	174.8	197.2	*	177	198.4	+	178.9	199.5	+	2-23-5	
	0.05	50	275.9	311.8	277.8	311.4	+	278.8	312.3	+	280	313	+	0-28-2	
		100	357	400.6	364.4	402.6	*	367.1	405	+	369.7	406.7	+	1-28-1	
		150	357	400.6	364.2	402.7	*	366.8	405.2	+	369.9	407.1	+	0-27-3	
	0.1	50	489.8	534	490.6	533.7	*	492.6	534	+	493.4	534	+	6-22-2	
		100	647.8	680.5	643.7	679.3	-	649.5	683.7	+	653	686.4	+	2-24-4	
		150	648	680.5	642.3	680.9	*	649.6	683.8	+	652.2	687	+	2-22-6	
	0.2	50	866.9	921.8	867.9	920.6	*	871.5	921.6	+	873.4	921.8	+	1-26-3	
		100	1120	1182	1120	1181	+	1125	1188	+	1128	1192	+	0-29-1	
		150	1120	1182	1122	1181	+	1125	1189	+	1129	1193	+	0-30-0	

**Table 2.** Experimental results for partition matroid constraints cases. Ranges of final objective values across 30 graphs are shown for each setting. The signed-rank U-tests are used to compare GREEDY's with GSEMO<sup>-</sup>, GSEMO<sup>\*</sup>, GSEMO<sup>+</sup> for each setting, pairing by instances, with 95% confidence level. '+' denotes values being significantly greater than GREEDY's, '-' denotes less, and '\*' denotes no significant difference. Additionally, numbers of losses, wins and ties (L-W-T) GSEMO has over GREEDY are shown, which are determined by separate U-tests on individual instances.

n	Density	k	GREEDY			GSEMO <sup>-</sup>			GSEMO <sup>*</sup>			GSEMO <sup>+</sup>			L-W-T
			Min	Max	Stat	Min	Max	Stat	Min	Max	Stat	Min	Max	Stat	
50	0.01	2	6.638	13.27	6.625	13.15	-	6.65	13.26	*	6.706	13.27	+	1-7-22	
		5	6.638	13.27	6.625	13.13	-	6.654	13.25	*	6.706	13.27	+	3-7-20	
		10	6.638	13.27	6.625	13.15	-	6.65	13.25	*	6.706	13.27	+	1-7-22	
	0.02	2	13.5	20.27	13.32	20.24	-	13.47	20.28	+	13.5	20.33	+	7-16-7	
		5	13.5	19.69	13.14	19.53	-	13.4	19.65	+	13.5	19.69	+	7-13-10	
		10	13.11	20.27	12.94	20.07	-	13.09	20.28	+	13.11	20.33	+	7-14-9	
	0.05	2	25.8	37.14	26.63	36.82	-	27.07	37.07	+	27.42	37.14	+	5-18-7	
		5	25.78	36.75	26.49	35.86	-	27.04	36.58	+	27.66	36.88	+	8-18-4	
		10	25.9	36.83	25.73	35.28	-	27.38	36.17	+	28.09	36.83	+	6-20-4	
	0.1	2	46.61	57.93	44.87	56.97	-	46.72	57.7	+	47.28	58.05	+	9-18-3	
		5	45.91	56.69	45.3	55.85	-	46.23	56.4	+	46.81	56.94	+	8-18-4	
		10	46.59	56.46	44.13	55.73	-	46.41	56.6	+	47.08	57.13	+	6-18-6	
0.2	2	78.89	92.5	79.16	91.53	-	80.54	92.18	+	80.82	92.66	+	6-19-5		
	5	74.37	91.89	74.96	90.83	-	76.44	91.61	+	77.39	91.89	+	6-22-2		
	10	75.85	92.57	75.62	90.86	-	76.7	92.08	+	77.97	92.51	+	4-21-5		
100	0.01	2	27.87	37.79	27.69	37.7	-	28.15	37.99	+	28.67	38.26	+	10-14-6	
		5	27.87	37.79	27.78	37.65	-	28.1	38	+	28.67	38.26	+	10-14-6	
		10	27.87	36.81	27.59	36.51	-	27.88	36.91	*	28.31	37.14	+	11-12-7	
	0.02	2	51.92	66	51.64	65.44	-	52.55	66.17	+	52.79	66.4	+	10-15-5	
		5	51.62	65.82	51.57	65.48	-	52.41	66.04	*	52.71	66.22	+	8-13-10	
		10	51.69	63.41	51.38	62.63	-	51.9	63.29	*	52.19	63.75	+	8-13-9	
	0.05	2	100.1	120.7	99.68	119.3	-	100.9	120.2	+	101.7	120.7	+	8-17-5	
		5	99.61	119.5	98.19	117.4	-	99.86	119	+	100.6	119.7	+	7-17-6	
		10	97	116.6	95.82	116	-	98.14	117.6	+	99.37	118.3	+	5-18-7	
	0.1	2	177.6	197.8	176.8	198.6	*	177.6	199.9	+	178.4	200.7	+	6-19-5	
		5	173.5	196.6	172.6	197.6	-	174.3	199.1	+	175.8	200.2	+	2-21-7	
		10	173.3	192.9	171.7	193.9	-	173.6	195.9	*	175.3	198.2	+	12-12-6	
0.2	2	294.2	325.1	294.8	324.8	*	297.4	325.9	+	301.6	326.4	+	6-22-2		
	5	292.7	320.5	293.2	318.5	*	297.3	321.8	+	299.5	323.7	+	3-23-4		
	10	288.3	322.7	288.2	317.3	-	292.3	321.6	+	296.3	323.8	+	5-22-3		
200	0.01	2	103.1	118.4	103.6	118.9	*	104.6	120.2	+	105.1	121.6	+	4-22-4	
		5	103.1	118.4	103.4	118.5	-	104.2	119.9	+	104.7	121.4	+	3-20-7	
		10	102.7	117.4	102.6	117.2	-	104	118.1	+	104.7	119.3	+	4-21-5	
	0.02	2	173.3	198.2	174.8	197.2	*	176.5	198.5	+	178.6	199.5	+	3-22-5	
		5	172.8	196.7	173.7	195.7	-	176.4	197.6	+	179.1	198.6	+	2-20-8	
		10	172.5	193.3	173.7	193.9	*	176.1	195.6	+	177.9	196.8	+	1-23-6	
	0.05	2	356.4	400.6	362.7	401.6	*	366.2	404.8	+	369.1	406.1	+	2-28-0	
		5	353.7	399.4	359.6	400.5	*	363.5	403	+	365.5	404.7	+	1-28-1	
		10	352.9	394.3	355.1	396.7	*	360.1	399.4	+	363.2	401.5	+	1-26-3	
	0.1	2	645.2	680.5	642.8	679	*	647.4	682.9	+	650.7	685.5	+	3-25-2	
		5	641.7	678.7	637.4	676.2	-	643.3	680.6	+	647.4	685.2	+	4-20-6	
		10	637.2	669.8	632.3	667.4	-	641.5	672.7	+	645.1	677.3	+	2-24-4	
0.2	2	1119	1182	1118	1183	+	1123	1187	+	1128	1193	+	0-30-0		
	5	1117	1178	1116	1173	*	1121	1179	+	1125	1184	+	0-29-1		
	10	1105	1170	1111	1175	*	1117	1181	+	1122	1188	+	2-28-0		

Additionally, the outputs of both algorithms are generally decreased with increased  $k$ , due to restricted feasible solution spaces. There are few exceptions which could be attributed to noises from random partitioning since they occur in both GREEDY's and GSEMO's simultaneously. Furthermore, the variation in  $k$  seems to slightly affect the gaps between GSEMO's and GREEDY's, which are somewhat smaller at higher  $k$ . It seems to support the notion that GREEDY performs well in small search spaces while GSEMO excels in large search spaces. This coincides with the observations in the cardinality constraint cases, and explains the main trend.

## 5 Conclusion

In this work, we consider the problem of maximizing a set function under partition matroid constraints, and analyze GSEMO's approximation performance on such problems. Theoretical performance guarantees are derived for GSEMO in cases of submodular objective functions, and monotone objective functions. We show that GSEMO guarantees good approximation quality within polynomial expected run time in both cases. Additionally, experiments with Max Cut instances generated from varying settings have been conducted to gain insight on its empirical performance, based on comparison against simple GREEDY's. The results show that GSEMO generally outperforms GREEDY within quadratic run time, particularly when the feasible solution space is large.

**Acknowledgements.** The experiments were run using the HPC service provided by the University of Adelaide.

## References

1. Auger, A., Doerr, B. (eds.): Theory of Randomized Search Heuristics: Foundations and Recent Developments. World Scientific Publishing Co., Inc., Singapore (2011)
2. Balcan, M.F., Blum, A., Mansour, Y.: Item pricing for revenue maximization. In: Proceedings of the 9th ACM Conference on Electronic Commerce, EC 2008, pp. 50–59. ACM, New York (2008). <https://doi.org/10.1145/1386790.1386802>
3. Bhawalkar, K., Roughgarden, T.: Welfare guarantees for combinatorial auctions with item bidding. In: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, pp. 700–709. Society for Industrial and Applied Mathematics, Philadelphia (2011). <https://doi.org/10.1137/1.9781611973082.55>
4. Bian, A.A., Buhmann, J.M., Krause, A., Tschitschek, S.: Guarantees for greedy maximization of non-submodular functions with applications. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, vol. 70, pp. 498–507. JMLR.org (2017)
5. Buchbinder, N., Feldman, M., Garg, M.: Deterministic  $(1/2 + \epsilon)$ -approximation for submodular maximization over a matroid. In: Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, pp. 241–254. Society for Industrial and Applied Mathematics, Philadelphia (2019)



6. Chekuri, C., Kumar, A.: Maximum coverage problem with group budget constraints and applications. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) APPROX/RANDOM -2004. LNCS, vol. 3122, pp. 72–83. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-27821-4\\_7](https://doi.org/10.1007/978-3-540-27821-4_7)
7. Conforti, M., Cornuéjols, G.: Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discret. Appl. Math.* **7**(3), 251–274 (1984). [https://doi.org/10.1016/0166-218X\(84\)90003-9](https://doi.org/10.1016/0166-218X(84)90003-9)
8. Corder, G.W., Foreman, D.I.: *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Wiley, Hoboken (2009)
9. Cornuejols, G., Fisher, M.L., Nemhauser, G.L.: Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Manag. Sci.* **23**(8), 789–810 (1977). <https://doi.org/10.1287/mnsc.23.8.789>
10. Călinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.* **40**(6), 1740–1766 (2011). <https://doi.org/10.1137/080733991>
11. Das, A., Kempe, D.: Submodular meets spectral: greedy algorithms for subset selection, sparse approximation and dictionary selection. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML 2011, Omnipress, Madison, WI, USA, pp. 1057–1064 (2011)
12. Doerr, B., Neumann, F. (eds.): *Theory of Evolutionary Computation - Recent Developments in Discrete Optimization*. Natural Computing Series. Springer, Cham (2020). <https://doi.org/10.1007/978-3-030-29414-4>
13. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. *SIAM J. Comput.* **40**(4), 1133–1153 (2011). <https://doi.org/10.1137/090779346>
14. Fleischer, L., Goemans, M.X., Mirrokni, V.S., Sviridenko, M.: Tight approximation algorithms for maximum general assignment problems. In: Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm, SODA 2006, pp. 611–620. Society for Industrial and Applied Mathematics, Philadelphia (2006). <https://doi.org/10.1145/1109557.1109624>
15. Friedrich, T., Göbel, A., Neumann, F., Quinzan, F., Rothenberger, R.: Greedy maximization of functions with bounded curvature under partition matroid constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 2272–2279 (2019). <https://doi.org/10.1609/aaai.v33i01.33012272>
16. 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). [https://doi.org/10.1162/EVCO\\_a.00003](https://doi.org/10.1162/EVCO_a.00003)
17. Friedrich, T., Neumann, F.: Maximizing submodular functions under matroid constraints by evolutionary algorithms. *Evol. Comput.* **23**(4), 543–558 (2015). [https://doi.org/10.1162/EVCO\\_a.00159](https://doi.org/10.1162/EVCO_a.00159)
18. 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). <https://doi.org/10.1145/227683.227684>
19. Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM* **48**(4), 761–777 (2001). <https://doi.org/10.1145/502090.502096>
20. Jansen, T.: *Analyzing Evolutionary Algorithms - The Computer Science Perspective*. Natural Computing Series. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-17339-4>

21. Jegelka, S., Bilmes, J.: Submodularity beyond submodular energies: coupling edges in graph cuts. In: CVPR 2011, pp. 1897–1904 (2011). <https://doi.org/10.1109/CVPR.2011.5995589>
22. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 137–146. ACM, New York (2003). <https://doi.org/10.1145/956750.956769>
23. Krause, A., Guestrin, C.: Submodularity and its applications in optimized information gathering. *ACM Trans. Intell. Syst. Technol.* **2**(4), 32:1–32:20 (2011). <https://doi.org/10.1145/1989734.1989736>
24. Krause, A., Singh, A., Guestrin, C.: Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.* **9**, 235–284 (2008). <https://doi.org/10.1145/1390681.1390689>
25. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. *IEEE Trans. Evol. Comput.* **8**(2), 170–182 (2004). <https://doi.org/10.1109/TEVC.2004.823470>
26. Lee, J., Mirrokni, V.S., Nagarajan, V., Sviridenko, M.: Non-monotone submodular maximization under matroid and knapsack constraints. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 323–332. ACM, New York (2009). <https://doi.org/10.1145/1536414.1536459>
27. Lengler, J.: Drift analysis. CoRR abs/1712.00964 (2017). <http://arxiv.org/abs/1712.00964>
28. Lin, H., Bilmes, J.: Multi-document summarization via budgeted maximization of submodular functions. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT 2010, pp. 912–920. Association for Computational Linguistics, Cambridge (2010)
29. Lin, H., Bilmes, J.: A class of submodular functions for document summarization. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, HLT 2011, pp. 510–520. Association for Computational Linguistics, Portland (2011)
30. Liu, Y., Wei, K., Kirchhoff, K., Song, Y., Bilmes, J.: Submodular feature selection for high-dimensional acoustic score spaces. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 7184–7188 (2013). <https://doi.org/10.1109/ICASSP.2013.6639057>
31. Lovász, L.: Submodular functions and convexity. In: Bachem, A., Korte, B., Grotschel, M. (eds.) *Mathematical Programming the State of the Art*, pp. 235–257. Springer, Heidelberg (1983). [https://doi.org/10.1007/978-3-642-68874-4\\_10](https://doi.org/10.1007/978-3-642-68874-4_10)
32. Nemhauser, G.L., Wolsey, L.A.: Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.* **3**(3), 177–188 (1978). <https://doi.org/10.1287/moor.3.3.177>
33. Neumann, F., Witt, C.: *Bioinspired Computation in Combinatorial Optimization*. Natural Computing Series. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-16544-3>
34. Qian, C., Shi, J.C., Yu, Y., Tang, K.: On subset selection with general cost constraints. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 2613–2619 (2017). <https://doi.org/10.24963/ijcai.2017/364>
35. Qian, C., Shi, J.C., Yu, Y., Tang, K., Zhou, Z.H.: Parallel pareto optimization for subset selection. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, pp. 1939–1945. AAAI Press (2016)

36. Qian, C., Yu, Y., Tang, K., Yao, X., Zhou, Z.H.: Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artif. Intell.* **275**, 279–294 (2019). <https://doi.org/10.1016/j.artint.2019.06.005>
37. Queyranne, M.: A combinatorial algorithm for minimizing symmetric submodular functions. In: *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1995*, pp. 98–101. Society for Industrial and Applied Mathematics, Philadelphia (1995)
38. Stobbe, P., Krause, A.: Efficient minimization of decomposable submodular functions. In: *Proceedings of the 23rd International Conference on Neural Information Processing Systems, NIPS 2010*, vol. 2, pp. 2208–2216. Curran Associates Inc., New York (2010)
39. Syrgkanis, V., Tardos, É.: Composable and efficient mechanisms. In: *Proceedings of the 45th Annual ACM Symposium on Theory of Computing, STOC 2013*, pp. 211–220. ACM, New York (2013). <https://doi.org/10.1145/2488608.2488635>
40. Vondrák, J.: Submodularity and curvature: the optimal algorithm. *RIMS Kôkyûroku Bessatsu B23*, pp. 253–266 (2010)
41. Wei, K., Iyer, R., Bilmes, J.: Submodularity in data subset selection and active learning. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning, ICML 2015*, vol. 37, pp. 1954–1963. JMLR.org (2015)
42. Zhou, Z., Yu, Y., Qian, C.: *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, Singapore (2019). <https://doi.org/10.1007/978-981-13-5956-9>



# Lower Bounds for Non-elitist Evolutionary Algorithms via Negative Multiplicative Drift

Benjamin Doerr<sup>(✉)</sup>

Laboratoire d'Informatique (LIX), CNRS, École Polytechnique,  
Institut Polytechnique de Paris, Palaiseau, France  
doerr@lix.polytechnique.fr

**Abstract.** A decent number of lower bounds for non-elitist population-based evolutionary algorithms has been shown by now. Most of them are technically demanding due to the (hard to avoid) use of negative drift theorems – general results which translate an expected progress away from the target into a high hitting time.

We propose a simple negative drift theorem for multiplicative drift scenarios and show that it can simplify existing analyses. We discuss in more detail Lehre's (PPSN 2010) *negative drift in populations* method, one of the most general tools to prove lower bounds on the runtime of non-elitist mutation-based evolutionary algorithms for discrete search spaces. Together with other arguments, we obtain an alternative and simpler proof, which also strengthens and simplifies this method. In particular, now only three of the five technical conditions of the previous result have to be verified. The lower bounds we obtain are explicit instead of only asymptotic. This allows to compute concrete lower bounds for concrete algorithms, but also enables us to show that super-polynomial runtimes appear already when the reproduction rate is only a  $(1 - \omega(n^{-1/2}))$  factor below the threshold. As one particular result, we apply this method and a novel domination argument to show an exponential lower bound for the runtime of the mutation-only simple GA on ONEMAX for arbitrary population size.

**Keywords:** Runtime analysis · Drift analysis · Lower bounds · Population-based algorithms · Theory · Discrete optimization

## 1 Introduction

Lower bounds for the runtimes of evolutionary algorithms are important as they can warn the algorithm user that certain algorithms or certain parameter settings will not lead to good solutions in acceptable time. Unfortunately, the existing results in this direction, for non-elitist algorithms in particular, are very technical. In the case of Lehre's powerful *negative drift in populations* method [24], this also renders the method difficult to use.

One reason for this high complexity is the use of drift analysis, which seems hard to circumvent. Drift analysis [26] is a set of tools that all try to derive useful information on a hitting time (e.g., the first time a solution of a certain quality is found) from information on the expected progress in one iteration. The hope is that the progress in a single iteration can be analyzed with only moderate difficulty and then the drift theorem does the remaining work. While more direct analysis methods exist and have been successfully used for simple algorithms, for population-based algorithms and in particular non-elitist ones, it is hard to imagine that the complicated population dynamics can be captured in proofs not using more advanced tools such as drift analysis.

Drift analysis has been used with great success to prove upper bounds on runtimes of evolutionary algorithms. Tools such as the additive [19], multiplicative [13], and variable drift theorem [22, 28] all allow to easily obtain an upper bound on a hitting time solely from the expected progress in one iteration. Unfortunately, proving matching lower bounds is much harder since here the drift theorems also require additional technical assumptions on the distribution of the progress in one iteration. This is even more true in the case of so-called *negative drift*, where the drift is away from the target and we aim at proving a high lower bound on the hitting time.

In this work, we propose a very simple negative drift theorem for the case of multiplicative drift (Lemma 1). We briefly show that this result can simplify two classic lower bound analyses (Sect. 2).

In more detail, we use the new drift theorem (and some more arguments) to rework Lehre's *negative drift in populations* method [24]. This highly general analysis method allows to show exponential lower bounds on the runtime of a large class of evolutionary algorithms solely by comparing the reproduction rate of individuals in the population with a threshold that depends only on the mutation rate.

The downside of Lehre's method is that both the result and its proof is very technical. To apply the general result (and not the specialization to algorithms using standard bit mutation), five technical conditions need to be verified, which requires the user to choose suitable values for six different constants; these have an influence on the lower bound one obtains. This renders the method of Lehre hard to use. Among the 54 citations to [24] (according to Google scholar on June 9, 2020), only the two works [6, 25] apply this method. To hopefully ease future analyses of negative drift in populations, we revisit this method and obtain the following improvements.

*A Simpler Result:* We manage to show essentially the same lower bounds by only verifying three of the five conditions Lehre was using (Theorem 2 and 3). This also reduces the number of constants one needs to choose from six to four.

*A Non-asymptotic Result:* Our general tool proves explicit lower bounds, that is, free from asymptotic notation or unspecified constants. Consequently, our specialization to algorithms using standard bit mutation (Theorem 4) also gives explicit bounds. This allows one to prove concrete bounds for specific situations

(e.g., that the  $(\mu, \lambda)$  EA with  $\lambda = 2\mu$  needs more than 13 million fitness evaluations to find a unique optimum of problem defined over bit strings of length  $n = 500$ , see the example following Theorem 4) and gives more fine-grained theoretical results (by choosing Lehre's constant  $\delta$  as suitable function of the problems size, we show that a super-polynomial runtime behavior is observed already when the reproduction rate is only a  $(1 - \omega(n^{1/2}))$  factor below the threshold, see Corollary 5).

*A Simple Proof:* Besides the important aspect that a proof guarantees the result to be mathematically correct, an understandable proof can also tell us why a result is correct and give further insights into working principles of algorithms. While every reader will have a different view on how the ideal proof looks like, we felt that Lehre's proof, combining several deep and abstract tools such as multi-type branching processes, eigenvalue arguments, and Hajek's drift theorem [17], does not easily give a broader understanding of the proof mechanics and the working principles of the algorithms analyzed. Our proof, based on a simple potential function argument together with our negative drift theorem, hopefully is more accessible.

Finally, we analyze an algorithm using fitness proportionate selection. The negative drift in populations method is not immediately applicable to such algorithms since it is hard to provide a general unconditional upper bound on the reproduction rate. We show that at all times all search points are at least as good (in the stochastic domination sense) as random search points. This gives a simple proof of an exponential lower bound for the mutation-only simple genetic algorithm with arbitrary population size optimizing the simple ONEMAX benchmark, improving over the mildly sub-exponential lower bound in [29] and the exponential lower bound only for large population sizes in [25].

## 1.1 Related Works

A number of different drift theorems dealing with negative drift have been proven so far, among other, in [18, 23, 27, 31, 32, 34, 35, 39]. They all require some additional assumptions on the distribution of the one-step progress, which makes them non-trivial to use. We refer to [26, Section 2.4.3] for more details. Another approach to negative drift was used in [2, 8, 9]. There the original process was transformed suitably (via an exponential function), but in a way that the drift of the new process still is negative or at most very slowly approaches the target. To this transformed process the lower bound version of the additive drift theorem [19] was applied, which gave large lower bounds since the target, due to the exponential rescaling, now was far from the starting point of the process.

In terms of lower bounds for non-elitist algorithms, besides Lehre's general result [24], the following results for particular algorithms exist (always,  $n$  is the problem size,  $\varepsilon$  can be any positive constant, and  $e \approx 2.718$  is the base of the natural logarithm). Jägerskupper and Storch [21, Theorem 1] showed that the  $(1, \lambda)$  EA with  $\lambda \leq \frac{1}{14} \ln(n)$  is inefficient on any pseudo-Boolean function with a unique optimum. The asymptotically tight condition  $\lambda \leq (1 - \varepsilon) \log_{\frac{e}{e-1}} n$  to

yield a super-polynomial runtime was given by Rowe and Sudholt [35]. Happ, Johannsen, Klein, and Neumann [18] showed that two simple (1+1)-type hill-climbers with fitness proportionate selection cannot optimize efficiently any linear function with positive weights. Neumann, Oliveto, and Witt [29] showed that a mutation-only variant of the simple genetic algorithm (simple GA) with fitness proportionate selection is inefficient on the ONEMAX function when the population size  $\mu$  is at most polynomial, and it is inefficient on any pseudo-Boolean function with unique global optimum when  $\mu \leq \frac{1}{4} \ln(n)$ . The mildly subexponential lower bound for ONEMAX was improved to an exponential lower bound by Lehre [25], but only for  $\mu \geq n^3$ . In a series of remarkable works up to [34], Oliveto and Witt showed that the true simple GA using crossover cannot optimize ONEMAX efficiently when  $\mu \leq n^{\frac{1}{4}-\epsilon}$ . None of these results gives an explicit lower bound or specifies the base of the exponential function. In [2], an explicit lower bound for the runtime of the  $(\mu, \lambda)$  EA is proven (but stated only in the proof of Theorem 3.1 in [2]). Section 3 of [2] bears some similarity with ours, in fact, one can argue that our work extends [2, Section 3] from a particular algorithm to the general class of population-based processes regarded by Lehre [24] (where, naturally, [2] did not have the negative multiplicative drift result and therefore did not obtain bounds that hold with high probability).

## 2 Negative Multiplicative Drift

The following elementary result allows to prove lower bounds on the time to reach a target in the presence of multiplicative drift away from the target. While looking innocent, it has the potential to replace more the complicated lower bound arguments previously used in analyses of non-elitist algorithms such as *simplified drift theorems* ([29, Theorem 1], [33, Theorem 22], [34, Theorem 2]). We discuss this briefly at the end of this section.

**Lemma 1 (Negative multiplicative drift theorem).** *Let  $X_0, X_1, \dots$  be a random process in a finite subset of  $\mathbb{R}_{\geq 0}$ . Assume that there are  $\Delta, \delta > 0$  such that for each  $t \geq 0$ , the following multiplicative drift condition with additive disturbance holds:*

$$E[X_{t+1}] \leq (1 - \delta)E[X_t] + \Delta. \tag{1}$$

*Assume further that  $E[X_0] \leq \frac{\Delta}{\delta}$ . Then the following two assertions hold.*

- For all  $t \geq 0$ ,  $E[X_t] \leq \frac{\Delta}{\delta}$ .
- Let  $M > \frac{\Delta}{\delta}$  and  $T = \min\{t \geq 0 \mid X_t \geq M\}$ . Then for all integers  $L \geq 0$ ,

$$\Pr[T \geq L] \geq 1 - L \frac{\Delta}{\delta M},$$

and  $E[T] \geq \frac{\delta M}{2\Delta} - \frac{1}{2}$ .

The proof is an easy computation of expectations and an application of Markov’s inequality similar to the direct proof of the multiplicative drift theorem in [12]. We do not see a reason why the result should not also hold for processes taking more than a finite number of values, but since we are only interested in the finite setting, we spare us the more complicated world of continuous probability spaces.

*Proof (of Lemma 1).* If  $E[X_t] \leq \frac{\Delta}{\delta}$ , then  $E[X_{t+1}] \leq (1 - \delta)E[X_t] + \Delta \leq (1 - \delta)\frac{\Delta}{\delta} = \frac{\Delta}{\delta}$  by (1). Hence the first claim follows by induction. To prove the second claim, we compute

$$\Pr[T < L] \leq \Pr[X_0 + \dots + X_{L-1} \geq M] \leq \frac{E[X_0 + \dots + X_{L-1}]}{M} \leq \frac{L\Delta}{\delta M},$$

where the middle inequality follows from Markov’s inequality and the fact that the  $X_t$  by assumption are all non-negative. From this estimate, using the shorthand  $s = \lfloor \frac{\delta M}{\Delta} \rfloor$ , we compute  $E[T] = \sum_{t=1}^{\infty} \Pr[T \geq t] \geq \sum_{t=1}^s (1 - \frac{t\Delta}{\delta M}) = s - \frac{1}{2}s(s+1)\frac{\Delta}{\delta M} \geq \frac{\delta M}{2\Delta} - \frac{1}{2}$ , where the first equality is a standard way to express the expectation of a random variable taking non-negative integral values and the last inequality is an elementary computation omitted here. □

We note that in the typical application of this result (as in the proof of Theorem 2 below), we expect to see the condition that for all  $t \geq 0$ ,

$$E[X_{t+1} \mid X_t] \leq (1 - \delta)X_t + \Delta. \tag{2}$$

Clearly, this condition implies (1) by the law of total expectation.

We now argue that our negative multiplicative drift theorem is likely to find applications beyond ours to the negative drift in populations method in the following section. To this aim, we regard two classic lower bound analyses of non-elitist algorithms and point out where our drift theorem would have eased the analysis.

In [29], Neumann, Oliveto, and Witt show that the variant of the simple genetic algorithm (simple GA) not using crossover needs time  $2^{n^{1-O(1/\log \log n)}}$  to optimize the simple ONEMAX benchmark. The key argument in [29] is as follows. The potential  $X_t$  of the population  $P^{(t)}$  in iteration  $t$  is defined as  $X_t = \sum_{x \in P^{(t)}} 8^{\text{ONEMAX}(x)}$ . For this potential, it is shown [29, Lemma 7] that if  $X_t \geq 8^{0.996n}$ , then  $E[X_{t+1}] \leq (1 - \delta)X_t$  for some constant  $\delta > 0$ . By bluntly estimating  $E[X_{t+1}]$  in the case that  $X_t < 8^{0.996n}$ , this bound could easily be extended to  $E[X_{t+1} \mid X_t] \leq (1 - \delta)X_t + \Delta$  for some number  $\Delta$ . This suffices to employ our negative drift theorem and obtain the desired lower bound. Without our drift theorem at hand, in [29] the potential  $Y_t = \log_8(X_t)$  was considered, it was argued that it displays an additive drift away from the target and that  $Y_t$  satisfies certain concentration statements necessary for the subsequent use of a negative drift theorem for additive drift.

A second example using similar techniques, and thus most likely profiting from our drift theorem, is the work of Oliveto and Witt [33, 34] analyzing the



simple GA with crossover optimizing ONEMAX. Due to the use of crossover, this work is much more involved, so without much detail we point the reader interested in the details to the location where we feel that our drift theorem would have eased the analysis. In Lemma 19 of [34], again a multiplicative drift statement (away from the target) is proven. To use a negative drift theorem for additive drift (Theorem 2 in [34]), in the proof of Lemma 20 the logarithm of the original process is regarded. So here again, we feel that a direct application of our drift theorem would have eased the analysis.

### 3 Negative Drift in Populations Revisited

In this section, we use our negative multiplicative drift result and some more arguments to rework Lehre’s negative drift in populations method [24] and obtain Theorem 2 further below. This method allows to analyze a broad class of evolutionary algorithms, namely all that give rise to the following *population selection-mutation (PSM) process* (identical to the one defined in [24] even though we use a slightly more algorithmic language). Let  $\Omega$  be a finite set. We call  $\Omega$  the *search space* and its elements *solution candidates* or *individuals*. Let  $\lambda \in \mathbb{N}$  be called the *population size* of the process. An ordered multi-set of cardinality  $\lambda$ , in other words, a  $\lambda$ -tuple, over the search space  $\Omega$  is called a *population*. Let  $\mathcal{P} = \Omega^\lambda$  be the set of all populations. For  $P \in \mathcal{P}$ , we write  $P_1, \dots, P_\lambda$  to denote the elements of  $P$ . We also write  $x \in P$  to denote that there is an  $i \in [1..\lambda]$  such that  $x = P_i$ .

A PSM process starts with some, possibly random, population  $P^{(0)}$ . In each iteration  $t = 1, 2, \dots$ , a new population  $P^{(t)}$  is sampled from the previous one  $P^{(t-1)}$  as follows. Via a (possibly) randomized *selection operator*  $\text{sel}(\cdot)$ , a  $\lambda$ -tuple of individuals is selected and then each of them creates an offspring through the application of a randomized *mutation operator*  $\text{mut}(\cdot)$ .

The *selection operator* can be arbitrary except that it only selects individuals from  $P^{(t-1)}$ . In particular, we do not assume that the selected individuals are independent. Formally speaking, the outcome of the selection process is a random  $\lambda$ -tuple  $Q = \text{sel}(P) \in [1..\lambda]^\lambda$  such that  $P_{Q_1}^{(t-1)}, \dots, P_{Q_\lambda}^{(t-1)}$  are the selected parents.

From each selected parent  $P_{Q_i}^{(t-1)}$ , a single offspring  $P_i^{(t)}$  is generated via a randomized *mutation operator*  $P_i^{(t)} = \text{mut}(P_{Q_i}^{(t-1)})$ . Formally speaking, for each  $x \in \Omega$ ,  $\text{mut}(x)$  is a probability distribution on  $\Omega$  and we write  $y = \text{mut}(x)$  to indicate that  $y$  is sampled from this distribution. We assume that each sample, that is, each call of a mutation operator, uses independent randomness. With this notation, we can write the new population as  $P^{(t)} = (\text{mut}(P_{\text{sel}(P)_1}^{(t-1)}), \dots, \text{mut}(P_{\text{sel}(P)_\lambda}^{(t-1)}))$ . From the definition it is clear that a PSM process is a Markov process with state space  $\mathcal{P}$ .

The following characteristic of the selection operator was found to be crucial for the analysis of PSM processes in [24]. Let  $P \in \mathcal{P}$  and  $i \in [1..\lambda]$ . Then the random variable  $R(i, P) = |\{j \in [1..\lambda] \mid \text{sel}(P)_j = P_i\}|$ , called *reproduction number of the  $i$ -th individual in  $P$* , denotes the number of times  $P_i$  was selected from  $P$  as parent. Its expectation  $E[R(i, P)]$  is called *reproduction rate*.

Our version of the negative drift in populations method now is the following.

**Theorem 2.** *Consider a PSM process  $(P^{(t)})_{t \geq 0}$  as described above. Let  $g : \Omega \rightarrow \mathbb{Z}_{\geq 0}$ , called potential function, and  $a, b \in \mathbb{Z}_{\geq 0}$  with  $a \leq b$ . Assume that for all  $x \in P^{(0)}$  we have  $g(x) \geq b$ . Let  $T = \min\{t \geq 0 \mid \exists i \in [1.. \lambda] : g(P_i^{(t)}) \leq a\}$  the first time we have a search point with potential  $a$  or less in the population. Assume that the following three conditions are satisfied.*

- (i) *There is an  $\alpha \geq 1$  such that for all populations  $P \in \mathcal{P}$  with  $\min\{g(P_i) \mid i \in [1.. \lambda]\} > a$  and all  $i \in [1.. \lambda]$  with  $g(P_i) < b$ , we have  $E[R(i, P)] \leq \alpha$ .*
- (ii) *There is a  $\kappa > 0$  and a  $0 < \delta < 1$  such that for all  $x \in \Omega$  with  $a < g(x) < b$  we have*

$$E[\exp(-\kappa g(\text{mut}(x)))] \leq \frac{1}{\alpha}(1 - \delta) \exp(-\kappa g(x)).$$

- (iii) *There is a  $D \geq \delta$  such for all  $x \in \Omega$  with  $g(x) \geq b$ , we have*

$$E[\exp(-\kappa g(\text{mut}(x)))] \leq D \exp(-\kappa b).$$

Then

- $E[T] \geq \frac{\delta}{2D\lambda} \exp(\kappa(b - a)) - \frac{1}{2}$ , and
- for all  $L \geq 1$ , we have  $\Pr[T < L] \leq L\lambda \frac{D}{\delta} \exp(-\kappa(b - a))$ .

Before proceeding with the proof, we compare our result with Theorem 1 of [24]. We first note that, apart from a technicality which we discuss toward the end of this comparison, the assumptions of our result are weaker than the ones on [24] since we do not need the technical fourth and fifth assumption of [24], which in our notation would read as follows.

- There is a  $\delta_2 > 0$  such that for all  $i \in [a..b]$  and all  $k, \ell \in \mathbb{Z}$  with  $1 \leq k + \ell$  and all  $x, y \in \Omega$  with  $g(x) = i$  and  $g(y) = i - \ell$  we have

$$\begin{aligned} \Pr[g(\text{mut}(x)) = i - \ell \wedge g(\text{mut}(y)) = i - \ell - k] \\ \leq \exp(\kappa(1 - \delta_2)(b - a)) \Pr[g(\text{mut}(x)) = i - k - \ell]. \end{aligned}$$

- There is a  $\delta_3 > 0$  such that for all  $i, j, k, \ell \in \mathbb{Z}$  with  $a \leq i \leq b$  and  $1 \leq k + \ell \leq j$  and all  $x, y \in \Omega$  with  $g(x) = i$  and  $g(y) = i - k$  we have

$$\Pr[g(\text{mut}(x)) = i - j] \leq \delta_3 \Pr[g(\text{mut}(y)) = i - k - \ell].$$

The assertion of our result is of the same type as in [24], but stronger in terms of numbers. For the probability  $\Pr[T < L]$  to find a potential of at most  $a$  in time less than  $L$ , a bound of

$$O(\lambda L^2 D (b - a) \exp(-\kappa \delta_2 (b - a)))$$

is shown in [24]. Hence our result is smaller by a factor of  $\Omega(L(b - a) \exp(-\kappa(1 - \delta_2)(b - a)))$ . In addition, our result is non-asymptotic, that is, the lower bound contains no asymptotic notation or unspecified constants.

The one point where Lehre’s [24] result potentially is stronger is that it needs assumptions only on the average drift, whereas we require the same assertion on the point-wise drift. More concretely, Lehre uses the notation  $(X_t)_{t \geq 0}$  to denote the Markov process on  $\Omega$  associated with the mutation operator (it is not said in [24] what is  $X_0$ , that is, how this process is started). Then  $\Delta_t(i) = (g(X_{t+1}) - g(X_t) \mid g(X_t) = i)$  defines the potential gain in step  $t$  when the current state has potential  $i$ . With this notation, instead of our second and third condition, Lehre [24] requires only the weaker conditions (here again translated into our notation).

- (ii’) For all  $t \geq 0$  and all  $a < i < b$ ,  $E[\exp(-\kappa \Delta_t(i))] < \frac{1}{\alpha}(1 - \delta)$ .
- (iii’) For all  $t \geq 0$ ,  $E[\exp(-\kappa(g(X_{t+1}) - b)) \mid g(X_t) \geq b] < D$ .

So Lehre only requires that the random individual at time  $t$ , conditional on having a certain potential, gives rise to a certain drift, whereas we require that each particular individual with this potential gives rise to this drift. On the formal level, Lehre’s condition is much weaker than ours (assuming that the unclear point of what is  $X_0$  can be fixed). That said, to exploit such weaker conditions, one would need to be able to compute such average drifts and they would need to be smaller than the worst-case point-wise drift. We are not aware of many examples where average drift was successfully used in drift analysis (one is Jägersküpfer’s remarkable analysis of the linear functions problem [20]) despite the fact that many classic drift theorems only require conditions on the average drift to hold.

We now prove Theorem 2. Before stating the formal proof, we describe on a high level its main ingredients and how it differs from Lehre’s proof.

The main challenge when using drift analysis is designing a potential function that suitably measures the progress. For simple hillclimbers and optimization problems, the fitness of the current solution may suffice, but already the analysis of the  $(1 + 1)$  EA on linear functions resisted such easy approaches [13, 16, 19, 38]. For population-based algorithms, the additional challenge is to capture the quality of the whole population in a single number. We note at this point that the notion of “negative drift in populations” was used in Lehre to informally describe the characteristic of the population processes regarded, but drift analysis as a mathematical tool was employed only on the level of single individuals and the resulting findings were lifted to the whole population via advanced tools like branching processes and eigenvalue arguments.

To prove upper bounds, in [1, 3–5, 14, 25, 37], implicitly or explicitly potential functions were used that build on the fitness of the best individual in the population and the number of individuals having this fitness. Regarding only the current-best individuals, these potential functions might not be suitable for lower bound proofs.

The lower bound proofs in [2, 29, 33, 34] all define a natural potential for single individuals, namely the Hamming distance to the optimum, and then lift this potential to populations by summing over all individuals an exponential transformation of their base potential (this ingenious definition was, to the best of our knowledge, not known in the theory of evolutionary algorithms before

the work of Neumann, Oliveto, and Witt [29]). This is the type of potential we shall use as well, and given the assumptions of Theorem 2, it is not surprising that  $\sum_{x \in P} \exp(-\kappa g(x))$  is a good choice. For this potential, we shall then show with only mild effort that it satisfies the assumptions of our drift theorem, which yields the desired lower bounds on the runtime (using that a single good solution in the population already requires a very high potential due to the exponential scaling). We now give the details of this proof idea.

*Proof (of Theorem 2).* We consider the process  $(X_t)$  defined by  $X_t = \sum_{i=1}^\lambda \exp(-\kappa g(P_i^{(t)}))$ . To apply drift arguments, we first analyze the expected state after one iteration, that is,  $E[X_t \mid X_{t-1}]$ . To this end, let us consider a fixed parent population  $P = P^{(t-1)}$  in iteration  $t$ . Let  $Q = \text{sel}(P)$  be the indices of the individuals selected for generating offspring.

We first condition on  $Q$  (and as always on  $P$ ), that is, we regard only the probability space defined via the mutation operator, and compute

$$\begin{aligned} E[X_t \mid Q] &= E \left[ \sum_{j=1}^\lambda \exp(-\kappa g(\text{mut}(P_{Q_j})) \right) \\ &= \sum_{i=1}^\lambda (R(i, P) \mid Q) E[\exp(-\kappa g(\text{mut}(P_i)))]. \end{aligned}$$

Using that  $\sum_{i=1}^\lambda R(i, P) = \lambda$  and not anymore conditioning on  $Q$ , by the law of total expectation, we have

$$\begin{aligned} E[X_t] &= E_Q[E[X_t \mid Q]] \\ &= \sum_{i=1}^\lambda E[R(i, P)] E[\exp(-\kappa g(\text{mut}(P_i)))] \\ &= \sum_{P_i: g(P_i) < b} \alpha E[\exp(-\kappa g(\text{mut}(P_i)))] + \sum_{P_i: g(P_i) \geq b} E[R(i, P)] D \exp(-\kappa b) \\ &\leq \sum_{P_i: g(P_i) < b} \alpha \cdot \frac{1}{\alpha} (1 - \delta) \exp(-\kappa g(P_i)) + \lambda \cdot D \exp(-\kappa b) \\ &\leq (1 - \delta) X_{t-1} + \lambda D \exp(-\kappa b) \end{aligned}$$

and recall that this is conditional on  $P^{(t-1)}$ , hence also on  $X_{t-1}$ .

Let  $\Delta = \lambda D \exp(-\kappa b)$ . Since  $P^{(0)}$  contains no individual with potential below  $b$ , we have  $X_0 \leq \lambda \exp(-\kappa b) = \frac{\Delta}{D} \leq \frac{\Delta}{\delta}$ . Hence also the assumption  $E[X_0] \leq \frac{\Delta}{\delta}$  of Lemma 1 is fulfilled.

Let  $M = \exp(-\kappa a)$  and  $T' := \min\{t \geq 0 \mid X_t \geq M\}$ . Note that  $T$ , the first time to have an individual with potential at least  $a$  in the population, is at least  $T'$ . Now the negative multiplicative drift theorem (Lemma 1) gives  $\Pr[T < L] \leq \Pr[T' < L] \leq \frac{L\Delta}{M\delta} = L\lambda D \frac{\exp(-\kappa(b-a))}{\delta}$  and  $E[T] \geq E[T'] \geq \frac{\delta M}{2\Delta} - \frac{1}{2} = \frac{\delta}{2D\lambda} \exp(\kappa(b-a)) - \frac{1}{2}$ .  $\square$

We note that the proof above actually shows the following slightly stronger statement, which might be useful when working with random initial populations.

**Theorem 3.** *Theorem 2 remains valid when the assumption that all initial individuals have potential at least  $b$  is replaced by the assumption  $\sum_{i=1}^{\lambda} E[\exp(-\kappa g(P_i^{(0)}))] \leq \frac{\lambda D \exp(-\kappa b)}{\delta}$ .*

### 4 Processes Using Standard Bit Mutation

Since many EAs use standard bit mutation, as in [24] we now simplify our main result for processes using standard bit mutation and for  $g$  being the Hamming distance to a target solution. Hence in this section, we have  $\Omega = \{0, 1\}^n$  and  $y = \text{mut}(x)$  is obtained from  $x$  by flipping each bit of  $x$  independently with probability  $p$ . Since our results are non-asymptotic, we can work with any  $p \leq \frac{1}{2}$ .

**Theorem 4.** *Consider a PSM process with search space  $\Omega = \{0, 1\}^n$ , using standard bit mutation with mutation rate  $p \in [0, \frac{1}{2}]$  as mutation operator, and such that  $P_i^{(0)}$  is uniformly distributed in  $\Omega$  for each  $i \in [1.. \lambda]$ . Let  $x^* \in \Omega$  be the target of the process. For all  $x \in \Omega$ , let  $g(x) := H(x, x^*)$  denote the Hamming distance from the target.*

*Let  $\alpha > 1$  and  $0 < \delta < 1$  such that  $\ln(\frac{\alpha}{1-\delta}) < pn$ , that is, such that  $1 - \frac{1}{pn} \ln(\frac{\alpha}{1-\delta}) =: \varepsilon > 0$ . Let  $B = \frac{2}{\varepsilon}$ . Let  $a, b$  be integers such that  $0 \leq a < b$  and  $b \leq \tilde{b} := n \frac{1}{B^2 - 1}$ .*

*Selection condition: Assume that for all populations  $P \in \mathcal{P}$  with  $\min\{g(P_i) \mid i \in [1.. \lambda]\} > a$  and all  $i \in [1.. \lambda]$  with  $g(P_i) < b$ , we have  $E[R(i, P)] \leq \alpha$ .*

*Then the first time  $T := \min\{t \geq 0 \mid \exists i \in [1.. \lambda] : g(P_i^{(t)}) \leq a\}$  that the population contains an individual in distance  $a$  or less from  $x^*$  satisfies*

$$E[T] \geq \frac{1}{2\lambda} \min \left\{ \frac{\delta\alpha}{(1-\delta)}, 1 \right\} \exp \left( \ln \left( \frac{2}{1 - \frac{1}{pn} \ln(\frac{\alpha}{1-\delta})} \right) (b-a) \right) - \frac{1}{2},$$

$$\Pr[T < L] \leq L\lambda \max \left\{ \frac{(1-\delta)}{\delta\alpha}, 1 \right\} \exp \left( -\ln \left( \frac{2}{1 - \frac{1}{pn} \ln(\frac{\alpha}{1-\delta})} \right) (b-a) \right).$$

We have to defer the elementary proof, a reduction to Theorem 2, to the extended version [10] for reasons of space. To show that the second and third condition of Theorem 2 are satisfied, one has to estimate  $E[\exp(-\kappa(g(\text{mut}(x)) - g(x)))]$ , which is not difficult since  $g(\text{mut}(x)) - g(x)$  can be written as sum of independent random variables. With a similar computation, we show that the weaker starting condition of Theorem 3 is satisfied.

As a simple **example** for an application of this result, let us consider the classic  $(\mu, \lambda)$  EA (with uniform selection for variation, truncation selection for inclusion into the next generation, and mutation rate  $p = \frac{1}{n}$ ) with  $\lambda = 2\mu$  optimizing some function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ ,  $n = 500$ , with unique global optimum. For simplicity, let us take as performance measure  $\lambda T$ , that is, the number of

fitness evaluations in all iterations up to the one in which the optimum was found. Since  $\lambda = 2\mu$ , we have  $\alpha = 2$ . By taking  $\delta = 0.01$ , we obtain a concrete lower bound of more than 13 million fitness evaluations until the optimum is found (regardless of  $\mu$  and  $f$ ).

Since the result above is slightly technical, we now formulate the following corollary, which removes the variable  $\delta$  without significantly weakening the result. We note that the proof of this result applies Theorem 4 with a non-constant  $\delta$ , so we do not see how such a result could have been proven from Lehre’s result [24].

**Corollary 5.** *Consider a PSM process as in Theorem 4. Let  $x^* \in \Omega$  be the target of the process. For all  $x \in \Omega$ , let  $g(x) := H(x, x^*)$  denote the Hamming distance from the target. Assume that there is an  $\alpha > 1$  such that*

- $\ln(\alpha) \leq p(n - 1)$ , which is equivalent to  $\gamma := 1 - \frac{\ln \alpha}{pn} \geq \frac{1}{n}$ ;
- there is an  $a \leq b := \lfloor (1 - \frac{a}{n})n^{\frac{1}{\gamma^2} - 1} \rfloor$  such that for all populations  $P \in \mathcal{P}$  with  $\min\{g(P_i) \mid i \in [1.. \lambda]\} > a$  and for all  $i \in [1.. \lambda]$ , we have  $E[R(i, P)] \leq \alpha$ .

Then the first time  $T := \min\{t \geq 0 \mid \exists i \in [1.. \lambda] : g(P_i^{(t)}) \leq a\}$  that the population contains an individual in distance  $a$  or less from  $x^*$  satisfies

$$E[T] \geq \frac{p\alpha}{4\lambda n} \min\left\{1, \frac{2n}{p\alpha}\right\} \exp\left(\ln\left(\frac{2}{\gamma}\right)(b - a)\right) - \frac{1}{2},$$

$$\Pr[T < L] \leq \frac{2L\lambda n}{p\alpha} \max\left\{1, \frac{p\alpha}{2n}\right\} \exp\left(-\ln\left(\frac{2}{\gamma}\right)(b - a)\right).$$

In particular, if  $a \leq (1 - \varepsilon)b$  for some constant  $\varepsilon > 0$ , then  $T$  is super-polynomial in  $n$  (in expectation and with high probability) when  $\gamma = \omega(n^{-1/2})$  and exponential when  $\gamma = \Omega(1)$ .

We omit the proof for reasons of space. It can be found in [10]. The main argument is employing Theorem 4 with the  $\delta = \frac{p}{2n}$  and computing that this small  $\delta$  has no significant influence on the exponential term of the bounds.

## 5 Fitness Proportionate Selection

In this section, we apply our method to a mutation-only version of the simple genetic algorithm (simple GA). This algorithm starts with a population of  $\mu$  random bit strings of length  $n$ . In each iteration, it computes a new population by  $\mu$  times independently selecting an individual from the existing population via fitness proportionate selection and mutating it via standard bit mutation with mutation rate  $p = \frac{1}{n}$ .

The first work [29, Theorem 8] analyzing this algorithm showed that with  $\mu \leq \text{poly}(n)$  it needs with high probability more than  $2^{n^{1 - O(1/\log \log n)}}$  iterations to find the optimum of the ONEMAX function or any search point in Hamming distance at most  $0.003n$  from it. Hence this is only a subexponential lower bound. In [25,

Corollary 13], building on the lower bound method from [24], a truly exponential lower bound is shown for the task of finding a search point in Hamming distance at most  $0.029n$  from the optimum, but only for a relatively large population size of  $\mu \geq n^3$  (and again  $\mu \leq \text{poly}(n)$ ).

We now extend this result to arbitrary  $\mu$ , that is, we remove the conditions  $\mu \geq n^3$  and  $\mu \leq \text{poly}(n)$ . To obtain the constant 0.029, we have to compromise with the constants in the runtime, which consequently are only of a theoretical interest. We therefore do not specify the base of the exponential function or the leading constant. We note that this would have been easily possible since we only use a simple additive Chernoff bound and Corollary 5. We further note that Lehre [25] also shows lower bounds for a scaled version of fitness proportionate selection and a general  $\Theta(1/n)$  mutation rate. This would also be possible with our approach and would again remove the conditions on  $\lambda$ , but we do not see that the additional effort is justified here.

**Theorem 6.** *There is a  $T = \exp(\Omega(n))$  such that the mutation-only simple GA optimizing ONEMAX with any population size  $\mu$  with probability  $1 - \exp(-\Omega(n))$  does not find any solution  $x$  with  $\text{ONEMAX}(x) \geq 0.971n$  within  $T$  fitness evaluations.*

The main difficulty for proving lower bounds for algorithms using fitness proportionate selection (and maybe the reason why [24] does not show such bounds) is that the reproduction number is non-trivial to estimate. If all but one individual have a fitness of zero, then this individual is selected  $\mu$  times. Hence  $\mu$  is the only general upper bound for the reproduction number. The previous works and ours overcome this difficulty by arguing that the average fitness in the population cannot significantly drop below the initial value of  $n/2$ , which immediately yields that an individual with fitness  $k$  has a reproduction number of roughly at most  $\frac{k}{n/2}$ .

While it is natural that the typical fitness of an individual should not drop far below  $n/2$ , informally arguing that the individuals should be at least as good as random individuals, making this argument precise is not completely trivial. In [29, Lemma 6], it is informally argued that the situation with fitness proportionate selection cannot be worse than with uniform selection and for the latter situation a union bound over all lineages of individuals is employed and a negative-drift analysis from [30, Section 3] is used for a single lineage. The analysis in [25, Lemma 9] builds on the (positive) drift stemming from standard bit mutation when the fitness is below  $n/2$  (this argument needs a mutation rate of at least  $\Omega(1/n)$ ) and the independence of the offspring (here the lower bound  $\lambda \geq n^3$  is needed to allow the desired Chernoff bound estimates).

Our proof relies on a natural domination argument that shows that at all times all individuals are at least as good as random individuals in the sense of stochastic domination (see, e.g., [7]) of their fitness. This allows to use a simple Chernoff bound to argue that with high probability, for a long time all individuals have a fitness of at least  $(\frac{1}{2} - \varepsilon)n$ . The remainder of the proof is an application of Corollary 5. Clearly, Lehre's lower bound [24, Theorem 4] would

have been applicable as well with the main difference being that one has to deal with the constant  $\delta$ , which does not exist in Corollary 5. The full proof can again be found in [10].

## 6 Conclusion and Outlook

In this work, we have proven two technical tools which might ease future lower bound proofs in discrete evolutionary optimization. The negative multiplicative drift theorem has the potential to replace the more technical negative drift theorems used so far in different contexts. Our strengthening and simplification of the negative drift in populations method should help increasing our not very developed understanding of population-based algorithms in the future. Clearly, it is restricted to mutation-based algorithms – providing such a tool for crossover-based algorithms and extending our understanding how to prove lower bounds for these beyond the few results [11, 15, 34, 36] would be a great progress.

## References

1. Antipov, D., Doerr, B., Fang, J., Hetet, T.: Runtime analysis for the  $(\mu + \lambda)$  EA optimizing OneMax. In: Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 1459–1466. ACM (2018)
2. Antipov, D., Doerr, B., Yang, Q.: The efficiency threshold for the offspring population size of the  $(\mu, \lambda)$  EA. In: Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 1461–1469. ACM (2019)
3. Chen, T., He, J., Sun, G., Chen, G., Yao, X.: A new approach for analyzing average time complexity of population-based evolutionary algorithms on unimodal problems. *IEEE Trans. Syst. Man Cybern. Part B* **39**, 1092–1106 (2009)
4. Corus, D., Dang, D., Eremeev, A.V., Lehre, P.K.: Level-based analysis of genetic algorithms and other search processes. *IEEE Trans. Evol. Comput.* **22**, 707–719 (2018)
5. Dang, D., Lehre, P.K.: Runtime analysis of non-elitist populations: from classical optimisation to partial information. *Algorithmica* **75**, 428–461 (2016)
6. 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)
7. Doerr, B.: Analyzing randomized search heuristics via stochastic domination. *Theoret. Comput. Sci.* **773**, 115–137 (2019)
8. Doerr, B.: An exponential lower bound for the runtime of the compact genetic algorithm on jump functions. In: Foundations of Genetic Algorithms, FOGA 2019, pp. 25–33. ACM (2019)
9. Doerr, B.: Does comma selection help to cope with local optima? In: Genetic and Evolutionary Computation Conference, GECCO 2020. ACM (2020, to appear)
10. Doerr, B.: Lower bounds for non-elitist evolutionary algorithms via negative multiplicative drift. *CoRR abs/2004.01274* (2020)
11. Doerr, B.: Runtime analysis of evolutionary algorithms via symmetry arguments. *CoRR abs/2006.04663* (2020)



12. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. *Algorithmica* **65**, 224–250 (2013)
13. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. *Algorithmica* **64**, 673–697 (2012)
14. Doerr, B., Kötzing, T.: Multiplicative up-drift. In: Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 1470–1478. ACM (2019)
15. Doerr, B., Theile, M.: Improved analysis methods for crossover-based algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2009, pp. 247–254. ACM (2009)
16. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. *Theoret. Comput. Sci.* **276**, 51–81 (2002)
17. Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. *Adv. Appl. Probab.* **13**, 502–525 (1982)
18. Happ, E., Johannsen, D., Klein, C., Neumann, F.: Rigorous analyses of fitness-proportional selection for optimizing linear functions. In: Genetic and Evolutionary Computation Conference, GECCO 2008, pp. 953–960. ACM (2008)
19. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. *Artif. Intell.* **127**, 51–81 (2001)
20. Jägersküpper, J.: A blend of Markov-chain and drift analysis. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 41–51. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-87700-4\\_5](https://doi.org/10.1007/978-3-540-87700-4_5)
21. Jägersküpper, J., Storch, T.: When the plus strategy outperforms the comma strategy and when not. In: Foundations of Computational Intelligence, FOCI 2007, pp. 25–32. IEEE (2007)
22. Johannsen, D.: Random combinatorial structures and randomized search heuristics. Ph.D. thesis, Universität des Saarlandes (2010)
23. Kötzing, T.: Concentration of first hitting times under additive drift. *Algorithmica* **75**, 490–506 (2016)
24. Lehre, P.K.: Negative drift in populations. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 244–253. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_25](https://doi.org/10.1007/978-3-642-15844-5_25)
25. Lehre, P.K.: Fitness-levels for non-elitist populations. In: Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 2075–2082. ACM (2011)
26. Lengler, J.: Drift analysis. In: Doerr, B., Neumann, F. (eds.) *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pp. 89–131. Springer, Cham (2020). <https://arxiv.org/abs/1712.00964>
27. Lengler, J., Steger, A.: Drift analysis and evolutionary algorithms revisited. *Comb. Probab. Comput.* **27**, 643–666 (2018)
28. Mitavskiy, B., Rowe, J.E., Cannings, C.: Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. *Int. J. Intell. Comput. Cybern.* **2**, 243–284 (2009)
29. Neumann, F., Oliveto, P.S., Witt, C.: Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In: Genetic and Evolutionary Computation Conference, GECCO 2009, pp. 835–842. ACM (2009)
30. Oliveto, P.S., Witt, C.: Simplified drift analysis for proving lower bounds in evolutionary computation. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 82–91. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-87700-4\\_9](https://doi.org/10.1007/978-3-540-87700-4_9)
31. Oliveto, P.S., Witt, C.: Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica* **59**, 369–386 (2011)

32. Oliveto, P.S., Witt, C.: Erratum: simplified drift analysis for proving lower bounds in evolutionary computation. CoRR abs/1211.7184 (2012)
33. Oliveto, P.S., Witt, C.: On the runtime analysis of the simple genetic algorithm. Theoret. Comput. Sci. **545**, 2–19 (2014)
34. Oliveto, P.S., Witt, C.: Improved time complexity analysis of the simple genetic algorithm. Theoret. Comput. Sci. **605**, 21–41 (2015)
35. Rowe, J.E., Sudholt, D.: The choice of the offspring population size in the  $(1, \lambda)$  evolutionary algorithm. Theoret. Comput. Sci. **545**, 20–38 (2014)
36. Sutton, A.M., Witt, C.: Lower bounds on the runtime of crossover-based algorithms via decoupling and family graphs. In: Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 1515–1522. ACM (2019)
37. Witt, C.: Runtime analysis of the  $(\mu + 1)$  EA on simple pseudo-Boolean functions. Evol. Comput. **14**, 65–86 (2006)
38. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Comb. Probab. Comput. **22**, 294–318 (2013)
39. Witt, C.: Upper bounds on the running time of the univariate marginal distribution algorithm on OneMax. Algorithmica **81**, 632–667 (2019)



# Exponential Upper Bounds for the Runtime of Randomized Search Heuristics

Benjamin Doerr<sup>(✉)</sup>

Laboratoire d'Informatique (LIX), CNRS, École Polytechnique,  
Institut Polytechnique de Paris, Palaiseau, France  
`doerr@lix.polytechnique.fr`

**Abstract.** We argue that proven exponential upper bounds on runtimes, an established area in classic algorithms, are interesting also in evolutionary computation and we prove several such results. We show that any of the algorithms randomized local search, Metropolis algorithm, simulated annealing, and  $(1+1)$  evolutionary algorithm can optimize any pseudo-Boolean weakly monotonic function under a large set of noise assumptions in a runtime that is at most exponential in the problem dimension  $n$ . This drastically extends a previous such result, limited to the  $(1+1)$  EA, the LeadingOnes function, and one-bit or bit-wise prior noise with noise probability at most  $1/2$ , and at the same time simplifies its proof. With the same general argument, among others, we also derive a sub-exponential upper bound for the runtime of the  $(1, \lambda)$  evolutionary algorithm on the OneMax problem when the offspring population size  $\lambda$  is logarithmic, but below the efficiency threshold.

**Keywords:** Runtime analysis · Noisy optimization · Theory

## 1 Introduction

The mathematical analysis of runtimes of randomized search heuristics is an established field of the general area of heuristic search [3, 15, 29, 38]. The vast majority of the results in this area show that a certain algorithm can solve (or approximately solve) a certain problem within some polynomial runtime (polynomial upper bound on the runtime) or show that this is not possible by giving a super-polynomial, often exponential, lower bound on the runtime.

As a rare exception to this rule, in his extensive analysis of how the  $(1+1)$  evolutionary algorithm ( $(1+1)$  EA)<sup>1</sup> optimizes the LEADINGONES benchmark in the presence of prior noise, Sudholt [45, Theorem 6] showed that for one-bit or bit-wise noise with noise probability at most  $\frac{1}{2}$ , the  $(1+1)$  EA finds the optimum of LEADINGONES in time at most  $2^{O(n)}$ . While clearly a very natural result – everyone would agree that also with such noise the unimodal LEADINGONES

<sup>1</sup> See Section 2 for details on all technical terms used in this introduction.

For reasons of space, some technical details have been omitted from this extended abstract. The interested reader can find them in the extended version [10].

problem should not become harder than the needle-in-the-haystack problem – the technical, long, and problem-specific proof of this result, despite following the intuitive argument just laid out, suggests that such analyses can be harder than one would expect.

In this work, we will argue that such exponential upper bounds are interesting beyond completing a runtime picture of a given problem. We then show that with a different analysis method such uncommon runtime questions can be analyzed relatively easily. As one out of several results, we drastically extend the result in [45] and show that an exponential runtime guarantee holds for

- any of the algorithms randomized local search, Metropolis algorithm, simulated annealing, and  $(1 + 1)$  EA,
- when optimizing any weakly monotonic objective function, e.g., ONEMAX, linear functions, monotone polynomials, LEADINGONES, plateau functions, and the needle problem,
- in the presence of all common forms of prior and posterior noise with a noise probability of at most  $1 - \varepsilon$ ,  $\varepsilon > 0$  a constant.

### 1.1 Exponential Runtime Analysis

The area of mathematical runtime analysis, established as a recognized sub-field of the theory of evolutionary algorithms by Ingo Wegener and his research group around twenty years ago, seeks to understand the working principles of evolutionary computation via rigorously proven results on the performance of evolutionary algorithms and other search heuristics in a similar spirit as done in classic algorithms analysis for much longer time.

Adopting the view of classic algorithmics that runtimes polynomial in the problems size are efficient and larger runtimes are inefficient, the vast majority of the results in this field prove polynomial upper bounds or super-polynomial lower bounds. For two reasons, we feel that also super-polynomial and even exponential runtime guarantees are desirable in the theory of evolutionary algorithms.

Our first set of arguments is identical to the arguments made in the classic algorithms field, which led to a shift in paradigms and established the field of exact exponential algorithms [20,21]. These arguments are that (i) for many important problems nothing better than exponential time algorithms are known, so one cannot just ignore these problems in algorithms research, (ii) with the increase of computational power, also exponential time algorithms can be used for problems of moderate (and interesting) size, and (iii) that the existing research on exponential-time algorithms has produced many algorithms that, while still exponential time, are much faster than classic exponential-time approaches like exhaustive search.

Our second line of argument is that exponential time algorithms are of additional interest in evolutionary computation for the following reasons.

(i) *To increase our understanding of the working principles of evolutionary algorithms.* There is a large number of exponential lower bounds in our field, but for essentially none of them an upper bound better than the trivial  $n^{O(n)}$

bound exists. It is clear that matching upper and lower bounds tell us most, not only about the runtimes, but also about the working principles of EAs. Tight bounds naturally have to grasp the true way the EA progresses better than loose bounds. For example, the general  $n^{O(n)}$  upper bound for all algorithms using standard bit mutation is based on the simple argument that the optimum can be generated from any search point with probability at least  $n^{-n}$ . Besides being very pessimistic, this argument does not tell us a lot on how really the EA optimizes the problem at hand (except for the very particular case that the EA is stuck in a local optimum in Hamming distance  $n$  to the global optimum). In contrast, as a positive example, the matching  $(1 \pm o(1))en \ln n$  upper [35] and lower [24] bound for the runtime of the  $(1 + 1)$  EA on ONEMAX together with their proofs shows that for this optimization process, the effect of mutations flipping more than one bit has no influence on the runtime apart from lower order terms. In a broader sense, this insight suggests that flipping larger number of bits is mainly useful to leave local optima, but not to make fast progress along easy slopes of the fitness landscape.

(ii) *Because understanding runtimes in the exponential and super-exponential regime is important for the application of EAs.* Many classic evolutionary algorithms can easily have a super-exponential runtime. For example, Witt [48] has shown that the simple  $(1 + 1)$  EA has an expected runtime<sup>2</sup> of  $n^{\Theta(n)}$  on the minimum makespan scheduling problem. Hence knowing that an evolutionary algorithm “only” has an exponential runtime can be interesting.

We note that for problems with exponential-size search spaces (such as the search space  $\{0, 1\}^n$  regarded exclusively in this work) blind random search and exhaustive search are exponential-time alternatives. For that reason, in addition to knowing that an EA has an exponential runtime guarantee (that is, a runtime of at most  $C^n$  for some constant  $C > 1$ ), it would be very desirable to also have a good estimate for the base of the exponential function, that is, the constant  $C$ . Unfortunately, at this moment where we just start reducing the trivial  $n^{O(n)}$  upper bound to exponential upper bounds, we are not yet in the position to optimize the constants in the exponent. We are optimistic though (and give some indication for this in Sect. 6) that our methods can be fine-tuned to give interesting values for the base of the exponential function as well. We recall that such an incremental progress is not untypical for the mathematical runtime analysis of EAs – in the regime of polynomial bounds, subject to intensive research since the 1990s, the leading constants for elementary problems such as LEADINGONES and linear functions were only determined from 2010 on [6, 44, 49].

With this motivation in mind and spurred by the observation that exponential upper bounds are not trivial to obtain, we start in this work a first general attack on the problem of proving exponential upper bounds.

---

<sup>2</sup> As common both in classic algorithms and in our field, by runtime we mean the worst-case runtime taken over all input instances.

## 1.2 State of the Art

We are not aware of many previous works on exponential or super-exponential upper bounds on runtimes of EAs. In the maybe first work proving an exponential upper bound, Droste, Jansen, and Wegener [18, Theorem 9] show that the  $(1 + 1)$  EA optimizes the NEEDLE function (called peak function there) in expected time at most  $(2\pi)^{-1}n^{1/2} \exp(2n)$ . Only a year later, Garnier, Kallel, and Schoenauer [24, Proposition 3.1] in a remarkably precise analysis showed that the expected runtime of the  $(1 + 1)$  EA on the NEEDLE function is  $(1 \pm o(1))(1 - \frac{1}{e})^{-1}2^n$ .

A general upper bound of  $n^n$  for the expected runtime of the  $(1+1)$  EA on any pseudo-Boolean functions was given in [19, Theorem 6]. Analogous arguments showed an upper bound of  $4^n \log_2 n$  for the  $(1 + 1)$  EA using the  $2^i/n$  mutation rates in a cyclic fashion [30, Theorem 3] and an upper bound of  $O(n^\beta 2^n)$  for the fast  $(1 + 1)$  EA with (constant) power-law exponent  $\beta > 1$  [14, Theorem 5.3].

The general  $n^{O(n)}$  upper bound of [19] is tight as witnessed, among others, by the trap function [19, Theorem 8] and the minimum makespan scheduling problem [48]. There are a few analyses for parameterized problems showing bounds that can become exponential or worse when the problem parameter is chosen in an extreme manner. Here the  $\Theta(n^k)$  runtime bound for the  $(1+1)$  EA optimizing jump functions with jump size  $k \geq 2$  [19, Theorem 25] is the best known example. More interesting results have been derived in the context of parameterized complexity [37], but again these results have been derived with small parameter values in mind and thus are most interesting for this case.

In contrast to these sporadic upper bounds, there is a large number of exponential lower bounds, e.g., for a broad class of non-elitist algorithms with too low selection pressure [32], for some algorithms using fitness-proportionate selection [26], for the simple genetic algorithm with an only moderately large population size [39], and for various problems in noisy optimization [25, 41, 45].

Apart from a single exception, for none of these lower bounds it is known whether the runtime is really exponential or is higher, say  $n^{\Theta(n)}$ . The exceptional exponential upper bound shown in [45, Theorem 6] reads as follows. Consider optimizing the LEADINGONES benchmark function defined on bit strings of length  $n$  via the  $(1 + 1)$  EA. Assume that in each iteration, the fitness evaluation of both parent and offspring is subject to stochastically independent prior noise of one of the following two types. (i) With probability  $p \leq \frac{1}{2}$ , not the true fitness is returned, but the fitness of a random Hamming neighbor. (ii) With probability  $p' \in [0, 1]$ , the search point to be evaluated is disturbed by flipping each bit independently with some probability  $q \leq \frac{1}{2}$  and the fitness of this disturbed search point is returned, with probability  $1 - p'$ , the fitness of the original search point is returned; here we assume that  $p' \min\{1, qn\} \leq \frac{1}{2}$ . Then the expected optimization time, that is, the number of iterations until the optimum is sampled, is at most exponential in  $n$ .

With a noise probability of at most  $\frac{1}{2}$  and a weakly monotonic, that is, weakly preferring 1-bits over 0-bits, fitness function one would think that this optimization process in some suitable sense is at least as good as the corresponding process

on the NEEDLE function, where absolutely no fitness signal guides the search. This is indeed true, as the proof in [45] shows. Surprisingly, as this proof also shows, it is highly non-trivial to make this intuitive argument mathematically rigorous. The proof in [45] is around four pages long (including the one of the preliminary lemma) and builds on a technical estimate of the mixing time, which heavily exploits characteristics of the LEADINGONES objective function. Consequently, this proof does not easily generalize to other easy benchmark functions such as ONEMAX or linear functions.

### 1.3 Our Results

Observing that the natural approach taken in [45] is unexpectedly difficult, we develop an alternative approach to proving exponential upper bounds. It builds on the following elementary observation. In the, slightly extremal, situation that we aim at an exponential upper bound, we can wait for an exponentially unlikely “lucky” way to generate the optimum. Being at most exponentially unlikely, that is, having a probability of  $p = 2^{-O(n)}$ , it takes  $2^{O(n)}$  attempts until we succeed. Hence if each attempt takes at most exponential time  $T_0$  (all our attempts will only take polynomial time), we obtain an exponential upper bound on the expected runtime, and moreover, the distributional bound that the runtime is stochastically dominated by  $T_0$  times a geometric distribution with success rate  $p$ . This general argument (without the elementary rephrasing in the stochastic domination language) was already used in the proof of the  $\text{poly}(n)e^{2n}$  upper bound on the expected runtime of the  $(1 + 1)$  EA on the NEEDLE function by Droste, Jansen, and Wegener [18] more than twenty years ago. It is apparently not very well known in the community, most likely due to the fact that only one year later, Garnier, Kallel, and Schoenauer [24] presented a much tighter analysis of this runtime via different methods. We are not aware of any other use of this argument, which might explain why it was overlooked in [45] (and we give in that we also learned it only very recently).

How powerful this simple approach is, naturally, depends on how easy it is to exhibit lucky ways to find the optimum fast. As we demonstrate, this is in fact often easy. For example (see Theorem 3 for the details), it suffices that in each iteration the probability to move to a Hamming neighbor one step closer to the optimum is  $\Omega(n^{-1})$ . From this, we can show that from any starting point, the probability to reach the optimum in at most  $n$  iterations is at least  $2^{-O(n)}$ . As argued in the preceding paragraph, this yields an expected runtime of  $n2^{O(n)} = 2^{O(n)}$ . This argument, without noise and used for the  $(1 + 1)$  EA only, was also used in the NEEDLE analysis in [18].

Together with some elementary computations, this approach suffices to show that a large number of  $(1 + 1)$ -type algorithms in the presence of a large variety of types of noise with noise probability at most  $1 - \varepsilon$ ,  $\varepsilon > 0$  a constant, optimize any weakly monotonic function (including, e.g., ONEMAX, LEADINGONES, and the needle function) in at most exponential time (Theorem 4).

With a few additional arguments, we apply our general approach to a variety of other problems and show exponential upper bounds (i) for the  $(1 + 1)$  EA

optimizing jump functions with jump size at most  $\frac{n}{\ln n}$  (Theorem 5), (ii) for any of the above-described algorithms optimizing ONEMAX in the presence of prior noise flipping each bit independently with probability at most  $1 - \varepsilon$ , where  $\varepsilon > 0$  can be any constant (Theorem 6), and (iii) for the  $(1 + 1)$  EA with fitness-proportionate selection optimizing any linear function (Theorem 7). Finally, as an example that our approach can also yield sub-exponential upper bounds, we show that the  $(1, \lambda)$  EA with  $\lambda \geq (1 - \varepsilon) \log_{\frac{\varepsilon}{\varepsilon-1}}(n)$ , and thus potentially below the threshold for polynomial time, optimizes ONEMAX in time  $\exp(O(n^\varepsilon))$  (Theorem 8).

## 2 Preliminaries

In this section, we briefly describe the algorithms, the noise models, and the benchmark problems considered in this work. We only consider optimization problems defined on the search space  $\{0, 1\}^n$  of bit strings of length  $n$ ; we thus also formulate all algorithms only for this setting. We have not doubt, though, that our methods can also be applied to other discrete optimization problems.

We write  $[a..b] := \{z \in \mathbb{Z} \mid a \leq z \leq b\}$  and denote by  $H(x, y) := |\{i \in [1..n] \mid x_i \neq y_i\}|$  the *Hamming distance* of two bit strings  $x, y \in \{0, 1\}^n$ . We denote by  $\text{Geom}(p)$  the *geometric distribution* with success rate  $p \in (0, 1]$ . Hence if a random variable  $X$  is geometrically distributed with parameter  $p$ , we write  $X \sim \text{Geom}(p)$  to denote this, then  $\Pr[X = k] = (1 - p)^{k-1}p$  for all  $k \in \mathbb{Z}_{\geq 1}$ . For two random variables  $X, Y$  we write  $X \preceq Y$  to denote that  $Y$  *stochastically dominates*  $X$ , that is, that  $\Pr[X \geq \lambda] \leq \Pr[Y \geq \lambda]$  for all  $\lambda \in \mathbb{R}$ .

**Algorithms.** We call a randomized search heuristic *single-trajectory* search algorithm if it is an iterative heuristic which starts with a single solution  $x^{(0)}$  and in each iteration  $t = 1, 2, \dots$  updates this solution to a solution  $x^{(t)}$ . We do not make any assumption on how this update is computed. In particular, the next solution may be computed from more than one solution candidate sampled in this iteration. We do, in principle, allow that information other than the search point  $x^{(t-1)}$  is taken into iteration  $t$ . However, in our main technical result we require that the key condition can be checked only from the search point  $x^{(t-1)}$ . Formally speaking, this means that for any possibly history of the search process up to this point, when conditioning on this history, the key condition is true. To ease the language, we shall write “regardless of what happened in the first  $t - 1$  iterations” to express this conditioning.

Examples for single-trajectory algorithms are (randomized) local search, the Metropolis algorithm, simulated annealing, and evolutionary algorithms working with a parent population of size one, such as the  $(1 + 1)$  EA, the fast  $(1 + 1)$  EA [14],  $(1 + \lambda)$  EA,  $(1, \lambda)$  EA,  $(1 + (\lambda, \lambda))$  GA [11], and SSWM algorithm [40]. We call a single-trajectory algorithm  $(1 + 1)$ -*type algorithm* if in each iteration  $t$  it generates one solution  $y$  and takes as next parent individual  $x^{(t)}$  either  $y$  or  $x^{(t-1)}$ . Among the above examples, exactly (randomized) local search, the



Metropolis algorithm, simulated annealing, and the (fast)  $(1+1)$  EA are  $(1+1)$ -type algorithms.

We spare further details on these algorithms and refer the reader to the classic literature for the standard algorithms and to the references given above for the more recent algorithms. For evolutionary algorithms using standard bit mutation, we shall assume that the standard mutation rate  $p = \frac{1}{n}$  is used. For our purposes, we mostly need the following property, which in simple words says that the algorithms move to any Hamming neighbor that is not worse than the parent with probability  $\Omega(\frac{1}{n})$ .

**Proposition 1.** *For any  $(1+1)$ -type algorithm  $A$  named above (and any choice of the parameters not fixed yet), there is a constant  $c_A > 0$  such that the following holds.*

*For any iteration  $t$  and any  $z$  with  $H(z, x^{(t-1)}) = 1$ , and regardless of what happened in the previous iterations, the offspring  $y$  generated by  $A$  in iteration  $t$  satisfies  $\Pr[y = z] \geq \frac{c_A}{n}$ . If  $f(y) \geq f(x^{(t-1)})$ , then also  $\Pr[x^{(t)} = z] \geq \frac{c_A}{n}$ .*

**Noise Models.** Optimization in the presence of noise, that is, stochastically disturbed access to the problem instance, is an important topic in the optimization of real-world problems. The most common form are noisy objective functions, that is, that the optimization algorithm does not always learn the correct quality (fitness) of a search point. Randomized search heuristics are generally believed to be reasonably robust to noise, see, e.g., [5, 31], which differs from problem-specific deterministic algorithms, which often cannot cope with any noise. Some theoretical work exists on how randomized search heuristics cope with noise, started by the seminal paper of Droste [17] and, quite some time later, continued with, among others, [1, 4, 8, 9, 16, 22, 23, 25, 41, 42, 45, 46]. We refer to the later papers or the survey [36] for a detailed discussion of the state of the art.

In theoretical studies on how randomized search heuristics cope with noise, the usual assumption is that all fitness evaluations are subject to independently sampled noise. Also, it is usually assumed that whenever the fitness of a search point is used, say in a selection step, then it is evaluated anew. In *prior noise* models, the search point  $x$  to be evaluated is subject to a stochastic modification and the algorithm learns the fitness  $f$  of the disturbed search point (but not the disturbed search point itself). In **one-bit noise with probability  $p$** , with probability  $p$  the fitness of a random Hamming neighbor of  $x$  is returned, otherwise the correct fitness  $f(x)$  is returned. In **independent bit-flip noise with rate  $q$** , from  $x$  a search point  $y$  is obtained by flipping each bit independently with probability  $q$ ; then  $f(y)$  is returned. In  **$(p, q)$ -noise**, with probability  $p$  a search point  $y$  is obtained from  $x$  by flipping each bit independently with probability  $q$  and  $f(y)$  is returned; otherwise,  $f(x)$  is returned.

In the *posterior noise* model, the search point  $x$  is first correctly evaluated, but then the obtained fitness  $f(x)$  is disturbed. The most common posterior noise is **additive noise**, that is, the returned fitness is  $f(x) + X$ , where  $X$  is a random variable sampled from some given distribution, which does not depend on  $x$  (that is, for all search points the difference between the true and the noisy fitness is

identically distributed). The most common setting is that  $X$  follows a Gaussian distribution. We note that regardless of  $X$ , *independent additive posterior noise gives a correct comparison of two search points of different quality with probability at least  $\frac{1}{2}$* .

Since our aim is showing that also in the presence of extreme noise we still have at most exponential runtimes, we also consider the following **unrestricted adversarial noise with probability  $p$** . In this model, with probability  $1 - p$  the true fitness is returned. With probability  $p$ , however, an all-powerful adversary decides the returned fitness value. This adversary knows the algorithm, the optimization problem, and the full history of the optimization process. He does not know, though, the outcome of future random events (both concerning the algorithm and the noise).

Complementing the corresponding statement for posterior noise, the following basic observation estimates the probability that a noisy fitness comparison gives the right result.

**Proposition 2.** *Let  $\varepsilon > 0$ . Let  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ . Let  $x, y \in \{0, 1\}^n$  such that  $f(x) \leq f(y)$ . Consider any noise model described above except the one of additive posterior noise. Assume that  $p \leq 1 - \varepsilon$  in the case of one-bit noise or unrestricted adversarial noise,  $(1 - q)^n \geq \varepsilon$  in the case of bit-wise noise,  $1 - p(1 - (1 - q)^n) \geq \varepsilon$  in the case of  $(p, q)$ -noise. Denote by  $\tilde{f}$  the noisy version of  $f$  with our convention that each noise evaluation of  $f$  uses fresh independent randomness. Then  $\Pr[\tilde{f}(x) \leq \tilde{f}(y)] \geq \varepsilon^2$ .*

*Proof.* Under the conditions named above, with probability at least  $\varepsilon$  the noisy fitness returns the true fitness value. Consequently, with probability at least  $\varepsilon^2$  this happens for both  $x$  and  $y$  and we have thus  $\tilde{f}(x) \leq \tilde{f}(y)$ .

**Benchmark Problems.** We now briefly describe those benchmark problems for which the particular structure is important in the remainder. For further details on these and on all other problems only mentioned in this work, we refer to the literature [3, 15, 29, 38].

As said earlier, we only regard problems defined on bit-strings of length  $n$ , hence all functions are  $\{0, 1\}^n \rightarrow \mathbb{R}$ . The easiest in many respects benchmark problem is the function **OneMax** defined by  $\text{ONEMAX}(x) = \|x\|_1 = \sum_{i=1}^n x_i$  for all  $x = (x_1, \dots, x_n) \in \{0, 1\}^n$ . Still unimodal, but not anymore strictly monotonic is the classic **LeadingOnes** function, which counts the number of ones up to the first zero. Formally,  $\text{LEADINGONES}(x) := \max\{i \in [0..n] \mid \forall j \in [1..i] : x_j = 1\}$ . A classic multimodal benchmark is the class of **jump functions**. The jump function with *jump parameter (jump size)  $k \in [1..n]$*  is defined by

$$\text{JUMP}_{nk}(x) = \begin{cases} \|x\|_1 + k & \text{if } \|x\|_1 \in [0..n - k] \cup \{n\}, \\ n - \|x\|_1 & \text{if } \|x\|_1 \in [n - k + 1..n - 1]. \end{cases}$$

Hence for  $k = 1$ , we have a fitness landscape isomorphic to the one of **ONEMAX**, but for larger values of  $k$  there is a fitness valley (“gap”)  $G_{nk} := \{x \in \{0, 1\}^n \mid n - k < \|x\|_1 < n\}$ , which is impossible or hard to cross for most iterative search heuristics.

### 3 Proving Exponential Upper Bounds

We now state our general technical result which in many situations allows one to prove exponential upper bounds without greater difficulties. We formulate our result for single-trajectory algorithms since this is notationally convenient and covers all our applications (which, in fact, all even concern only  $(1 + 1)$ -type algorithms), but we are optimistic that it extends to more general settings. The result is formulated for hitting a general search point  $x^*$  as this might turn out to be useful in some applications, but the natural application will be for  $x^*$  being the optimum solution.

We remind the reader that the key argument of the proof, running from an arbitrary search point to the target in time  $O(n)$  with probability  $e^{-O(n)}$ , has already appeared in the conference paper [18], but to the best of our knowledge has not been used again since then.

**Theorem 3.** *Let  $A$  be a single-trajectory search algorithm for the optimization of pseudo-Boolean functions. Let  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  and let  $x^* \in \{0, 1\}^n$ . Assume that we use  $A$  to optimize  $f$ , possibly in the presence of noise. Assume that this optimization process satisfies the following property.*

- (A) *There is a number  $0 < c \leq 1$  such that the following is true. Let  $t \geq 1$  and  $x, z \in \{0, 1\}^n$  such that  $x \neq x^*$ ,  $H(x, z) = 1$ , and  $H(x, x^*) = H(z, x^*) + 1$ . Regardless of what happened in the first  $t-1$  iterations of optimization process, if  $x^{(t-1)} = x$ , then  $\Pr[x^{(t)} = z] \geq \frac{c}{n}$ .*

*Let  $T = \min\{t \geq 0 \mid x^{(t)} = x^*\}$ . Then  $T$  is stochastically dominated by  $n\text{Geom}(\frac{c}{e})^n$ . In particular,  $E[T] \leq n(\frac{e}{c})^n$ .*

### 4 Noisy Optimization of Weakly Monotonic Functions

We now prove that all  $(1 + 1)$ -type algorithms discussed in Sect. 2 optimize any weakly monotonic function in at most exponential time even in the presence of any noise discussed in Sect. 2 as long as the noise probability is at most  $1 - \varepsilon$ ,  $\varepsilon > 0$  a constant, in the cases of prior or adversarial noise. We recall that the only previous result in this direction [45] shows this claim in the particular case of the  $(1 + 1)$  EA optimizing the LEADINGONES function subject to one-bit or  $(p, q)$  prior noise with noise probability at most  $\frac{1}{2}$ .

We say that a function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  is *weakly monotonic* (or weakly monotonically increasing) if for all  $x, y \in \{0, 1\}^n$  the condition  $x \leq y$  (component-wise) implies  $f(x) \leq f(y)$ . The class of weakly monotonic functions includes, obviously, all strictly monotonic functions [7, 13, 28, 33] and thus in particular the classic benchmarks ONEMAX and linear functions with non-negative coefficients [12, 19, 49]. However, this class also contains more difficult functions like LEADINGONES, monotonic polynomials [47], plateau functions [2], and the needle function.

**Theorem 4.** *Let  $\varepsilon > 0$  be a constant. Let  $A$  be one of the randomized search heuristics RLS, the Metropolis algorithm, simulated annealing, or the (1+1) EA using standard bit mutation with mutation rate  $\frac{1}{n}$  or using the fast mutation operator with  $\beta > 1$ . Let  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  be any weakly monotonic function. Assume that  $A$  optimizes  $f$  under one of the following noise assumptions: one-bit noise or unrestricted adversarial noise with  $p \leq 1 - \varepsilon$ , bit-wise noise with  $(1 - q)^n \geq \varepsilon$ ,  $(p, q)$ -noise with  $1 - p(1 - (1 - q)^n) \geq \varepsilon$ , or posterior noise with an arbitrary noise distribution.*

*Then there is a constant  $C > 1$ , depending only on  $\varepsilon$  and the choice of  $A$ , such that the time  $T$  to sample the optimum  $(1, \dots, 1)$  of  $f$  is stochastically dominated by  $n \text{Geom}(C^{-n})$ . In particular, the expected optimization time is at most  $E[T] \leq nC^n$ .*

*Proof.* By Theorem 3, it suffices to show that condition (A) is satisfied for  $x^* = (1, \dots, 1)$ . To this aim, let  $x, z \in \{0, 1\}^n$  such that  $H(x, z) = 1$  and  $H(x, x^*) = H(z, x^*) + 1$ . Assume that for some iteration  $t$  the parent individual satisfies  $x^{(t-1)} = x$ . By Proposition 1, there is a constant  $c_A$  such that the offspring  $y$  generated by  $A$  in this iteration is equal to  $z$  with probability at least  $\frac{c_A}{n}$ . By the weak monotonicity of  $f$ , we have  $f(z) \geq f(x)$ . By Proposition 2 or the corresponding statement for additive posterior noise, there is a constant  $c_N = \min\{\frac{1}{2}, \varepsilon^2\}$  depending on the noise model such that the noisy evaluations of both  $x^{(t-1)}$  and  $y = z$  in iteration  $t$  return an at least as good fitness value for  $z$  as for  $x$ . In this case,  $A$  accepts  $z$  with probability one, that is, we have  $x^{(t)} = z$ . In summary, we have shown  $\Pr[x^{(t)} = z] \geq \frac{c_A c_N}{n}$  as desired. Now Theorem 3 immediately gives the claim with  $C = \frac{e}{c_A c_N}$ .

## 5 Other Applications of Our Method

To show the versatility of our general approach, we continue with a number of results of varying flavor.

**Noisy Optimization of Jump Functions.** We first show that the (1+1) EA can optimize noisy jump functions with jump size at most  $\frac{n}{\ln n}$  in exponential time. The main argument is that as lucky event we can regard the event that the algorithm progresses towards the optimum by one Hamming step per iteration until the local optimum is reached and then the optimum is reached in one step. The probability of this event is different from the one regarded before and the number of ways to approach the optimum is smaller by a factor of  $k!$  (which counts against us), but with the assumption  $k \leq \frac{n}{\ln n}$  we obtain the desired exponential runtime.

**Theorem 5.** *The result of Theorem 4 holds also for the (1+1) EA optimizing  $\text{JUMP}_{nk}$  when  $k \leq \frac{n}{\ln n}$ .*

**Optimization of OneMax Under Extreme Bit-Wise Noise.** The following result shows that our general method can also exploit particular noise models.

Here, for example, we show that ONEMAX can be optimized in exponential time even in the presence of bit-wise noise with constant rate  $q < 1$ . Recall that this means that the search point to be evaluated is disturbed in an expected number of  $qn$  bits! To prove this result, we cannot simply invoke Proposition 2, since with probability  $1 - o(1)$  the noisy fitness differs from the true fitness. Instead, we show that despite the noise, with probability at least  $\frac{1}{2}(1 - q)^2$  the better offspring is accepted.

**Theorem 6.** *Let  $\varepsilon > 0$  be a constant. Let  $A$  be one of the randomized search heuristics RLS, the Metropolis algorithm, simulated annealing, or the  $(1 + 1)$  EA using standard bit mutation with mutation rate  $\frac{1}{n}$  or using the fast mutation operator with  $\beta > 1$ . Consider optimizing the ONEMAX benchmark function via  $A$  in the presence of bit-wise noise with rate  $q \leq 1 - \varepsilon$ . Then the expected time to find the optimum is at most  $nK^n$ , where  $K$  is a constant depending on  $\varepsilon$  and the algorithm used.*

**Fitness Proportionate Selection.** We now prove an upper bound matching an exponential lower bound proven in [26], namely that the  $(1 + 1)$  EA needs at least exponential time to optimize any linear function with positive coefficients when the usual elitist selection is replaced by fitness-proportionate selection. Here an offspring  $y$  of the parent  $x$  is accepted with probability  $\frac{f(y)}{f(x)+f(y)}$  (and with probability  $\frac{1}{2}$  when  $f(x) + f(y) = 0$ ). We now show that this result is tight, that is, that an exponential number of iterations suffices to optimize any linear function with this algorithm. This follows easily from Theorem 3 by noting that in the selection step a Hamming neighbor with better fitness is accepted with probability at least  $\frac{1}{2}$ .

**Theorem 7.** *Let  $A$  be the  $(1 + 1)$  EA with fitness-proportionate selection. Let  $f$  be any linear function with positive coefficients. Then the first iteration  $T$  in which the optimum of  $f$  is generated satisfies  $E[T] \leq (2e^2)^n$ .*

**Subexponential Upper Bounds.** Finally, we show that our method is not restricted to showing runtime bounds that are exponential in the problem dimension. We recall that the  $(1, \lambda)$  EA is a simple non-elitist algorithm working with a parent population of size one, initialized with a random individual. In each iteration, the algorithm creates independently  $\lambda$  offspring via standard bit mutation (here: with mutation rate  $\frac{1}{n}$ ) and takes a random best offspring as new parent. In their very precise determination of the efficiency threshold of the  $(1, \lambda)$  EA on ONEMAX, Rowe and Sudholt [43] showed that the  $(1, \lambda)$  EA has a runtime of at least  $\exp(\Omega(n^{\varepsilon/2}))$  when  $\lambda \leq (1 - \varepsilon) \log_{\frac{e}{e-1}}(n)$ ,  $\varepsilon > 0$  a constant. We now show an upper bound of  $\exp(O(n^\varepsilon))$  for this runtime. We do not know what is the right asymptotic order of the exponent. From the fact that there is a considerable negative drift when the fitness distance is below  $\frac{n^\varepsilon}{2\lambda}$ , we would rather suspect that also a lower bound of  $\exp(\Omega(\frac{n^\varepsilon}{\lambda}))$  iterations, and hence  $\lambda \exp(\Omega(\frac{n^\varepsilon}{\lambda}))$  fitness evaluations, comes true. Since this is not the main topic of this work, we leave this an open problem.

**Theorem 8.** *Let  $0 < \varepsilon < 1$  be a constant. Then there is a constant  $C_\varepsilon$  such that for all  $\lambda \geq (1 - \varepsilon) \log_{\frac{e}{e-1}}(n)$  the expected runtime of the  $(1, \lambda)$  EA on ONEMAX is at most  $\exp(C_\varepsilon n^\varepsilon)$ .*

The main proof idea is to first exploit additive drift [27, 34] to reach in a short polynomial time of  $O(n^{2-\varepsilon})$  a search point in Hamming distance  $d_0 = \frac{2e^2 n^\varepsilon}{\lambda}$ . We then use an argument analogous to Theorem 3 to show that from such a search point, with probability at least  $\exp(-O(n^\varepsilon))$  the optimum is reached in  $d_0$  iterations. This then easily yields the claim.

## 6 Conclusion and Outlook

In this work, we argued for proving exponential runtime guarantees for evolutionary algorithms. With Theorem 3, we provided a simple and general approach to such problems. It easily gave exponential upper bounds for various algorithmic settings.

In this first work on exponential-time evolutionary algorithms, we have surely not developed the full potential of this perspective in evolutionary computation. The clearly most important question for future work is what can be said about the constant  $C$  in the  $\text{poly}(n)C^n$  runtime guarantee. A  $C$  less than 2 shows that the algorithm is superior to random or exhaustive search. Taking again the field of classic algorithms as example, another interesting question is if there are EAs with “nice” exponential runtimes such as, e.g., the  $1.0836^n$  runtime of the algorithm of Xiao and Nagamochi [50] for finding maximum independent sets in graphs with maximum degree 3.

Concerning the constant  $C$ , we note that the proof of [45], which also is not optimized for giving good constants, shows an upper bound that is at least  $\exp(3en) \geq (3480)^n$ . Under the noise assumptions taken in [45], we have a probability of at least  $c_N \geq \frac{1}{4}$  that parent and offspring are not subject to noise. Regarding the  $(1 + 1)$  EA, the probability that a particular Hamming neighbor of the parent is generated as offspring is at least  $c_A \geq \frac{1}{en}$ . This gives a runtime bound of at most  $n(\frac{e}{c_A c_N})^n = n(4e^2)^n \leq n(30)^n$ . We are optimistic that with more problem-specific arguments, the constant can be lowered further, possibly below the  $2^n$  barrier. For example, (i) when optimizing any weakly monotonic function subject to 1-bit noise, we accept an offspring strictly dominating the parent (as in the proof of Theorem 3) unless the noise flips a zero-bit of the parent or a one-bit of the offspring. This undesired event happens with probability at most  $c_N = \frac{1}{2}$  (instead of  $c_N = \frac{1}{4}$ ), (ii) when optimizing ONEMAX subject to 1-bit noise, then a better offspring is discarded only if both a one-bit of the offspring and a zero-bit of the parent is flipped. This allows to take  $c_N = (1 - O(\frac{1}{n}))\frac{15}{16}$ , (iii) when using the  $(1 + 1)$  EA, instead of waiting for the lucky event that in each iteration we approach the target by one Hamming step, we do so with two steps; this reduces the number of different ways to go from a starting point to the optimum by a factor of  $2^{n/2}$ , but also saves  $\frac{n}{2}$  times the factor of  $\frac{1}{e}$  for flipping exactly one bit, giving an improvement by a factor of

$(2/e)^{n/2}$ . These and further ideas give us some optimism that the constant  $C$  can be lowered, possibly to less than 2 (which would prove the algorithm superior to random search).

## References

1. Akimoto, Y., Morales, S.A., Teytaud, O.: Analysis of runtime of optimization algorithms for noisy functions over discrete codomains. *Theor. Comput. Sci.* **605**, 42–50 (2015)
2. Antipov, D., Doerr, B.: Precise runtime analysis for plateaus. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) *PPSN 2018*. LNCS, vol. 11102, pp. 117–128. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_10](https://doi.org/10.1007/978-3-319-99259-4_10)
3. Auger, A., Doerr, B. (eds.): *Theory of Randomized Search Heuristics*. World Scientific Publishing (2011)
4. Bian, C., Qian, C., Tang, K.: Towards a running time analysis of the  $(1 + 1)$ -EA for OneMax and LeadingOnes under general bit-wise noise. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) *PPSN 2018*. LNCS, vol. 11102, pp. 165–177. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_14](https://doi.org/10.1007/978-3-319-99259-4_14)
5. Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J.: A survey on metaheuristics for stochastic combinatorial optimization. *Nat. Comput.* **8**, 239–287 (2009). <https://doi.org/10.1007/s11047-008-9098-4>
6. Böttcher, S., Doerr, B., Neumann, F.: Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) *PPSN 2010*. LNCS, vol. 6238, pp. 1–10. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_1](https://doi.org/10.1007/978-3-642-15844-5_1)
7. Colin, S., Doerr, B., Férey, G.: Monotonic functions in EC: anything but monotone! In: *Genetic and Evolutionary Computation Conference, GECCO 2014*, pp. 753–760. ACM (2014)
8. Dang, D., Lehre, P.K.: Simplified runtime analysis of estimation of distribution algorithms. In: *Genetic and Evolutionary Computation Conference, GECCO 2015*, pp. 513–518. ACM (2015)
9. Dang-Nhu, R., Dardinier, T., Doerr, B., Izacard, G., Nogneng, D.: A new analysis method for evolutionary optimization of dynamic and noisy objective functions. In: *Genetic and Evolutionary Computation Conference, GECCO 2018*, pp. 1467–1474. ACM (2018)
10. Doerr, B.: Exponential upper bounds for the runtime of randomized search heuristics. *CoRR* abs/2004.05733 (2020)
11. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. *Theor. Comput. Sci.* **567**, 87–104 (2015)
12. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. *Algorithmica* **65**, 224–250 (2013). <https://doi.org/10.1007/s00453-011-9585-3>
13. Doerr, B., Jansen, T., Sudholt, D., Winzen, C., Zarges, C.: Mutation rate matters even when optimizing monotone functions. *Evol. Comput.* **21**, 1–21 (2013)
14. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: *Genetic and Evolutionary Computation Conference, GECCO 2017*, pp. 777–784. ACM (2017)



15. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation-Recent Developments in Discrete Optimization. Springer, Cham (2020). <https://doi.org/10.1007/978-3-030-29414-4>
16. Doerr, B., Sutton, A.M.: When resampling to cope with noise, use median, not mean. In: Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 242–248. ACM (2019)
17. Droste, S.: Analysis of the (1 + 1) EA for a noisy ONEMAX. In: Deb, K. (ed.) GECCO 2004. LNCS, vol. 3102, pp. 1088–1099. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24854-5\\_107](https://doi.org/10.1007/978-3-540-24854-5_107)
18. Droste, S., Jansen, T., Wegener, I.: On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 13–22. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0056845>
19. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. *Theor. Comput. Sci.* **276**, 51–81 (2002)
20. Fomin, F.V., Kaski, P.: Exact exponential algorithms. *Commun. ACM* **56**, 80–88 (2013)
21. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. TTCSAES. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-16533-7>
22. Friedrich, T., Kötzing, T., Krejca, M.S., Sutton, A.M.: Robustness of ant colony optimization to noise. In: Genetic and Evolutionary Computation Conference, GECCO 2015, pp. 17–24. ACM (2015)
23. Friedrich, T., Kötzing, T., Krejca, M.S., Sutton, A.M.: The compact genetic algorithm is efficient under extreme Gaussian noise. *IEEE Trans. Evol. Comput.* **21**, 477–490 (2017)
24. Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. *Evol. Comput.* **7**, 173–203 (1999)
25. Gießen, C., Kötzing, T.: Robustness of populations in stochastic environments. *Algorithmica* **75**, 462–489 (2016). <https://doi.org/10.1007/s00453-015-0072-0>
26. Happ, E., Johannsen, D., Klein, C., Neumann, F.: Rigorous analyses of fitness-proportional selection for optimizing linear functions. In: Genetic and Evolutionary Computation Conference, GECCO 2008, pp. 953–960. ACM (2008)
27. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. *Artif. Intell.* **127**, 51–81 (2001)
28. Jansen, T.: On the brittleness of evolutionary algorithms. In: Stephens, C.R., Toussaint, M., Whitley, D., Stadler, P.F. (eds.) FOGA 2007. LNCS, vol. 4436, pp. 54–69. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-73482-6\\_4](https://doi.org/10.1007/978-3-540-73482-6_4)
29. Jansen, T.: Analyzing Evolutionary Algorithms. The Computer Science Perspective. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-17339-4>
30. Jansen, T., Wegener, I.: On the analysis of a dynamic evolutionary algorithm. *J. Discrete Algorithms* **4**, 181–199 (2006)
31. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. *IEEE Trans. Evol. Comput.* **9**, 303–317 (2005)
32. Lehre, P.K.: Negative drift in populations. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 244–253. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_25](https://doi.org/10.1007/978-3-642-15844-5_25)
33. Lengler, J.: A general dichotomy of evolutionary algorithms on monotone functions. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 3–15. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_1](https://doi.org/10.1007/978-3-319-99259-4_1)



34. Lengler, J.: Drift analysis. *Theory of Evolutionary Computation*. NCS, pp. 89–131. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-29414-4\\_2](https://doi.org/10.1007/978-3-030-29414-4_2)
35. Mühlenbein, H.: How genetic algorithms really work: mutation and hill climbing. In: *Parallel problem solving from nature*, PPSN 1992, pp. 15–26. Elsevier (1992)
36. Neumann, F., Pourhassan, M., Roostapour, V.: Analysis of evolutionary algorithms in dynamic and stochastic environments. *Theory of Evolutionary Computation*. NCS, pp. 323–357. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-29414-4\\_7](https://doi.org/10.1007/978-3-030-29414-4_7)
37. Neumann, F., Sutton, A.M.: Parameterized complexity analysis of randomized search heuristics. *Theory of Evolutionary Computation*. NCS, pp. 213–248. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-29414-4\\_4](https://doi.org/10.1007/978-3-030-29414-4_4)
38. Neumann, F., Witt, C.: *Bioinspired Computation in Combinatorial Optimization - Algorithms and Their Computational Complexity*. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-16544-3>
39. Oliveto, P.S., Witt, C.: Improved time complexity analysis of the simple genetic algorithm. *Theoret. Comput. Sci.* **605**, 21–41 (2015)
40. Paixão, T., Heredia, J.P., Sudholt, D., Trubenová, B.: Towards a runtime comparison of natural and artificial evolution. *Algorithmica* **78**, 681–713 (2017)
41. Qian, C., Bian, C., Jiang, W., Tang, K.: Running time analysis of the  $(1 + 1)$ -EA for OneMax and LeadingOnes under bit-wise noise. *Algorithmica* **81**, 749–795 (2019)
42. Qian, C., Yu, Y., Zhou, Z.: Analyzing evolutionary optimization in noisy environments. *Evol. Comput.* **26**, 1–41 (2018)
43. Rowe, J.E., Sudholt, D.: The choice of the offspring population size in the  $(1, \lambda)$  evolutionary algorithm. *Theoret. Comput. Sci.* **545**, 20–38 (2014)
44. Sudholt, D.: A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Trans. Evol. Comput.* **17**, 418–435 (2013)
45. Sudholt, D.: Analysing the robustness of evolutionary algorithms to noise: refined runtime bounds and an example where noise is beneficial. *Algorithmica* (2020, to appear). <https://doi.org/10.1007/s00453-020-00671-0>
46. Sudholt, D., Thyssen, C.: A simple ant colony optimizer for stochastic shortest path problems. *Algorithmica* **64**, 643–672 (2012). <https://doi.org/10.1007/s00453-011-9606-2>
47. Wegener, I., Witt, C.: On the optimization of monotone polynomials by simple randomized search heuristics. *Comb. Probab. Comput.* **14**, 225–247 (2005)
48. 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)
49. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Comb. Probab. Comput.* **22**, 294–318 (2013)
50. Xiao, M., Nagamochi, H.: Confining sets and avoiding bottleneck cases: a simple maximum independent set algorithm in degree-3 graphs. *Theoret. Comput. Sci.* **469**, 92–104 (2013)



# Analysis on the Efficiency of Multifactorial Evolutionary Algorithms

Zhengxin Huang<sup>1,3</sup>, Zefeng Chen<sup>2</sup>, and Yuren Zhou<sup>1(✉)</sup>

<sup>1</sup> School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China

huangzhx26@mail2.sysu.edu.cn, zhouyuren@mail.sysu.edu.cn

<sup>2</sup> School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

zefeng.chen@ntu.edu.sg

<sup>3</sup> Department of Computer Science and Information Technology, Youjiang Medical University for Nationalities, Baise 533000, China

**Abstract.** Many experimental studies have demonstrated the superiority of multifactorial evolutionary algorithms (MFEAs) over traditional methods of solving each task independently. In this paper, we investigate this topic from theoretical analysis aspect. We present a runtime analysis of a (4+2) MFEA on several benchmark pseudo-Boolean functions, which include problems with similar tasks and problems with dissimilar tasks. Our analysis results show that, by properly setting the parameter *rm<sub>p</sub>* (i.e., the random mating probability), for the group of problems with similar tasks, the upper bound of expected runtime of the (4+2) MFEA on the harder task can be improved to be the same as on the easier one. As for the group of problems with dissimilar tasks, the expected upper bound of (4+2) MFEA on each task are the same as that of solving them independently. This study theoretically explains why some existing MFEAs perform better than traditional methods in experimental studies and provides insights into the parameter setting of MFEAs.

**Keywords:** Evolutionary multitasking · Multifactorial evolutionary algorithm · Running time analysis

## 1 Introduction

Evolutionary algorithms (EAs) are randomized search heuristics that are inspired from the process of natural evolution [1, 23]. EAs have been successfully applied to solve lots of real-world complex optimization problems in the past decades [8]. Recently, the notion of evolutionary multitasking [13] has emerged in the field of optimization and evolutionary computation, which aims to accelerate

---

Y. Zhou—This work was supported by the National Natural Science Foundation of China (61773410, 61673403).

© Springer Nature Switzerland AG 2020

T. Bäck et al. (Eds.): PPSN 2020, LNCS 12270, pp. 634–647, 2020.

[https://doi.org/10.1007/978-3-030-58115-2\\_44](https://doi.org/10.1007/978-3-030-58115-2_44)

convergence and enhance global search capabilities across simultaneously optimizing multiple problems (tasks) by sharing genetic information and transferring knowledge. For this purpose, multifactorial optimization (MFO) problems, which contain multiple distinct optimization tasks to be solved simultaneously, are formalized and a multifactorial evolutionary algorithm (MFEA) is proposed to tackle them in [13]. The key feature of MFEAs is that they are able to transfer genetic information and knowledge between different tasks during the evolutionary process. MFEAs have been shown to be effective in accelerating convergence and enhancing global search capabilities, with the help of positive knowledge transferring among tasks [20].

Although multiobjective optimization (MOO) and MFO are both problems that involve optimizing a set of objective functions (tasks) simultaneously, there are two main differences between them. First, tasks in MFEA can be independent while the objective functions in MOO are always conflicting to some extent. Second, tasks in MFO may have their own search spaces while all objectives in MOO have the same search spaces. In the optimization process, MFEAs aim to exploit latent genetic complementarity between multiple tasks using the implicit paradigm of population-based search, while multiobjective EAs (MOEAs) try to trade off several conflicting objectives of a problem efficiently [13].

MFEAs have received lots of research attentions in the field of evolutionary computation in the past years since the first MFEA [13] was proposed. Many search strategies and mechanisms have been employed to improve the performance of MFEAs, e.g., [2, 12, 17, 19, 20, 23]. Tang *et al.* [23] proposed a group-based MFEA (GMFEA), which groups tasks of similar types and selectively transfers the genetic information only within the groups. Feng *et al.* [12] proposed an improved MFEA with explicit genetic transfer across tasks, allowing the incorporation of multiple search mechanisms with different biases in the evolutionary multitasking paradigm. Experiments have demonstrated the advantages of their improved algorithms. MFEAs have been successfully applied to solve many real-world problems, e.g., cloud computing service composition (CCSC) problem [3], vehicle routing problem [11, 24], sparse reconstruction problem [18], modular training [4] or knowledge representation [5] in neural networks.

These previous studies investigate the superiorities of MFEAs from computational experiments. As far as we know, there are no research on investigating this topic from theoretical analysis aspect. Runtime (or running time) analysis is a powerful and essential theory tool to understand the performance and working principles of EAs [7, 9, 22]. In this paper, we present a runtime analysis of a simple (4+2) MFEA and (2+2) GA on optimizing several benchmark pseudo-Boolean functions, which include problems with similar tasks and problems with dissimilar tasks. Our analysis results show that, by properly setting the parameter *rpm*, for the group of problems with similar tasks, the upper bound of expected runtime of the (4+2) MFEA on the harder task can be improved to be the same as on the easier one. As for the group of problems with dissimilar tasks, the expected upper bound of (4+2) MFEA on each task are the same as that of solving them independently by the (2+2) GA. Furthermore, a simple

generalized result on the (4+2) MFEA is proved. This study theoretically explains why some existing MFEAs perform better than traditional methods in computational experiments and provides some insights into the working principle of MFEAs.

## 2 Preliminaries

### 2.1 Analyzed Algorithms

In MFEA, multiple tasks are simultaneously optimized by a population  $P$ , in which all individuals are encoded in a unified search space. Individuals in  $P$  can be decoded into a task-specific solution for every task with respect to its search space. The *factorial rank* of an individual on a given task is defined as the index of sorting the fitness values (in descending order for maximize problem) of all individuals on that task. The *scalar fitness* of an individual  $x_i$  is the reciprocal of the best factorial rank on all tasks. The *skill factor* of  $x_i$  represents its cultural bias, denoted as the index of the most effective task, on which the scalar fitness of the individual is obtained. Individuals with the same skill factor are devoted to the optimization the corresponding task. In some sense, this is equivalent to assigning a subpopulation of  $P$  for each task. MFEA is desired to accelerate the optimizations by exchanging knowledge between different subpopulations.

The main steps in the framework of MFEA presented in [13] are as follows:

- 1: Generate an initial population  $P$ .
- 2: Evaluate the fitness value of every individual in  $P$  with respect to every optimization task in the multitasking environment.
- 3: Compute the *skill factor* ( $\tau_i$ ) of each individual  $x_i \in P$ .
- 4: Apply genetic operators on  $P$  to generate an offspring population  $C$ , where each offspring only inherits the skill factor from their parents.
- 5: Compute the fitness values of every individual  $x_j$  in  $C$  for selected optimization tasks only, and set its fitness values on all other task to  $-\infty$ .
- 6: Update the *scalar fitness* and skill factor of every individual in  $P \cup C$ .
- 7: Select the fittest individuals from  $P \cup C$  to form the next population  $P$ . If stop condition is satisfied, output  $P$ . Otherwise go to Step 4.

Based on this framework, we propose a simple (4+2) MFEA, described in Algorithm 1, for MFO problems. In the initialization, the algorithm first generates 4 individuals in the unified search space uniformly at random and evaluates their fitness values for every task. Then it computes the skill factor ( $\tau_i$ ) for each individual  $x_i$  in  $P$  and selects the 2 fittest individuals with respect to task  $j$  in  $P$  to form  $P_0$ . In some sense, this is equivalent to assigning a subpopulation in  $P$  for each task. To simplify the analysis and expression, we assume that the skill factors of the 2 fittest individuals with respect to task  $j$  are  $j$ . We claim that this assumption is reasonable, because if an individual  $x_i$  with  $\tau_i \neq j$  is one of the 2 fittest individuals for task  $j$ , we can think that it is an additional individual in  $P_i$  and its fitness value on the other task are artificially set to  $-\infty$ . Note that this operation will not increase the expected number of fitness evaluations, i.e., the expected runtime of Algorithm 1.

**Algorithm 1.** A (4+2) MFEA for two-task optimization**Input:**  $K = 2$  optimization tasks, stopping criterion and parameter  $rpm$ .**Output:** The best individual in the population for each task.

- 1: Generate population  $P$  with 4 individuals in  $\{0, 1\}^n$  uniformly at random.
- 2: Evaluate the fitness values of every individual in  $P$  with respect to every task.
- 3: Set  $t := 0$ . Compute the *skill factor* ( $\tau_i$ ) of every individual  $x_i$  in  $P$ , and select the 2 fittest individuals with respect to each task  $j$  to form population  $P_0 := \{x_1, x_2, x_3, x_4\}$ .
- 4: **while** stopping criterion is unsatisfied **do**
- 5:   Set  $C := \emptyset$ . Select 2 parent individuals from  $P_t$ , denoted as  $p_a$  and  $p_b$ , uniformly at random.
- 6:   **if**  $\tau_a \neq \tau_b$  and  $rand \geq rpm$  **then**
- 7:     Create 2 offspring  $c_a$  and  $c_b$  by respectively applying standard bit mutation to  $p_a$  and  $p_b$ , and inherit the skill factor of its unique parent ( $\tau_a$  or  $\tau_b$ ).
- 8:   **else**
- 9:     Create 2 offspring  $c_a$  and  $c_b$  by applying one-point crossover on  $p_a$  and  $p_b$ , and inherit the skill factors of their parents ( $\tau_a$  and  $\tau_b$ ).
- 10:   **end if**
- 11:   Set  $C := \{c_a, c_b\}$ .
- 12:   Compute the fitness value of every individual in  $C$  on the task related to the skill factor, and set its fitness values on all other tasks to  $-\infty$ .
- 13:   Update the *scalar fitness* and skill factor of every individual in the population  $P_t \cup C$ .
- 14:   Select the 2 fittest individuals for each task  $j$  from  $P_t \cup C$  to form  $P_{t+1}$ . Set  $t := t + 1$ .
- 15: **end while**

Mutation and crossover are both employed as variation operators in Algorithm 1. For crossover operator (line 9 in Algorithm 1), it follows the rule that the two parent individuals ( $p_a$  and  $p_b$ ) possessing the same skill factor ( $\tau_a = \tau_b$ ), i.e., selected from the same subpopulation, can crossover freely. In some sense, this is similar to the fact that people tend to marry with ones belonging to the same cultural background [13]. Moreover, as argued in [13], if the two parent individuals have different skill factors, crossover operator will be also applied in some random rounds, which are controlled by parameter  $rpm$ . Otherwise, mutation operator is executed (line 7 in Algorithm 1). Note that  $rpm$  is an important parameter in Algorithm 1 since it determines the chance of exchanging genetic information between distinct tasks. For specific variation operators, we consider the typical one-point crossover and standard bit mutation [15] (flipping each bit in the solution (bit-string) with independent probability  $p_m = \frac{1}{n}$ , where  $n$  is the length of the bit-string), respectively.

After offspring individuals are created, inheritance strategy in the view of biological culture where offspring is directly affected by the phenotype of its parents [6, 10] is applied Algorithm 1. If the offspring is created by crossover, it inherits the two skill factors of its parents (line 9), and the offspring only inherits a skill factor from its unique parent if it is created by mutation (line 7).

---

**Algorithm 2.** (2+2) GA

---

**Input:** An optimization task, stopping criterion and crossover probability  $p_c$ .

**Output:** The best individual in the population.

- 1: Set  $t := 0$ . Generate population  $P_0 := \{x_1, x_2\}$  in search space  $\{0, 1\}^n$  uniformly at random.
  - 2: **while** stopping criterion is unsatisfied **do**
  - 3:   Set  $C := \emptyset$ .
  - 4:   **if**  $\text{rand} \geq p_c$  **then**
  - 5:     Create two offspring  $c_1$  and  $c_2$  by using standard bit mutation operator to  $x_1$  and  $x_2$ , respectively.
  - 6:   **else**
  - 7:     Create two offspring  $c_1$  and  $c_2$  by using one-point crossover operator to  $x_1$  and  $x_2$ .
  - 8:   **end if**
  - 9:   Set  $C := \{c_1, c_2\}$ .
  - 10:   Select the 2 fittest individuals from  $P_t \cup C$  to form  $P_{t+1}$ . Set  $t := t + 1$ .
  - 11: **end while**
- 

As argument in [13], to reduce the total number of consumed fitness evaluations, the fitness values of offspring individuals on all unevaluated tasks are artificially set to  $-\infty$  (line 12). In the environment selection (line 14), elitist strategy, only accepting not worse solutions, is applied. Specifically, the algorithm selects the 2 fittest individuals for each task  $j$  from  $P_t \cup C$  to form a new population  $P_{t+1}$ .

To compare and illustrate the optimization ability of the MFEA, we also analyze the expected runtime of a (2+2) GA, which is the single task optimization version of Algorithm 1, optimizing all considered problems independently. The (2+2) GA is shown in Algorithm 2.

**2.2 Problems**

For a given bit-string (solution)  $x$ , we let  $x[i]$  denote the value of the  $i$ -th bit in it. The analyzed pseudo-Boolean functions in this paper are described as follows. They have been widely used in the field of runtime analysis for EAs [15].

**Definition 1 (OneMax).** For any  $x \in \{0, 1\}^n$ , the pseudo-Boolean function  $\text{ONEMAX} : \{0, 1\}^n \rightarrow \mathbb{N}$  is defined as

$$\text{ONEMAX}(x) = \sum_{i=1}^n x[i].$$

As shown Definition 1, the function value of  $\text{ONEMAX}$  becomes better when increasing the number of 1-bits in solution  $x$ .

**Definition 2 (LeadingOnes).** For any  $x \in \{0, 1\}^n$ , the pseudo-Boolean function  $\text{LEADINGONES} : \{0, 1\}^n \rightarrow \mathbb{N}$  is defined as

$$\text{LEADINGONES}(x) = \sum_{i=1}^n \prod_{j=1}^i x[j].$$

For LEADINGONES, the function value increases when increasing the number of leading 1-bits in solution  $x$ .

**Definition 3 (TrailingZeros).** For any  $x \in \{0, 1\}^n$ , the pseudo-Boolean function TRAILINGZEROS :  $\{0, 1\}^n \rightarrow \mathbb{N}$  is defined as

$$\text{TRAILINGZEROS}(x) = \sum_{i=1}^n \prod_{j=i}^n (1 - x[j]).$$

For TRAILINGZEROS, the function value increases when increasing the number of trailing 0-bits in solution  $x$ .

In this analysis, ONEMAX and LEADINGONES are set as the first group of problems while ONEMAX and TRAILINGZEROS are set as the second group of problems. Note that when increasing the number of 1-bits in a solution, the function values of ONEMAX and LEADINGONES will never become worse, while the function value of TRAILINGZEROS will never become better. Thus, the two groups of problems can serve as the multitasking optimization problems with similar tasks and the one with dissimilar tasks for MFEAs, respectively.

### 3 Runtime Analysis

#### 3.1 Analysis on (2+2) GA

In this subsection, we analyze the expected runtime of the (2+2) GA to optimize ONEMAX, LEADINGONES and TRAILINGZEROS problems. Note that the (2+2) GA optimizes the three problems independently. For ease of express, we assume that the fitness value of  $x_1$  is not smaller than that of  $x_2$  for any  $P_t = \{x_1, x_2\}$  in the (2+2) GA.

**Theorem 1.** For any constant crossover probability  $p_c < 1$ , Algorithm 2 finds the optimal solution for ONEMAX in expected runtime  $O(n \log n)$ .

*Proof.* We use the fitness-based partition method [15, 21] to prove this theorem. We partition the search space into  $n + 1$  disjoint layers according their fitness values, i.e.,  $L_i = \{x \in \{0, 1\}^n \mid \text{ONEMAX}(x) = i\}$  for  $i = 0, 1, \dots, n$ . Thus, the numbers of 1-bits and 0-bits in any solution in  $L_i$  are  $i$  and  $n - i$ , respectively. It is not difficult to see that the optimal solution  $1^n$  lies in the  $n$ -th layer. In the following, we show that for any population  $P_t$ , if  $x_1$  lies in  $L_i$  for  $i < n$ , then in that generation Algorithm 2 creates an offspring in  $L_j$  for  $j > i$ , namely better than  $x_1$ , with probability at least  $\frac{(1-p_c)(n-i)}{en}$ .

Recall that in a generation, the probability of Algorithm 2 executing the crossover and mutation operator are  $p_c$  and  $1 - p_c$ , respectively. In a mutation step, the two offspring are created by respectively using the standard bit mutation operator to  $x_1$  and  $x_2$ , that is, flipping any bit in  $x_1$  and  $x_2$  with independent probability  $\frac{1}{n}$ . If the mutation flips one of 0-bits in  $x_1$  into 1-bit (denoted as  $E_1$ ) and keeps all other bits unchanged (denoted as  $E_2$ ), then an offspring

in  $L_{i+1}$  is created. The probability of this event is  $s_m = Pr(E_1) \cdot Pr(E_2) = \binom{n-i}{1} \cdot \frac{1}{n} \cdot (1 - \frac{1}{n})^{n-1} \geq \frac{n-i}{en}$ . In a crossover step, for the selected crossover point  $j \in [1, n]$  if  $\sum_{i=1}^j x_1[i] < \sum_{i=1}^j x_2[i]$  or  $\sum_{i=j+1}^n x_1[i] < \sum_{i=j+1}^n x_2[i]$ , then an offspring with 1-bits more than  $x_1$ , i.e., lie in layer higher than  $x_1$ , is created. Let  $s_c$  denote the probability of this event. The exact value of  $s_c$  is hard to estimate since it depends on the distribution of 1-bits in  $x_1$  and  $x_2$ . But it is obvious that  $s_c \geq 0$ . Thus, in a generation, the probability of Algorithm 2 creating an offspring in layer higher than  $i$  is at least

$$s_i \geq (1 - p_c) \cdot s_m + p_c \cdot s_c \geq \frac{(1 - p_c)(n - i)}{en}.$$

Therefore, the expected runtime of Algorithm 2 optimizing ONEMAX is upper bounded by

$$2 \sum_{i=0}^{n-1} \frac{1}{s_i} \leq \frac{en}{1 - p_c} \sum_{i=0}^{n-1} \frac{1}{n - i} = O(n \log n).$$

Note that Algorithm 2 spends two fitness evaluations in any generation and  $p_c < 1$  is a constant. □

**Theorem 2.** *For any constant crossover probability  $p_c < 1$ , Algorithm 2 finds the optimal solution for LEADINGONES and TRAILINGZEROS in expected runtime  $\Theta(n^2)$ .*

In the following, we only give the detailed proof of expected runtime  $\Theta(n^2)$  for LEADINGONES since the proof for TRAILINGZEROS is similar. The only difference is that for LEADINGONES the algorithm accepts offspring with increasing number of leading 1-bits while it accepts offspring with increasing number of trailing 0-bits for TRAILINGZEROS.

*Proof.* We first use the fitness-based partition method to prove the expected upper bound of  $O(n^2)$ . We partition the search space into  $n + 1$  layers, where  $L_i = \{x \in \{0, 1\}^n \mid \text{LEADINGONES}(x) = i\}$  for  $i = 0, 1, \dots, n$ . We show that for any  $P_t = \{x_1, x_2\}$ , in that generation Algorithm 2 creates an offspring in higher layer with probability at least  $\frac{1-p_c}{en}$  if  $x_1$  lies in  $L_i$  for  $i < n$ .

Before computing a lower bound of progress in a generation for the expected upper bound, we first claim that for any  $P_t$ , all bits next to the leftmost 0-bit in  $x_1$  and  $x_2$  are distributed uniformly at random. Intuitively, these bits are initialized uniformly, and in later iterations the algorithm selects individuals for reproduction according to their fitness values and these bits have never contributed to the fitness. Formally, from the proof of Theorem 5.16 in [15], we know that the claim holds for any mutation step. For a crossover step, it creates two offspring by recombining two parents where all bits after the leftmost 0-bit are uniform distributed. So the bits that after the leftmost 0-bit in the two fittest individuals in  $P_t \cup C$  are uniform distributed. Recall that the two fittest individuals are survived in any generation. Thus, the claim also holds for any crossover step.



In a mutation step, if the leftmost 0-bit in  $x_1$  is flipped and all leading 1-bits are kept unchanged, then an offspring in  $L_{i+1}$  is created. The probability of this event is  $s_m = \frac{1}{n} \cdot (1 - \frac{1}{n})^i \geq \frac{1}{en}$ . Note that if  $x_1$  and  $x_2$  are in the same layer, the probability that an offspring in  $L_{i+1}$  is created by applying the mutation operator to  $x_2$  is also at least  $s_m$ . For a crossover step, if  $x_1$  and  $x_2$  are in  $L_i$ , the algorithm creates an offspring in layer higher than  $i$  with probability 0, because of  $x_1[i+1] = x_2[i+1] = 0$ . If  $x_2$  lies in layer lower than  $x_1$ , the algorithm creates an offspring in  $L_{i+1}$  with probability at least  $\frac{1}{2n}$ , since it happens if the crossover position is  $i$  and the value of the  $(i+1)$ -th bit in  $x_2$  is 1. The probability of the first and second event are  $\frac{1}{n}$  and  $\frac{1}{2}$ , respectively.

Thus, in the  $t$ -th generation Algorithm 2 creates an offspring in layer higher than  $i$  with probability at least

$$s_i \geq \min\{(1 - p_c)[1 - (1 - s_m)^2], (1 - p_c)s_m + \frac{p_c}{2n}\} \geq (1 - p_c)s_m \geq \frac{1 - p_c}{en}.$$

Therefore, the expected runtime of Algorithm 2 finding the optimal solution for LEADINGONES is upper bounded by

$$2 \sum_{i=0}^{n-1} \frac{1}{s_i} \leq \frac{2en}{1 - p_c} \sum_{i=0}^{n-1} 1 = O(n^2).$$

We now prove the lower bound  $\Omega(n^2)$  by using the drift analysis method [14, 16]. We define the following random process  $X_t := n - \text{LEADINGONES}(x_1)$  for each  $P_t = \{x_1, x_2\}$ . Recall that  $x_1$  denotes the better individual in  $P_t$ . It is not difficult to see that  $0 \leq X_t \leq n$  for any  $t \in \mathbb{N}$  and  $X_t = 0$  if and only if the optimal solution is contained in  $P_t$ . Since individuals in  $P_0$  are generated in  $\{0, 1\}^n$  uniformly at random, there are exact  $i < n$  ( $i = n$ ) leading 1-bits in  $x_1$  with probability  $2^{-(i+1)}$  ( $2^{-n}$ ). Thus, we have

$$E(X_0) = n - \frac{n}{2^n} - \sum_{i=0}^{n-1} i \cdot 2^{-(i+1)} = n - 1 + \frac{1}{2^n} \geq n - 1.$$

For a mutation step, the necessary condition of  $X_t - X_{t+1} = j > 0$  is that the leftmost 0-bit in  $x_1$  and the  $h$  0-bits in the  $(j - 1)$  bits next to it are flipped simultaneously. The probabilities of the first and the second event are  $\frac{1}{n}$  and  $(\frac{1}{n})^h \cdot 2^{-(j-1)+h}$ , respectively. Note that the values of all bits next to the leftmost 0-bit in  $x_1$  and  $x_2$  are distributed uniformly at random. Thus, in a mutation step we have  $E(X_t - X_{t+1}) \leq \sum_{j=1}^n j \cdot \frac{1}{n} \cdot (\sum_{h=0}^{j-1} \frac{1}{n^h} \cdot 2^{-(j-1)+h}) \leq \sum_{j=1}^n j \cdot \frac{1}{n} \cdot \frac{3}{2^{j-1}} \leq \frac{12}{n}$ .

For a crossover step, if  $x_1$  and  $x_2$  have the same fitness value, then  $E(X_t - X_{t+1}) = 0$ . Otherwise, we first consider the case that the crossover point is the leftmost 0-bit in  $x_1$  (denoted as  $l_0$ ). If and only if there are  $j$  1-bits next to  $l_0$  in  $x_2$ , then  $X_t - X_{t+1} = j \geq 1$  holds. The probabilities of the first and the second event are  $\frac{1}{n}$  and  $2^{-j}$ , respectively. Thus, in this case we have  $E(X_t - X_{t+1}) = \sum_{j=1}^n j \cdot \frac{1}{n} \cdot \frac{1}{2^j} \leq \frac{2}{n}$ . For the case that the crossover point is the  $(l_0 - 1)$ -th bit, if and only if there are  $(j + 1)$  1-bits next to the  $(l_0 - 1)$ -th bit in  $x_2$ , then

$X_t - X_{t+1} = j$  also holds. The probabilities of the two events are  $\frac{1}{n}$  and  $2^{-(j+1)}$ , respectively. So we have  $E(X_t - X_{t+1}) = \sum_{j=1}^n j \cdot \frac{1}{n} \cdot \frac{1}{2^{j+1}} \leq \frac{2}{n} \cdot \frac{1}{2}$ . By analogy, we have that  $E(X_t - X_{t+1}) \leq \frac{2}{n} \cdot \frac{1}{2^i}$  for the case that the crossover point is the  $(l_0 - i)$ -th bit. Thus, in a crossover step we have  $E(X_t - X_{t+1}) = \sum_{i=1}^{l_0} \frac{2}{n} \cdot \frac{1}{2^{l_0-i}} \leq \sum_{i=1}^n \frac{2}{n} \cdot \frac{1}{2^{n-i}} \leq \frac{4}{n}$ .

In summary, in the  $t$ -th generation we have

$$E(X_t - X_{t+1}) \leq (1 - p_c) \cdot \frac{12}{n} + p_c \cdot \frac{4}{n} \leq \frac{12}{n} = \delta.$$

Hence, the expected runtime of Algorithm 2 optimizing LEADINGONES is lower bounded by

$$E(T|X_0) = \frac{E(X_0)}{\delta} \geq 2(n - 1) \cdot \frac{n}{12} = \Omega(n^2).$$

Therefore, combined with the upper bound of  $O(n^2)$ , the expected runtime of  $\Theta(n^2)$  is proved.  $\square$

### 3.2 Analysis on (4+2) MFEA

Different from the (2+2) GA, where one task is optimized by the whole population, the (4+2) MFEA simultaneously optimizes two tasks with a population of 4 individuals, which is divided into two subpopulations with size of 2 each. Since the population in Algorithm 1 are divided into two subpopulations for optimizing the two tasks, for any  $P_t = \{x_1, x_2, x_3, x_4\}$ , in following analyses we assume that subpopulations  $P_{t,1} = \{x_1, x_2\}$  and  $P_{t,2} = \{x_3, x_4\}$  are assigned to optimize task 1 and task 2, respectively. And the better individuals in  $P_{t,1}$  and  $P_{t,2}$  are respectively  $x_1$  and  $x_3$ .

**Lemma 1.** *For Algorithm 1, the probabilities of executing the mutation and crossover operators in a generation are  $\frac{2(1-rmp)}{3}$  and  $\frac{1+2rmp}{3}$ , respectively.*

Note that in a generation, Algorithm 1 executes crossover operator in the following two cases: (1)  $\tau_a = \tau_b$ , that is,  $p_a$  and  $p_b$  are selected from the same subpopulation, the probability is  $2 \cdot \binom{4}{2}^{-1} = \frac{1}{3}$ ; (2)  $\tau_a \neq \tau_b$  and  $rand < rmp$ , the probability is  $(1 - \frac{1}{3}) \cdot rmp = \frac{2rmp}{3}$ . So the probabilities of Algorithm 1 executing the crossover and mutation operators in a generation are  $\frac{1+2rmp}{3}$  and  $\frac{2(1-rmp)}{3}$ , respectively. Thus, in a generation Algorithm 1 consumes  $2 \cdot (1 - \frac{2rmp}{3}) + 4 \cdot \frac{2rmp}{3} = \frac{6+4rmp}{3}$  fitness evaluations on expectation. In the following, we analyze the expected runtime of Algorithm 1 to simultaneously optimize the group problems with dissimilar tasks.

**Theorem 3.** *To optimize TRAILINGZEROS and ONEMAX problems simultaneously, Algorithm 1 finds their optimal solutions in expected runtime  $O(\frac{(6+4rmp)n^2}{1-rmp})$  and  $O(\frac{(6+4rmp)n \log n}{1-rmp})$ , respectively.*

*Proof.* For ONEMAX problem (denoted as task 1), similar to the proof of Theorem 1, we partition the search space into  $n + 1$  disjoint layers according to their fitness values, that is,  $L_i = \{x \in \{0, 1\}^n \mid \text{ONEMAX}(x) = i\}$  for  $i = 0, 1, \dots, n$ . Assume that in the current population  $P_t$ ,  $x_1$  lies in  $L_i$  for  $i < n$ . We show that Algorithm 1 creates an offspring in  $L_{i+1}$  with probability at least  $\frac{(1-rmp)(n-i)}{3en}$ .

In a mutation step, the parent individuals are selected from different subpopulations, namely one from  $P_{t,1}$  and the other from  $P_{t,2}$ . If the mutation operator flips one of the  $(n-i)$  0-bits in  $x_1$  and keeps all other bits unchanged, then an offspring in  $L_{i+1}$  is created. The probability of this event is  $\frac{1}{2} \cdot \frac{n-i}{n} \cdot (1-\frac{1}{n})^{n-1} \geq \frac{n-i}{2en}$ . Note that individual  $x_1$  is selected from  $P_{t,1}$  to be a parent with probability  $\frac{1}{2}$ . For a crossover step, the two parent individuals can be  $x_1$  and  $x_2$  or one from  $P_{t,1}$  and the other from  $P_{t,2}$ . As discussed in the proof of Theorem 1, for ONEMAX the probability of Algorithm 1 creating an offspring in  $L_{i+1}$  is hard to estimate since it depends on the distribution of 1-bits in the two parent individuals. However the trivial lower bound of 0 holds. Thus, if  $x_1 \in P_t$  lies in  $L_i$  for  $i < n$ , Algorithm 1 creates an offspring in  $L_j$  for  $j > i$  in the generation with probability at least  $\frac{2(1-rmp)}{3} \cdot \frac{n-i}{2en} \geq \frac{(1-rmp)(n-i)}{3en}$ .

Therefore, the upper bound of expected runtime of Algorithm 1 optimizing ONEMAX is

$$\sum_{i=0}^{n-1} \frac{(6 + 4rmp)3en}{3(1 - rmp)(n - i)} = O\left(\frac{(6 + 4rmp)n \log n}{1 - rmp}\right). \tag{1}$$

For TRAILINGZEROS problem (task 2), we partition the search space into  $n + 1$  disjoint layers according to their fitness values, that is,  $L_i = \{x \in \{0, 1\}^n \mid \text{TRAILINGZEROS}(x) = i\}$  for  $i = 0, 1, \dots, n$ . Thus there are exact  $i$  trailing 0-bits in any solution in  $L_i$ .

Assume that individual  $x_3$  in  $P_t$  lies in  $L_i$  for  $i < n$ . For a mutation step, if the algorithm flips the leftmost 1-bit in  $x_3$  and keeps all  $i$  trailing 0-bits unchanged, an offspring in  $L_{i+1}$  is created. Thus, in the  $t$ -th generation Algorithm 1 creates an offspring in  $L_{i+1}$  with probability at least  $\frac{2(1-rmp)}{3} \cdot \frac{1}{2} \cdot \frac{1}{n} \cdot (1 - \frac{1}{n})^{n-i} \geq \frac{1-rmp}{3en}$ . Recall that in a generation Algorithm 1 executes the mutation operator with probability  $\frac{2(1-rmp)}{3}$  and a crossover step creates an offspring in  $L_{i+1}$  with probability at least 0. Therefore, the expected runtime of Algorithm 1 optimizing TRAILINGZEROS is upper bounded by

$$\sum_{i=0}^{n-1} \frac{(6 + 4rmp)3en}{3(1 - rmp)} = O\left(\frac{(6 + 4rmp)n^2}{1 - rmp}\right). \tag{2}$$

Furthermore, if parameter  $rmp$  is set to a constant in  $(0, 1)$ , by Eqs. (1) and (2), we know that Algorithm 1 finds the optimal solutions for ONEMAX and TRAILINGZEROS in expected runtime  $O(n \log n)$  and  $O(n^2)$ , respectively.  $\square$

Different from Algorithm 2, the lower bound of progress probability  $\frac{1}{2n}$  in a crossover step (see the proof of Theorem 2) is not ensured for Algorithm

1, because applying crossover to individuals selected from  $P_{t,2}$  and  $P_{t,1}$  means that these bits before the rightmost 1-bit are not uniformly distributed during the whole evolutionary process. Thus, applying crossover to individuals with distinct skill factors in Algorithm 1 cannot accelerate the optimization of TRAILINGZEROS. Instead, it may lose the progress probability  $\frac{1}{2n}$ , compared with Algorithm 2, in a crossover step. We next show that to optimize ONEMIN and LEADINGONES simultaneously, such a mechanism can apparently reduce the optimizing time of LEADINGONES. This indicates that the knowledge transfer between different tasks in MFEA can really help optimizing in some cases.

**Theorem 4.** *To optimize ONEMAX and LEADINGONES problems simultaneously, Algorithm 1 finds their optimal solutions in expected runtime  $O((6 + 4rmp) \cdot (\frac{n \log n}{rmp} + \frac{n \log n}{1-rmp}))$ . Furthermore, the expected runtime is  $O(n \log n)$  if parameter  $rmp$  is set to a constant in  $(0, 1)$ .*

*Proof.* Let task 1 and task 2 denote the ONEMAX and LEADINGONES functions, respectively. From the proof of Theorem 3, we have that the optimal solution  $1^n$  of ONEMAX is obtained by subpopulation  $P_{t,1}$  in expected runtime  $O(\frac{(6+4rmp)n \log n}{1-rmp})$ . Afterward, solution  $1^n$  will be kept in  $P_{t,1}$  forever since Algorithm 1 is elitist.

We now consider these solutions created by crossover operator in the case that  $\tau_a \neq \tau_b$  and  $rand < rmp$  after solution  $1^n$  has been added into  $P_{t,1}$ . Note that any offspring created by such a crossover step will be evaluated on the two tasks since they inherit the skill factors of their parents (see line 8 in Algorithm 1). In the following, we show that in generation  $t$  Algorithm 1 creates an improved solution for LEADINGONES with probability at least  $\frac{rmp(n-j)}{3n}$ , where  $j$  denotes the number of leading 1-bits in  $x_3 \in P_{t,2}$ .

First, by Lemma 1, such a crossover step happens in a generation with probability  $\frac{2rmp}{3}$ . Second, in such a crossover step, if the individual  $x_1 = 1^n$  in  $P_{t,1}$  is selected to be a parent (with probability at least  $\frac{1}{2}$ ) and the crossover point is larger than  $j$ , then an offspring with leading 1-bits larger than  $j$  (an improved an improved solution for LEADINGONES) is created. Thus, an improved solution for LEADINGONES is created with probability at least  $\frac{2rmp}{3} \cdot \frac{1}{2} \cdot \binom{n-j}{1} \cdot \frac{1}{n} = \frac{rmp(n-j)}{3n}$ . Note that any bit is selected to be the crossover point with probability  $\frac{1}{n}$ . Hence, after solution  $1^n$  has been added into  $P_{t,1}$ , the expected runtime of Algorithm 1 finding the optimal solution for LEADINGONES is upper bounded by

$$\frac{6 + 4rmp}{3} \sum_{j=0}^{n-1} \frac{3n}{rmp(n-j)} = O(\frac{(6 + 4rmp)n \log n}{rmp}).$$

Therefore, Algorithm 1 finds the optimal solution for LEADINGONES in expected runtime

$$O(\frac{(6 + 4rmp)n \log n}{1 - rmp}) + O(\frac{(6 + 4rmp)n \log n}{rmp}) = O((6 + 4rmp) \cdot (\frac{n \log n}{rmp} + \frac{n \log n}{1 - rmp})). \tag{3}$$

From Eq. (3), we have that expected runtime is  $O(n \log n)$  if parameter  $rpm$  is a constant in  $(0, 1)$ .  $\square$

Theorem 4 shows that by properly setting the value of parameter  $rpm$ , the upper bound of expected runtime of Algorithm 1 on LEADINGONES can be improved to  $O(n \log n)$ , which is the same as that of the (2+2) GA on ONEMAX. What we are more interested in is whether similar results also hold when optimizing other functions. We have the following simple theorem for this question.

**Theorem 5.** *Given any two pseudo-Boolean functions  $f_1$  and  $f_2$  that the Hamming distance between their optimal solutions in the unified search space is 0. Let  $E(T_1)$  and  $E(T_2)$  be the upper bound of expected runtime of Algorithm 1 optimizing  $f_1$  and  $f_2$ , respectively. Then we have  $E(T_2) \leq E(T_1) + O(n)$  if parameter  $rpm$  is set to a constant in  $(0, 1)$  and  $E(T_1) \leq E(T_2)$ .*

*Proof.* We let task 1 and task 2 denote functions  $f_1$  and  $f_2$ , respectively. Let  $x^*$  be the optimal solution in the unified search space for  $f_1$ . By the assumption, we have that  $x^*$  is created for  $f_1$  by Algorithm 1 in  $E(T_1)$  expected fitness evaluations and is kept in  $P_{t,1}$ . Similar to the proof of Theorem 4, we consider these solutions created by crossover operator in the case that  $\tau_a \neq \tau_b$  and  $rand < rpm$  (with probability  $\frac{2rpm}{3}$ ). Observe that if the individual  $x^*$  in  $P_{t,1}$  is selected to be a parent (with probability at least  $\frac{1}{2}$ ) and the crossover point is the  $n$ -th bit (with probability  $\frac{1}{n}$ ), an offspring encoded as  $x^*$  will be created and evaluate on  $f_2$ . The above event happens with probability at least  $\frac{2rpm}{3} \cdot \frac{1}{2} \cdot \frac{1}{n} = \frac{rpm}{3n}$  and the expected waiting time is  $\frac{(6+4rpm)}{3} \cdot \frac{3n}{rpm} = \frac{(6+4rpm)n}{rpm}$ . Therefore, we have  $E(T_2) \leq E(T_1) + O(n)$  if parameter  $rpm$  is set to a constant in  $(0, 1)$ .  $\square$

This theorem implies that how to map the solutions of distinct tasks into a new search space such that their optima are located in the same position in the evolutionary process is significant for MFEA. We noted that a decision variable translation strategy has been designed to handle this topic in [8]. Their experiments have showed the effectiveness of the proposed strategy. This analysis can enhance the effectiveness of the proposed strategy from theoretical aspect.

## 4 Conclusion

This paper investigates the superiority of MFEAs over traditional methods of solving each task independently from theoretical analysis aspect. We present runtime analysis of a baseline (4+2) MFEA and (2+2) GA on several benchmark pseudo-Boolean functions, which include problems with similar tasks and dissimilar tasks. The results show that by properly setting the parameter  $rpm$ , for the group of problems with similar tasks, the upper bound of expected runtime of the (4+2) MFEA on the harder task can be improved to be the same as on the easier one, while the expected upper bound on the group of problems with dissimilar tasks are the same as that of solving them independently by the (2+2) GA. This paper theoretically explains why some existing MFEAs work better than traditional methods of solving each task independently in numerical experiments and provides some insights into the working principles of MFEAs.



## References

1. Back, T., Hammel, U., Schwefel, H.P.: Evolutionary computation: comments on the history and current state. *IEEE Trans. Evol. Comput.* **1**(1), 3–17 (1997)
2. Bali, K.K., Ong, Y.S., Gupta, A., Tan, P.S.: Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II. *IEEE Trans. Evol. Comput.* **24**(1), 69–83 (2020)
3. Bao, L., et al.: An evolutionary multitasking algorithm for cloud computing service composition. In: Yang, A., et al. (eds.) *SERVICES 2018*. LNCS, vol. 10975, pp. 130–144. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-94472-2\\_10](https://doi.org/10.1007/978-3-319-94472-2_10)
4. Chandra, R., Gupta, A., Ong, Y.-S., Goh, C.-K.: Evolutionary multi-task learning for modular training of feedforward neural networks. In: Hirose, A., Ozawa, S., Doya, K., Ikeda, K., Lee, M., Liu, D. (eds.) *ICONIP 2016*. LNCS, vol. 9948, pp. 37–46. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46672-9\\_5](https://doi.org/10.1007/978-3-319-46672-9_5)
5. Chandra, R., Gupta, A., Ong, Y.S., Goh, C.K.: Evolutionary multi-task learning for modular knowledge representation in neural networks. *Neural Process. Lett.* **47**(3), 993–1009 (2018). <https://doi.org/10.1007/s11063-017-9718-z>
6. Cloninger, C.R., Rice, J., Reich, T.: Multifactorial inheritance with cultural transmission and assortative mating. ii. a general model of combined polygenic and cultural inheritance. *Am. J. Hum. Genet.* **31**(2), 176 (1979)
7. Dang, D.C., et al.: Escaping local optima using crossover with emergent diversity. *IEEE Trans. Evol. Comput.* **22**(3), 484–497 (2018)
8. Ding, J., Yang, C., Jin, Y., Chai, T.: Generalized multitasking for evolutionary optimization of expensive problems. *IEEE Trans. Evol. Comput.* **23**(1), 44–58 (2019)
9. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. *Theor. Comput. Sci.* **567**, 87–104 (2015)
10. Feldman, M.W., Laland, K.N.: Gene-culture coevolutionary theory. *Trends Ecol. Evol.* **11**(11), 453–457 (1996)
11. Feng, L., et al.: Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking. *IEEE Trans. Cybern.* (2020, in press)
12. Feng, L., et al.: Evolutionary multitasking via explicit autoencoding. *IEEE Trans. Cybern.* **49**(9), 3457–3470 (2018)
13. Gupta, A., Ong, Y.S., Feng, L.: Multifactorial evolution: toward evolutionary multitasking. *IEEE Trans. Evol. Comput.* **20**(3), 343–357 (2016)
14. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. *Artif. Intell.* **127**(1), 57–85 (2001)
15. Jansen, T.: *Analyzing evolutionary algorithms: the computer science perspective*. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-17339-4>
16. Kötzing, T., Krejca, M.S.: First-hitting times under additive drift. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) *PPSN 2018*. LNCS, vol. 11102, pp. 92–104. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_8](https://doi.org/10.1007/978-3-319-99259-4_8)
17. Li, G., Lin, Q., Gao, W.: Multifactorial optimization via explicit multipopulation evolutionary framework. *Inf. Sci.* **512**, 1555–1570 (2020)
18. Li, H., Ong, Y., Gong, M., Wang, Z.: Evolutionary multitasking sparse reconstruction: framework and case study. *IEEE Trans. Evol. Comput.* **23**(5), 733–747 (2019)
19. Liaw, R.T., Ting, C.K.: Evolutionary many tasking optimization based on symbiosis in biocoenosis. In: *The Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI, pp. 4295–4303 (2019)

20. Lin, J., Liu, H.L., Xue, B., Zhang, M., Gu, F.: Multi-objective multi-tasking optimization based on incremental learning. *IEEE Trans. Evol. Comput.* (2020, in press)
21. Neumann, F., Witt, C.: *Bioinspired Computation in Combinatorial Optimization - Algorithms and Their Computational Complexity*. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-16544-3>
22. Qian, C., Yu, Y., Tang, K., Yao, X., Zhou, Z.H.: Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artif. Intell.* **275**, 279–294 (2019)
23. Tang, J., Chen, Y., Deng, Z., Xiang, Y., Joy, C.P.: A group-based approach to improve multifactorial evolutionary algorithm. In: *International Joint Conference on Artificial Intelligence, IJCAI*, pp. 3870–3876 (2018)
24. Zhou, L., Feng, L., Zhong, J., Ong, Y.S., Zhu, Z., Sha, E.: Evolutionary multi-tasking in combinatorial search spaces: a case study in capacitated vehicle routing problem. In: *IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8. IEEE (2016)



# Improved Fixed-Budget Results via Drift Analysis

Timo Kötzing<sup>1</sup> and Carsten Witt<sup>2</sup>

<sup>1</sup> Hasso Plattner Institute, Potsdam, Germany  
timo.koetzing@hpi.de

<sup>2</sup> Technical University of Denmark, Kgs. Lyngby, Denmark  
cawi@dtu.dk

**Abstract.** Fixed-budget theory is concerned with computing or bounding the fitness value achievable by randomized search heuristics within a given budget of fitness function evaluations. Despite recent progress in fixed-budget theory, there is a lack of general tools to derive such results. We transfer drift theory, the key tool to derive expected optimization times, to the fixed-budget perspective. A first and easy-to-use statement concerned with iterating drift in so-called greed-admitting scenarios immediately translates into bounds on the expected function value. Afterwards, we consider a more general tool based on the well-known variable drift theorem. Applications of this technique to the LEADINGONES benchmark function yield statements that are more precise than the previous state of the art.

## 1 Introduction

Randomized search heuristics are a class of optimization algorithms which use probabilistic choices with the aim of maximizing or minimizing a given objective function. Typical examples of such algorithms use inspiration from nature in order to determine the method of search, most prominently evolutionary algorithms, which use the concepts of *mutation* (slightly altering a solution) and *selection* (giving preference to solutions with better objective value).

The theory of randomized search heuristics aims at understanding such heuristics by explaining their optimization behavior. Recent results are typically phrased as run time results, for example by giving upper (and lower) bounds on the expected time until a solution of a certain quality (typically the best possible quality) is found. This is called the (expected) *optimization time*. A different approach, called *fixed-budget analysis*, bounds the quality of the current solution of the heuristic after a given amount of time. In order to ease the analysis and by convention, in this theoretical framework *time* is approximated as the number of evaluations of the objective function (called fitness evaluations).

In this paper we are concerned with the approach of giving a fixed-budget analysis. This approach was introduced to the analysis of randomized search heuristics by Jansen and Zarges [7], who derived fixed-budget results for the classical example functions ONEMAX and LEADINGONES by bounding the expected



progress in each iteration. A different perspective was proposed by Doerr, Jansen, Witt and Zarges [1], who showed that fixed-budget statements can be derived from bounds on optimization times if these exhibit strong concentration. Lengler and Spooner [13] proposed a variant of multiplicative drift for fixed-budget results and the use of differential equations in the context of ONEMAX and general linear functions. Nallaperuma, Neumann and Sudholt [15] applied fixed-budget theory to the analysis of evolutionary algorithms on the traveling salesman problem and Jansen and Zarges [8] to artificial immune systems. The quality gains of optimal black-box algorithms on ONEMAX in a fixed-budget perspective were analyzed by Doerr, Doerr and Yang [2]. In a recent technical report, He, Jansen and Zarges [5] consider so-called unlimited budgets to estimate fitness values in particular for points of time larger than the expected optimization time. A recent survey by Jansen [6] summarizes the state of the art in the area of fixed-budget analysis.

There are general methods easing the analysis of randomized search heuristics. Most importantly, in order to derive bounds on the optimization time, we can make use of drift theory. Drift theory is a general term for a collection of theorems that consider random processes and bound the expected time it takes the process to reach a certain value—the first-hitting time. The beauty and appeal of these theorems lie in them usually having few restrictions but yielding strong results. Intuitively speaking, in order to use a drift theorem, one only needs to estimate the expected change of a random process—the drift—at any given point in time. Hence, a drift theorem turns expected local changes of a process into expected first-hitting times. In other words, local information of the process is transformed into global information. See [12] for an extensive discussion of drift theory.

In contrast to the numerous drift theorems available for bounding the optimization time, there is no corresponding theorem for making a fixed-budget analysis apart from one for the multiplicative case given in [13]. With this paper we aim to provide several such drift theorems, applicable in different settings and with a different angle of conclusions. In each our main goal is to provide an *upper bound* on the distance to the optimum after  $t$  iterations, for  $t$  less than the expected optimization time. Upper bounds alone do not allow for a fair comparison of algorithms, since a bad upper bound does not exclude the possibility of a good performance of an algorithm; for this, we require lower bounds. However, one of our techniques also allows us to derive lower bounds. Furthermore, when upper and lower bounds are close together we can conclude that the derived bounds are correspondingly tight, highlighting the quality of our methods.

We start, in Sect. 3, by giving a theorem which iteratively applies local drift estimates to derive a global drift estimate after  $t$  iterations. Crucial for this theorem is that the drift condition is *unlimited time*, by which we mean that the drift condition has to hold for all times  $t$ , not just (which is the typical case in the literature for drift theorems) those before the optimum is hit. This theorem is applicable in the case where there is no optimum (and optimization progresses indefinitely) and in the case that, in the optimum, the drift is 0.

In order to bypass these limitations we also give a variant in Sect. 3 which allows for *limited time* drift, where the drift condition only needs to hold before the optimum is hit; however, in this case we pick up an additional error term in the result, derived from the possibility of hitting the optimum within the allowed time budget of  $t$ . Thus, in order to apply this theorem, one will typically need concentrations bounds for the time to hit the optimum.

For both these theorems, the drift function (bounding the drift) has to be convex and *greed-admitting*, which intuitively says that being closer to the goal is always better in terms of the expected state after an additional iteration, while search points closer to the goal are required to have weaker drift. These conditions are fulfilled in many sample applications; as examples we give analyses of the (1+1) EA on LEADINGONES and ONEMAX. Note that these analyses seem to be rather tight, but we do not offer any lower bounds, since our techniques crucially only apply in one direction (owing to an application of Jensen's Inequality to convex drift functions).

In Sect. 4 we use a potential-based approach and give a variable drift theorem for fixed-budget analysis. As a special case, where the drift function is constant, we give an additive drift theorem for fixed-budget analysis and derive a result for (1+1) EA on LEADINGONES. In general, the approach bounds the expected value of the potential but not of the fitness. Therefore, we also study how to derive a bound on the fitness itself, both from above and from below, by inverting the potential function and using tail bounds on its value. The approach uses a generalized theorem showing tail bounds for martingale differences, which overcomes a weakness of existing martingale difference theorems in our specific application. This generalization may be of independent interest.

Our results allow for giving strong fixed-budget results which were not obtainable before. For the (1+1) EA on LEADINGONES with a budget of  $t = o(n^2)$  iterations, the original paper [7] gives a lower bound of  $2t/n - o(t/n)$  for the expected fitness after  $t$  iterations, which we recover with a simple proof in Theorem 8. Our theorem also allows budgets closer to the expected optimization time, where we get a lower bound of  $n \ln(1 + 2t/n^2) - O(1)$ .

For the (1+1) EA on ONEMAX, no concrete formula for a bound on the fitness value after  $t$  iterations was known: The original work [7] could only handle RLS on ONEMAX, not the (1+1) EA. The multiplicative drift theorem of [13] allows for deriving a lower bound of  $n/2 + t/(2e)$  for  $t = o(n)$  using a multiplicative drift constant of  $(1 - 1/n)^n/n$ . Since our drift theorem allows for variable drift, we can give a bound of  $n/2 + t/(2\sqrt{e}) - o(t)$  for the (1+1) EA on ONEMAX with  $t = o(n)$  (see Theorem 7). Note that [13] also gives bounds for values of  $t$  closer to the expected optimization time.

Furthermore, we are not only concerned with expected values but also give strong concentration bounds. We consider the (1+1) EA on LEADINGONES and show that the fitness after  $t$  steps is strongly concentrated around its expectation (see Theorem 12). The error term obtained is asymptotically smaller than in the previous work [1] and the statement is also less complex.

Fixed-budget results that hold with high probability are crucial for the analysis of algorithm configurators [4]. These configurators test different algorithms for fixed budgets in order to make statements about their appropriateness in a given setting. Thus, we believe that this work also contributes to the better understanding of the strengths and weaknesses of algorithm configurators.

The remainder of the paper is structured as follows. Next we give mathematical preliminaries, covering problem and algorithm definitions as well as some well-known results from the literature which we require later. In Sect. 3 we give our direct fixed-budget drift theorems, as well as its applications to the (1+1) EA on ONEMAX and LEADINGONES. In Sect. 4 we give a variable fixed-budget drift theorem and its corollary for additive drift. We show how to apply this variable fixed-budget drift theorem to obtain very strong bounds in Sect. 5. We conclude in Sect. 6. Due to space limitations, all proofs have been removed from this article. A full technical report is available at [11].

## 2 Preliminaries

The concrete objective functions we are concerned with in this paper are ONEMAX and LEADINGONES, studied in a large number of papers. These two functions are defined as follows. For a fixed natural number  $n$ , the functions map bit strings  $x \in \{0, 1\}^n$  of length  $n$  to natural numbers such that

$$\text{ONEMAX}(x) = \sum_{i=1}^n x_i$$

is the number of 1s in the bit string  $x$  and

$$\text{LEADINGONES}(x) = \sum_{i=1}^n \prod_{j=1}^i x_j$$

is the number of leading 1s in  $x$  before the first 0 (if any,  $n$  otherwise).

We consider for application only one algorithm, the well-known (1+1) EA given in Algorithm 1 below.

For any function  $f$  and  $i \geq 0$ , we let  $f^i$  denote the  $i$ -times self-composition of  $f$  (with  $f^0$  being the identity).

---

**Algorithm 1:** The (1+1) EA for maximizing function  $f$

---

```

1 choose  $x$  from  $\{0, 1\}^n$  uniformly at random;
2 while optimum not reached do
3    $y \leftarrow x$ ;
4   for  $i = 1$  to  $n$  do
5      $\lfloor$  with probability  $1/n$ :  $y_i \leftarrow 1 - y_i$ ;
6   if  $f(y) \geq f(x)$  then  $x \leftarrow y$ ;
```

---

### 2.1 Known Results for the (1+1) EA on LeadingOnes

We will use the following concentration result from [1], bounding the optimization time of the (1+1) EA on LEADINGONES.

**Theorem 1** ([1, Theorem 7]). *For all  $d \leq 2n^2$ , the probability that the optimization time of the (1+1) EA on LEADINGONES deviates from its expectation of  $(1/2)(n^2 - n)((1 + 1/(n-1))^n - 1)$  by at least  $d$ , is at most  $4 \exp(-d^2/(20e^2n^3))$ .*

The following lemma collects some important and well-known results for the optimization process of the (1+1) EA on LEADINGONES.

**Lemma 2.** *Consider the (1+1) EA on LEADINGONES, let  $x_t$  denote its search point at time  $t$  and  $X_t = n - \text{LEADINGONES}(x_t)$  the fitness distance. Then*

1.  $E(X_t - X_{t+1} \mid X_t) = (2 - 2^{1-X_t})(1 - 1/n)^{n-X_t}/n$
2.  $\Pr(X_{t+1} \neq X_t \mid X_t; T > t) = (1 - 1/n)^{n-X_t} \frac{1}{n}$
3. For  $j \geq 1$ ,  $\Pr(X_{t+1} = X_t - j) \leq \frac{1}{n} \left(\frac{1}{2}\right)^{j-1}$
4.  $G_t := X_t - X_{t+1}$  is a random variable with support  $0, \dots, X_t$  and the following conditional distribution on  $G_t \geq 1$ :
  - $\Pr(G_t = i) = (1/2)^i$  for  $i < X_t$
  - $\Pr(G_t = X_t) = (1/2)^{X_t-1}$

For the moment-generating function of this  $G_t$  (conditional on  $G_t \geq 1$ ) it holds that

$$E(e^{\eta G_t} \mid X_t) = \frac{(e^\eta/2)^{X_t}(1 - e^\eta) + (e^\eta/2)}{1 - e^\eta/2}.$$

5. The expected optimization time equals  $\frac{n^2-n}{2} \left( \left(1 + \frac{1}{n-1}\right)^n - 1 \right)$ , which is  $\frac{e-1}{2}n^2 \pm O(n)$ .

### 3 Direct Fixed-Budget Drift Theorems

In this section we give a drift theorem which gives a fixed-budget result without the detour via first hitting times. The idea is to focus on drift which gets monotonically weaker as we approach the optimum, but where being closer to the optimum is still better in terms of drift. To this end, we make the following definition.

**Definition 3.** *We say that a drift function  $h: S \rightarrow \mathbb{R}^{>0}$  is greed-admitting if  $\text{id} - h$  (the function  $x \mapsto x - h(x)$ ) is monotone non-decreasing.*

Intuitively, this formalizes the idea that being closer to the goal is always better (i.e. greed is good). Greed could be bad, if from one part of the search space, the drift is much higher than when being a bit closer, so that being a bit closer does not balance out the loss in drift. Note that any given differentiable  $h$  is greed-admitting if and only if  $h' \leq 1$ .

Typical drift functions are greed-admitting. For example, if we drift on integers, in many situations drift is less than 1, while being closer means being at

least one step closer, so being closer is always better in this sense. An example monotone process on  $\{0, 1, 2\}$  which has a drift which is not greed-admitting is the following:  $X_0$  is 2 and the process moves to any of the states 0, 1, 2 uniformly. State 0 is the target state, from state 1 there is only a very small probability to progress to 0 (say 0.1). Then it is better to stay in state 2 than be trapped in state 1, if the goal is to progress to state 0.

We now give two different versions of the direct fixed-budget drift theorem. The first considers *unlimited time*, that is, the situation where drift carries on for an arbitrary time (and does not stop once a certain threshold value is reached). This is applicable in situations where there is no end to the process (for example for random walks on the line) or when the drift eventually goes all the way down to 0 so that the drift condition holds vacuously even when no progress is possibly any more (this is for example the case for multiplicative drift, where the drift is  $\delta$  times the current value, which is naturally 0 once 0 has been reached). Note that this is a very strong requirement of the theorem, leading to a strong conclusion.

A special case of the following theorem is given in [13], where drift is necessarily multiplicative.

**Theorem 4 (Direct Fixed-Budget Drift, unlimited time).** *Let  $X_t, t \geq 0$ , be a stochastic process on  $S \subseteq \mathbb{R}$ , adapted to a filtration  $\mathcal{F}_t$ . Let  $h: S \rightarrow \mathbb{R}^{\geq 0}$  be a **convex and greed-admitting** function such that we have the drift condition*

$$\text{(D-ut)} \quad \mathbb{E}(X_t - X_{t+1} \mid \mathcal{F}_t) \geq h(X_t).$$

Define  $\tilde{h}(x) = x - h(x)$ . Thus, the drift condition is equivalent to

$$\text{(D-ut')} \quad \mathbb{E}(X_{t+1} \mid \mathcal{F}_t) \leq \tilde{h}(X_t).$$

We have that, for all  $t \geq 0$ ,<sup>1</sup>

$$\mathbb{E}(X_t \mid \mathcal{F}_0) \leq \tilde{h}^t(X_0)$$

and, in particular,

$$\mathbb{E}(X_t) \leq \tilde{h}^t(\mathbb{E}(X_0)).$$

Now we get to the second version of the theorem, considering the more frequent case where no guarantee on the drift can be given once the optimum has been found. This weaker requirement leads to a weaker conclusion.

**Theorem 5 (Direct Fixed-Budget Drift, limited time).** *Let  $X_t, t \geq 0$ , be a stochastic process on  $S \subseteq \mathbb{R}$ ,  $0 = \min S$ , adapted to a filtration  $\mathcal{F}_t$ . Let  $T := \min\{t \geq 0 \mid X_t = 0\}$  and  $h: S \rightarrow \mathbb{R}^{\geq 0}$  be a **differentiable, convex and greed-admitting** function such that  $\tilde{h}'(0) \in ]0, 1]$  and we have the drift condition*

$$\text{(D-lt)} \quad \mathbb{E}(X_t - X_{t+1} \mid \mathcal{F}_t; t < T) \geq h(X_t).$$

Define  $\tilde{h}(x) = x - h(x)$ . Thus, the drift condition is equivalent to

---

<sup>1</sup> Recall from the preliminaries that  $f^i$  is the  $i$ -times self-composition of a function  $f$ .

(D-It')  $E(X_{t+1} \mid \mathcal{F}_t; t < T) \leq \tilde{h}(X_t).$

We have that, for all  $t \geq 0,$

$$E(X_t \mid \mathcal{F}_0) \leq \tilde{h}^t(X_0) + \frac{\tilde{h}(0)}{\tilde{h}'(0)}$$

and, in particular,

$$E(X_t) \leq \tilde{h}^t(E(X_0)) - \frac{\tilde{h}(0)}{\tilde{h}'(0)} \cdot \Pr(t \geq T \mid \mathcal{F}_0).$$

With the following theorem we give a general way of iterating a greed-admitting function, as necessary for the application of the previous two theorems. From this we can see the similarity of this approach to the method of variable drift theory where the inverse of  $h$  is integrated over, see Theorem 9 and the discussion about drift theory in general in [12].

**Theorem 6.** *Let  $h$  be greed-admitting and let  $\tilde{h} = \text{id} - h$ . Then we have, for all starting points  $n$  and all target points  $m < n$  and all time budgets  $t,$*

$$\text{if } t \geq \sum_{i=m}^{n-1} \frac{1}{h(i)} \text{ then } \tilde{h}^t(n) \leq m.$$

### 3.1 Application to OneMax

In this section we show how we can apply Theorem 4 by using the optimization of the (1+1) EA on ONEMAX as an example (where we have multiplicative drift).

**Theorem 7.** *Let  $V_t$  be the number of 1s which the (1+1) EA on ONEMAX has found after  $t$  iterations of the algorithm. Then we have, for all  $t,$*

$$E(V_t) \geq \begin{cases} \frac{n}{2} + \frac{t}{2\sqrt{e}} - O(1), & \text{if } t = O(\sqrt{n}); \\ \frac{n}{2} + \frac{t}{2\sqrt{e}}(1 - o(1)), & \text{if } t = o(n). \end{cases}$$

Furthermore, for all  $t,$  we have  $E(V_t) \geq n(1 - \exp(-t/(en))/2).$

### 3.2 Application to LeadingOnes

In this section we want to use Theorem 5 to the progress of the (1+1) EA on LEADINGONES. The result is summarized in the following theorem.

**Theorem 8.** *Let  $V_t$  be the number of leading 1s which the (1+1) EA on LEADINGONES has found after  $t$  iterations of the algorithm. We have, for all  $t,$*

$$E(V_t) \geq \begin{cases} \frac{2t}{n} - O(1), & \text{if } t = O(n^{3/2}); \\ \frac{2t}{n} \cdot (1 - o(1)), & \text{if } t = o(n^2); \\ n \ln(1 + \frac{2t}{n^2}) - O(1), & \text{if } t \leq \frac{e-1}{2}n^2 - n^{3/2}. \end{cases}$$

### 4 Variable Drift Theorem for Fixed Budget

We now turn to an alternative approach to derive fixed-budget results via drift analysis. Our method is based on variable drift analysis that was introduced to the analysis of randomized search heuristics by Johannsen [9]. Crucially, variable drift analysis applies a specific transformation, the so-called potential function  $g$ , to the state space. Along with bounds on the hitting times, we obtain the following theorem estimating the expected value of the potential function after  $t$  steps. Subsequently, we will discuss how this information can be used to analyze the untransformed state.

**Theorem 9.** *Let  $X_t, t \geq 0$ , be a stochastic process, adapted to a filtration  $\mathcal{F}_t$ , on  $S := \{0\} \cup \mathbb{R}^{\geq x_{\min}}$  for some  $x_{\min} > 0$ . Let  $T := \min\{t \geq 0 \mid X_t = 0\}$  and  $h: S \rightarrow \mathbb{R}^{>0}$  be a non-decreasing function such that  $\mathbb{E}(X_t - X_{t+1} \mid \mathcal{F}_t; t < T) \geq h(X_t)$ . Define  $g: S \rightarrow \mathbb{R}$  by*

$$g(x) := \begin{cases} \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^x \frac{1}{h(z)} dz & \text{if } x \geq x_{\min} \\ 0 & \text{otherwise} \end{cases}.$$

Then it holds that

$$\mathbb{E}(g(X_t) \mid \mathcal{F}_0) \leq g(X_0) - \sum_{s=0}^{t-1} \Pr(s < T).$$

#### 4.1 Additive Drift as Special Case

A special case of variable drift is additive drift, when the drift function  $h$  is constant.

**Theorem 10.** *Let  $X_t, t \geq 0$ , be a stochastic process, adapted to a filtration  $\mathcal{F}_t$ , on  $S := \mathbb{R}^{\geq 0}$ . Let  $T := \min\{t \geq 0 \mid X_t = 0\}$  and  $\delta \in \mathbb{R}^{>0}$  be such that  $\mathbb{E}(X_t - X_{t+1} \mid \mathcal{F}_t; t < T) \geq \delta$ . Then we have*

$$\mathbb{E}(X_t \mid \mathcal{F}_0) \leq X_0 - \delta \sum_{s=0}^{t-1} \Pr(s < T).$$

The theorem is a corollary to Theorem 9 by using  $x_{\min} = \delta$ , the smallest value for which the condition of a drift of at least  $\delta$  can still be obtained, and thus the smallest value (other than 0) that the process can attain.

As a sample application, we can now derive an estimate of the best value found by the (1+1) EA on LEADINGONES within  $t$  steps, using the concentration result from [1] given in Theorem 1.

**Theorem 11.** *Let  $V_t$  be the number of leading 1s which the (1+1) EA on LEADINGONES has found after  $t$  iterations of the algorithm. Then, for all  $t \leq \frac{e-1}{2}n^2 - n^{3/2} \log(n)$ , we have*

$$\mathbb{E}(V_t) \geq \frac{2t}{en} - O(1).$$

Note that the result was proven very easily with a direct application of the additive version of the fixed-budget drift theorem in combination with a strong result on concentration. The price paid for this simplicity is that the lead constant in this time bound is not tight, as can be seen by comparing with the results given in Theorem 8.

### 5 Variable Drift and Concentration Inequalities

The expected  $g(X_t)$ -value derived in Theorem 9 is not very useful unless it allows us to make conclusions on the underlying  $X_t$ -value. The previous application in Sect. 4.1 only gives tight bounds in case that the drift is more or less constant throughout the search space. This is not the case for ONEMAX and LEADINGONES where the drift increases with the distance to the optimum (e. g., for ONEMAX the drift is  $\Theta(1/n)$  at distance 1 and  $\Theta(1)$  as distance  $n/2$ ; for LEADINGONES the drift can vary by a term of roughly  $e$ ). Hence, looking back into Theorem 9, we now are interested in characterizing  $g(X_t)$  more precisely than just in terms of expected value. If we manage to establish concentration of  $g(X_t)$  then we can (after inverting  $g$ ) derive a maximum of the  $X_t$ -value that holds with sufficient probability. Our main result achieved along this path is the following one.

**Theorem 12.** *Let  $V_t$  be the number of leading 1s which the (1+1) EA on LEADINGONES has found after  $t$  iterations. Then for  $t = \omega(n \log n)$  and  $t \leq (e - 1)n^2/2 - cn^{3/2}\sqrt{\log n}$ , where  $c$  is a sufficiently large constant the following statements hold. (a) With probability at least  $1 - 1/n^3$ ,*

$$\begin{aligned}
 -n \ln\left(1 - 2t/n^2 + O(\sqrt{t \log n}/n^{3/2})\right) &\leq V_t \\
 -n \ln\left(1 - 2t/n^2 - O(\sqrt{t \log n}/n^{3/2})\right) &\geq V_t.
 \end{aligned}$$

(b)  $E(V_t) = -n \ln(1 - 2t/n^2 + O(\sqrt{t \log n}/n^{3/2}))$ .

To compare with previous work, we note that the additive error turns out as  $O(\sqrt{t \log n}/n^{1/2})$ . This is asymptotically smaller than the additive error term of order  $\Omega(n^{3/2+\epsilon})$  that appears in the fixed-budget statements of [1] and moreover, it depends on  $t$ . Also, we think that the formulation of our statement is less complex than in that paper.

The proof of Theorem 12 overcomes several technical challenges. The first idea is to apply established concentration inequalities for stochastic processes. Since (after a reformulation discussed below) the process of  $g$ -values describes a (super)martingale, it is natural to take the method of bounded martingale differences. However, since there is no ready-to-use theorem for all our specific martingales, we present a generalization of martingale concentration inequalities in the following subsection Sect. 5.1. The concrete application is then given in Sects. 5.2 onwards.



### 5.1 Tail Bounds for Martingale Differences

The classical method of bounded martingale differences [14] considers a (super)martingale  $Y_t$ ,  $t \geq 0$ , and its corresponding martingale differences  $D_t = Y_{t+1} - Y_t$ . Given certain boundedness conditions for  $D_t$  (e. g., that  $|D_t| \leq c$  for a constant  $c$  almost surely), it is shown that the sum of martingale differences  $\sum_{i=0}^{t-1} D_i = Y_t - Y_0$  does not deviate much from its expectation  $Y_0$  (resp. is not much bigger in the case of supermartingales). This statement remains essentially true if  $D_t$  is allowed to have unbounded support but exhibits a strong concentration around its expected value. Usually, this concentration is formulated in terms of a so-called *subgaussian* (or, similarly, *subexponential*) property [3, 10]. Roughly speaking, this property requires that the moment-generating function (mgf.) of the differences can be bounded as  $E(e^{\lambda D_t} \mid \mathcal{F}_t) \leq e^{\lambda^2 \nu_t^2 / 2}$  for a certain parameter  $\nu_t$  and all  $\lambda < 1/b_t$ , where  $b_t$  is another parameter. In particular, the bound has to remain true when  $\lambda$  becomes arbitrarily small.

In one of our concrete applications of the martingale difference technique, the inequality  $E(e^{\lambda D_t} \mid \mathcal{F}_t) \leq e^{\lambda^2 \nu_t^2 / 2}$  is true for certain values of  $\lambda$  below a threshold  $1/b^*$ , but does not hold if  $\lambda$  is much smaller than  $1/b^*$ . We therefore show that the concentration of the sums of martingale differences to some extent remains true if the inequality only holds for  $\lambda \in [1/a^*, 1/b^*]$  where  $a^* > b^*$  is another parameter. The approach uses well-known arguments for the proof of concentration inequalities. Here, we were inspired by the notes [16], which require the classical subexponential property, though.

**Theorem 13.** *Let  $Y_t$ ,  $t \geq 0$ , be a supermartingale, adapted to a filtration  $\mathcal{F}_t$ , and let  $D_t = Y_{t+1} - Y_t$  be the corresponding martingale differences. Assume that there are  $0 < b_2 < b_1 \leq \infty$  and a sequence  $\nu_t$ ,  $t \geq 0$ , such that for  $\lambda \in [1/b_1, 1/b_2]$  it holds that  $E(e^{\lambda D_t} \mid \mathcal{F}_t) \leq e^{\lambda^2 \nu_t^2 / 2}$ . Then for all  $t \geq 0$  it holds that*

$$\Pr(Y_t - Y_0 \geq d) \leq \begin{cases} e^{-d/(2b_2)} & \text{if } d \geq \frac{\sum_{i=0}^{t-1} \nu_i^2}{b_2} \\ e^{-d^2/(2\sum_{i=0}^{t-1} \nu_i^2)} & \text{if } \frac{\sum_{i=0}^{t-1} \nu_i^2}{b_1} \leq d < \frac{\sum_{i=0}^{t-1} \nu_i^2}{b_2} \end{cases}$$

The theorem holds analogously for submartingales with respect to the tail bound  $\Pr(Y_t - Y_0 \leq -d)$ .

### 5.2 Preparing an Upper Tail Bound via the Martingale Difference Method

We now return to Theorem 9 and would like to show concentration of  $g(X_t)$  in order to show a bound for  $X_t$  that holds with sufficiently high probability. Note that by the statement of the theorem, we immediately have that  $Y_t := g(X_t) + \sum_{s=0}^{t-1} \Pr(T > s)$  is a supermartingale. By bounding the probability of  $Y_t \geq d$  for arbitrary  $t \geq 0$  and  $d \geq 0$ , i. e., establishing concentration of the supermartingale  $Y_t$  via Theorem 13, and inverting  $g$ , we will obtain a bound on the probability of the event  $g(X_t) \geq E(g(X_t))$ .

As we want to prove Theorem 12, the application is again the (1+1) EA on the LEADINGONES function, so  $X_t = n - \text{LEADINGONES}(x_t)$  is the fitness distance of the LEADINGONES-value at time  $t$  from the target.

Defining  $h(X_t) := E(X_t - X_{t+1} \mid X_t)$  according to Lemma 2 and  $g(X_t) = 1/h(1) + \int_1^{X_t} 1/h(z) dz$  according to Lemma 9, we will establish the following bound on the moment-generating function (mgf.) of the drift of our concrete  $g$ .

**Lemma 14.** *Let  $T$  denote the optimization time of the (1+1) EA on LEADINGONES. If  $\lambda \leq 1/(2en)$  then  $E(e^{\lambda(g(X_{t+1})-g(X_t)+\Pr(T>t))} \mid X_t) = e^{O(\lambda^2 n)}$ .*

Looking into Theorem 13 the required subexponential property of the martingale difference  $D_t$  has been proven with  $\nu = O(\sqrt{n})$  and  $\lambda \leq 1/(2en) = 1/b^*$ . Before we formally apply this lemma, we also establish concentration in the other direction.

### 5.3 Preparing a Lower Tail Bound

We will now complement the upper tail bound for  $g$  that we prepared in the previous subsection with a lower tail bound. The aim is again to apply Theorem 13, this time with respect to the sequence  $Y_t = g(X_t) + \sum_{s=0}^{t-1} \Pr(T > s) + r(t, n)$ , where  $X_t = n - \text{LEADINGONES}(x_t)$  is still the fitness distance of the LEADINGONES-value at time  $t$  from the target and  $r(t, n)$  is an “error term” that we will prove to be  $O(1/n)$  if  $g(X_t) > \log n$ . Moreover,  $r(t, n) = 0$  if  $g(X_t) = 0$ . The first step is to prove that  $Y_t$  is a submartingale, i. e.,  $E(Y_{t+1} \mid Y_t) \geq Y_t$ . Afterwards, we bound the mgf. of  $D_t = Y_t - Y_{t-1} = g(X_{t+1}) - g(X_t) + \Pr(T > t) + r(t, n)$ .

**Lemma 15.** *The sequence  $Y_t = g(X_t) + \sum_{s=0}^{t-1} \Pr(T > s) + r(t, n)$  is a submartingale with  $r(t, n) = O(1/n)$  for  $X_t > \log n$ .*

Recall that the aim is to apply Theorem 13 with respect to the submartingale sequence  $Y_t = g(X_t) + \sum_{s=0}^{t-1} \Pr(T > s) + r(t, n)$ . To this end, we shall bound the mgf. of  $D_t = Y_t - Y_{t-1} = g(X_{t+1}) - g(X_t) + \Pr(T > t) + r(t, n)$  in the following way.

**Lemma 16.** *The mgf. of  $D_t = Y_t - Y_{t-1} = g(X_{t+1}) - g(X_t) + \Pr(T > t) + r(t, n)$  satisfies  $E(e^{\lambda D_t} \mid X_t) = e^{O(\lambda^2 n)}$  for all  $\lambda \in [1/n^2, 1/(2en)]$ .*

Hence, we can satisfy the assumptions of Theorem 13 with  $b_2 = 2en$  and  $b_1 = n^2$ . We will apply this theorem in the following subsection, where we put everything together.

### 5.4 Main Concentration Result – Putting Everything Together

In the previous subsections we have derived (w. r. t. LEADINGONES) that the sequence  $\Delta_t^{(\ell)} = g(X_t) - g(X_{t+1}) + \sum_{s=0}^{t-1} \Pr(T > s)$  is a supermartingale and the sequence  $\Delta_t^{(h)} = g(X_t) - g(X_{t+1}) + \sum_{s=0}^{t-1} \Pr(T > s) + r(t, n)$ , where  $r(t, n) = O(1/n)$ , is a submartingale. We also know from Theorem 9 that  $E(g(X_t) \mid \mathcal{F}_0) \leq$

$g(X_0) - \sum_{s=0}^{T-1} \Pr(T > s)$ . Hence, using Theorem 13 with respect to the  $\Delta_t^{(\ell)}$ -sequence, choosing  $b_1 = \infty$  and  $b_2 = 2en$  according to our analysis of the mgf., we obtain (since  $\nu^2 = O(n)$ ) the first statement of the following theorem. Its second statement follows by applying Theorem 13 with respect to the  $\Delta_t^{(h)}$ -sequence, choosing  $b_2 = 2en$  and  $b_1 = n^2$ .

**Theorem 17.**

$$\Pr(g(X_t) \geq \mathbb{E}(g(X_t)) + d) \leq \begin{cases} e^{-d/(4en)}, & \text{if } d \geq Ct; \\ e^{-\Omega(d^2/(tn))}, & \text{otherwise,} \end{cases}$$

where  $C = \nu^2/(4en) = O(1)$ . Moreover,

$$\Pr(g(X_t) \leq \mathbb{E}(g(X_t)) - d - tr(t, n)) \leq \begin{cases} e^{-d/(4en)}, & \text{if } d \geq Ct; \\ e^{-\Omega(d^2/(tn))}, & \text{if } \frac{C't}{n} \leq d < Ct; \end{cases}$$

where  $C = \nu^2/(4en) = \Theta(1)$  and  $C' = \nu^2/n = \Theta(1)$ .

As mentioned above, Theorem 9 gives us an upper bound on  $\mathbb{E}(g(X_t))$  but we would like to know an upper bound on  $\mathbb{E}(X_t)$ . Unfortunately, since  $g$  is concave, it does **not** hold that  $\mathbb{E}(X_t) \leq g^{-1}(\mathbb{E}(g(X_t)))$ . However, using the concentration inequalities above, we can show that  $\mathbb{E}(X_t)$  is not much bigger than the right-hand side of this wrong estimate. Given  $t > 0$ , we choose a  $d^* > 0$  for the tail bound such that  $\Pr(g(X_t) > \mathbb{E}(g(X_t)) + d^*) \leq 1/n^3$ . If  $g(X_t) \leq \mathbb{E}(g(X_t)) + d^*$ , the concavity of  $g$  implies that the  $\mathbb{E}(X_t)$ -value is maximized if  $g(X_t)$  takes the value  $\mathbb{E}(g(X_t)) + d^*$  with probability  $\frac{\mathbb{E}(g(X_t))}{\mathbb{E}(g(X_t)) + d^*}$  and is 0 otherwise. Since  $g(X_t) = O(n^2)$ , we altogether have

$$\begin{aligned} \mathbb{E}(X_t) &\leq \frac{1}{n^3} O(n^2) + g^{-1}(\mathbb{E}(g(X_t)) + d^*) \frac{\mathbb{E}(g(X_t))}{\mathbb{E}(g(X_t)) + d^*} \\ &= g^{-1}(\mathbb{E}(g(X_t)) + d^*) \frac{\mathbb{E}(g(X_t))}{\mathbb{E}(g(X_t)) + d^*} + o(1). \end{aligned}$$

The omitted proof of Theorem 12 makes this idea concrete.

## 6 Conclusions

We have described two general approaches that derive fixed-budget results via drift analysis. The first approach is concerned with iterating drifts either in an unbounded time scenario, or, using bounds on hitting times, in the scenario that the underlying process stops at some target state. Applying this approach to the ONEMAX or LEADINGONES functions, we obtain strong lower bounds on the expected fitness value after a given number of iterations. The second approach is based on variable drift analysis and tail bounds for martingale differences. Exemplified for the LEADINGONES function, this technique allows us to derive statements that are more precise than the previous state of the art. We think that our drift theorems can be useful for future fixed-budget analyses.

## References

1. Doerr, B., Jansen, T., Witt, C., Zarges, C.: A method to derive fixed budget results from expected optimisation times. In: Proceedings of GECCO 2013, pp. 1581–1588. ACM Press (2013)
2. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. *Theor. Comput. Sci.* **801**, 1–34 (2020)
3. Fan, X., Grama, I., Liu, Q.: Exponential inequalities for martingales with applications. *Electron. J. Probab.* **20**, 22 (2015)
4. Hall, G.T., Oliveto, P.S., Sudholt, D.: On the impact of the cutoff time on the performance of algorithm configurators. In: Proceedings of GECCO 2019, pp. 907–915. ACM Press (2019)
5. He, J., Jansen, T., Zarges, C.: Unlimited budget analysis of randomised search heuristics. CoRR, abs/1909.03342 (2019). <http://arxiv.org/abs/1909.03342>
6. Jansen, T.: Analysing stochastic search heuristics operating on a fixed budget. *Theory of Evolutionary Computation*. NCS, pp. 249–270. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-29414-4\\_5](https://doi.org/10.1007/978-3-030-29414-4_5)
7. Jansen, T., Zarges, C.: Fixed budget computations: a different perspective on run time analysis. In: Proceedings of GECCO 2012, pp. 1325–1332. ACM Press (2012)
8. Jansen, T., Zarges, C.: Reevaluating immune-inspired hypermutations using the fixed budget perspective. *IEEE Trans. Evol. Comput.* **18**(5), 674–688 (2014)
9. Johannsen, D.: Random combinatorial structures and randomized search heuristics. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany and the Max-Planck-Institut für Informatik (2010)
10. Kötzing, T.: Concentration of first hitting times under additive drift. *Algorithmica* **75**, 490–506 (2016)
11. Kötzing, T., Witt, C.: Improved fixed-budget results via drift analysis (2020). <http://arxiv.org/abs/2006.07019>
12. Lengler, J.: Drift analysis. *Theory of Evolutionary Computation*. NCS, pp. 89–131. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-29414-4\\_2](https://doi.org/10.1007/978-3-030-29414-4_2)
13. Lengler, J., Spooner, N.: Fixed budget performance of the (1+1) EA on linear functions. In: Proceedings of FOGA 2015, pp. 52–61. ACM Press (2015)
14. McDiarmid, C.: Concentration. In: Habib, M., McDiarmid, C., Ramirez-Alfonsin, J., Reed, B. (eds.) *Probabilistic Methods for Algorithmic Discrete Mathematics*. Algorithms and Combinatorics, vol. 16, pp. 195–247. Springer, Heidelberg (1998). [https://doi.org/10.1007/978-3-662-12788-9\\_6](https://doi.org/10.1007/978-3-662-12788-9_6)
15. Nallaperuma, S., Neumann, F., Sudholt, D.: Expected fitness gains of randomized search heuristics for the traveling salesperson problem. *Evol. Comput.* **25**(4), 673–705 (2017)
16. Wainwright, M.: Basic tail and concentration bounds. Technical report (2015). Lecture Notes, University of Berkeley. [https://www.stat.berkeley.edu/~mjwain/stat210b/Chap2\\_TailBounds\\_Jan22\\_2015.pdf](https://www.stat.berkeley.edu/~mjwain/stat210b/Chap2_TailBounds_Jan22_2015.pdf)



# On Averaging the Best Samples in Evolutionary Computation

Laurent Meunier<sup>1,2(✉)</sup>, Yann Chevaleyre<sup>2</sup>, Jeremy Rapin<sup>1</sup>,  
Clément W. Royer<sup>2</sup>, and Olivier Teytaud<sup>1</sup>

<sup>1</sup> Facebook Artificial Intelligence Research (FAIR), Paris, France  
[laurentmeunier@fb.com](mailto:laurentmeunier@fb.com)

<sup>2</sup> LAMSADE, CNRS, Université Paris-Dauphine, Université PSL, Paris, France

**Abstract.** Choosing the right selection rate is a long standing issue in evolutionary computation. In the continuous unconstrained case, we prove mathematically that a single parent  $\mu = 1$  leads to a sub-optimal simple regret in the case of the sphere function. We provide a theoretically-based selection rate  $\mu/\lambda$  that leads to better progress rates. With our choice of selection rate, we get a provable regret of order  $O(\lambda^{-1})$  which has to be compared with  $O(\lambda^{-2/d})$  in the case where  $\mu = 1$ . We complete our study with experiments to confirm our theoretical claims.

## 1 Introduction

In evolutionary computation, the selected population size often depends linearly on the total population size, with a ratio between  $1/4$  and  $1/2$ : 0.270 is proposed in [4, 5, 10] suggest  $1/4$  and  $1/2$ . However, some sources [8] recommend a lower value  $1/7$ . Experimental results in [16] and theory in [9] together suggest a ratio  $\min(d, \lambda/4)$  with  $d$  the dimension, i.e. keep a population size at most the dimension. [12] suggests to keep increasing  $\mu$  besides that limit, but slowly enough so that rule  $\mu = \min(d, \lambda/4)$  would be still nearly optimal. Weighted recombination is common [1], but not with a clear gap when compared to truncation ratios [11], except in the case of large population size [17]. There is, overall, limited theory around the optimal choice of  $\mu$  for optimization in the continuous setting. In the present paper, we focus on a simple case (sphere function and single epoch), but prove exact theorems. We point out that the single epoch case is important by itself - this is fully parallel optimization [2, 6, 13, 14]. Experimental results with a publicly available platform support the approach.

## 2 Theory

We consider the case of a single batch of evaluated points. We generate  $\lambda$  points according to some probability distribution. We then select the  $\mu$  best and average them. The result is our approximation of the optimum. This is therefore an extreme case of evolutionary algorithm, with a single population; this is commonly used for e.g. hyperparameter search in machine learning [3, 6], though in most cases with the simplest case  $\mu = 1$ .

### 2.1 Outline

We consider the optimization of the simple function  $x \mapsto \|x - y\|^2$  for an unknown  $y \in \mathcal{B}(0, r)$ . In Sect. 2.2 we introduce notations. In Sect. 2.3 we analyze the case of random search uniformly in a ball of radius  $h$  centered on  $y$ . We can, therefore, exploit the knowledge of the optimum’s position and assume that  $y = 0$ . We then extend the results to random search in a ball of radius  $r$  centered on 0, provided that  $r > \|y\|$  and show that results are essentially the same up to an exponentially decreasing term (Sect. 2.4).

### 2.2 Notations

We are interested in minimizing the function  $f : x \in \mathbb{R}^d \mapsto \|x - y\|^2$  for a fixed unknown  $y$  in parallel one-shot black box optimization, i.e. we sample  $\lambda$  points  $X_1, \dots, X_\lambda$  from some distribution  $\mathcal{D}$  and we search for  $x^* = \arg \min_x f(x)$ . In what follows we will study the sampling from  $\mathcal{B}(0, r)$ , the uniform distribution on the  $\ell_2$ -ball of radius  $r$ ; w.l.o.g.  $\mathcal{B}(y, r)$  will also denote the  $\ell_2$ -ball centered in  $y$  and of radius  $r$ .

We are interested in comparing the strategy “ $\mu$ -best” vs “1-best”. We denote  $X_{(1)}, \dots, X_{(\lambda)}$ , the sorted values of  $X_i$  i.e.  $(1), \dots, (\lambda)$  are such that  $f(X_{(1)}) \leq \dots \leq f(X_{(\lambda)})$ . The “ $\mu$ -best” strategy is to return  $\bar{X}_{(\mu)} = \frac{1}{\mu} \sum_{i=1}^\mu X_{(i)}$  as an estimate of the optimum and the “1-best” is to return  $X_{(1)}$ . We will hence compare :  $\mathbb{E} [f(\bar{X}_{(\mu)})]$  and  $\mathbb{E} [f(X_{(1)})]$ . We recall the definition of the gamma function  $\Gamma : \forall z > 0, \Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$ , as well as the property  $\Gamma(z + 1) = z\Gamma(z)$ .

### 2.3 When the Center of the Distribution is also the Optimum

In this section we assume that  $y = 0$  (i.e.  $f(x) = \|x\|^2$ ) and consider sampling in  $\mathcal{B}(0, r) \subset \mathbb{R}^d$ . In this simple case, we show that keeping the best  $\mu > 1$  sampled points is asymptotically a better strategy than selecting a single best point. The choice of  $\mu$  will be discussed in Sect. 2.4.

**Theorem 1.** *For all  $\lambda > \mu \geq 2$  and  $d \geq 2, r > 0$ , for  $f(x) = \|x\|^2$ ,*

$$\mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0, r)} [f(\bar{X}_{(\mu)})] < \mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0, r)} [f(X_{(1)})].$$

To prove this result, we will compute the value of  $\mathbb{E} [f(\bar{X}_{(\mu)})]$  for all  $\lambda$  and  $\mu$ . The following lemma gives a simple way of computing the expectation of a function depending only on the norm of its argument.

**Lemma 2.** *Let  $d \in \mathbb{N}^*$ . Let  $X$  be drawn uniformly in  $\mathcal{B}(0, r)$  the  $d$ -dimensional ball of radius  $r$ . Then for any measurable function  $g : \mathbb{R} \rightarrow \mathbb{R}$ , we have*

$$\mathbb{E}_{X \sim \mathcal{B}(0, r)} [g(\|X\|)] = \frac{d}{r^d} \int_0^r g(\alpha) \alpha^{d-1} d\alpha.$$

*In particular, we have  $\mathbb{E}_{X \sim \mathcal{B}(0, r)} [\|X\|^2] = \frac{d}{d+2} \times r^2$ .*

*Proof.* Let  $V(r, d)$  be the volume of a ball of radius  $r$  in  $\mathbb{R}^d$  and  $S(r, d)$  be the surface of a sphere of radius  $r$  in  $\mathbb{R}^d$ . Then  $\forall r > 0, V(r, d) = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}r^d$  and  $S(r, d - 1) = \frac{2\pi^{d/2}}{\Gamma(\frac{d}{2})}r^{d-1}$ . Let  $g : \mathbb{R} \rightarrow \mathbb{R}$  be a continuous function. Then:

$$\begin{aligned} \mathbb{E}_{X \sim \mathcal{B}(0,r)} [g(\|X\|)] &= \frac{1}{V(r, d)} \int_{x:\|x\| \leq r} g(\|x\|) dx \\ &= \frac{1}{V(r, d)} \int_{\alpha=0}^r \int_{\theta:\|\theta\|=\alpha} g(\alpha) d\theta d\alpha \\ &= \frac{1}{V(r, d)} \int_{\alpha=0}^r g(\alpha) S(\alpha, d - 1) d\alpha \\ &= \frac{S(1, d - 1)}{V(r, d)} \int_{\alpha=0}^r g(\alpha) \alpha^{d-1} d\alpha = \frac{d}{r^d} \int_{\alpha=0}^r g(\alpha) \alpha^{d-1} d\alpha. \end{aligned}$$

$$\begin{aligned} \text{So, } \mathbb{E}_{X \sim \mathcal{B}(r)} [\|X\|^2] &= \frac{d}{r^d} \int_{\alpha=0}^r \alpha^2 \alpha^{d-1} d\alpha \\ &= \frac{d}{r^d} \left[ \frac{\alpha^{d+2}}{d+2} \right]_0^r = \frac{d}{d+2} r^2. \end{aligned}$$

□

We now use the previous lemma to compute the expected regret [7] of the average of the  $\mu$  best points conditionally to the value of  $f(X_{(\mu+1)})$ . The trick of the proof is that, conditionally to  $f(X_{(\mu+1)})$ , the order of  $X_{(1)}, \dots, X_{(\mu)}$  has no influence over the average. Computing the expected regret conditionally to  $f(X_{(\mu+1)})$  thus becomes straightforward.

**Lemma 3.** For all  $d > 0, r^2 > h > 0$  and  $\lambda > \mu \geq 1$ , for  $f(x) = \|x\|^2$ ,

$$\mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(y,r)} [f(\bar{X}_{(\mu)}) \mid f(X_{(\mu+1)}) = h] = \frac{h}{\mu} \times \frac{d}{d+2}.$$

*Proof.* Let us first compute  $\mathbb{E} [f(\bar{X}_{(\mu)}) \mid f(X_{(\mu+1)}) = h]$ . Note that for any function  $g : \mathbb{R}^d \rightarrow \mathbb{R}$  and distribution  $\mathcal{D}$ , we have

$$\begin{aligned} \mathbb{E}_{X_1 \dots X_\lambda \sim \mathcal{D}} [g(\bar{X}_{(\mu)}) \mid f(X_{(\mu+1)}) = h] \\ &= \mathbb{E}_{X_1 \dots X_\mu \sim \mathcal{D}} \left[ g \left( \frac{1}{\mu} \sum_{i=1}^{\mu} X_i \right) \mid X_1 \dots X_\mu \in \{x : f(x) \leq h\} \right] \\ &= \mathbb{E}_{X_1 \dots X_\mu \sim \mathcal{D}_h} \left[ g \left( \frac{1}{\mu} \sum_{i=1}^{\mu} X_i \right) \right], \end{aligned}$$

where  $\mathcal{D}_h$  is the restriction of  $\mathcal{D}$  to the level set  $\{x : f(x) \leq h\}$ . In our setting, we have  $\mathcal{D} = \mathcal{B}(0, r)$  and  $\mathcal{D}_h = \mathcal{B}(0, \sqrt{h})$ . Therefore,

$$\begin{aligned} &\mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0, r)} [f(\bar{X}(\mu)) \mid f(X_{(\mu+1)}) = h] \\ &= \mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0, r)} [\|\bar{X}(\mu)\|^2 \mid f(X_{(\mu+1)}) = h] \\ &= \mathbb{E}_{X_1 \dots X_\mu \sim \mathcal{B}(0, \sqrt{h})} \left[ \left\| \frac{1}{\mu} \sum_{i=1}^\mu X_i \right\|^2 \right] \\ &= \frac{1}{\mu^2} \mathbb{E}_{X_1 \dots X_\mu \sim \mathcal{B}(0, \sqrt{h})} \left[ \sum_{i,j=1}^\mu X_i^T X_j \right] \\ &= \frac{1}{\mu^2} \sum_{i,j=1, i \neq j}^\mu \mathbb{E}_{X_i \dots X_j \sim \mathcal{B}(0, \sqrt{h})} [X_i^T X_j] \\ &+ \frac{1}{\mu^2} \sum_{i=1}^\mu \mathbb{E}_{X_i \sim \mathcal{B}(0, \sqrt{h})} [\|X_i\|^2] = \frac{1}{\mu} \mathbb{E}_{X \sim \mathcal{B}(0, \sqrt{h})} [\|X\|^2]. \end{aligned}$$

By Lemma 2, we have:  $\mathbb{E}_{X \sim \mathcal{B}(0, \sqrt{h})} [\|X\|^2] = \frac{d}{d+2}h$ . Hence  $\mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0, r)} [f(\bar{X}(\mu)) \mid f(X_{(\mu+1)}) = h] = \frac{d}{d+2} \frac{h}{\mu}$ . □

The result of Lemma 3 shows that  $\mathbb{E} [f(\bar{X}(\mu)) \mid f(X_{(\mu+1)}) = h]$  depends linearly on  $h$ . We now establish a similar dependency for  $\mathbb{E} [f(X_{(1)}) \mid f(X_{(\mu+1)}) = h]$ .

**Lemma 4.** For  $d > 0$ ,  $h > 0$ ,  $\lambda > \mu \geq 1$ , and  $f(x) = \|x\|^2$ ,

$$\mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0, r)} [f(X_{(1)}) \mid f(X_{(\mu+1)}) = h] = h \frac{\Gamma(\frac{d+2}{d})\Gamma(\mu+1)}{\Gamma(\mu+1+2/d)}.$$

*Proof.* First note that using the same argument as in Lemma 3,  $\forall \beta \in (0, h]$ :

$$\begin{aligned} &\mathbb{P}_{X_1 \dots X_\lambda \sim \mathcal{B}(0, \sqrt{h})} [f(X_{(1)}) > \beta \mid f(X_{(\mu+1)}) = h] \\ &= \mathbb{P}_{X_1 \dots X_\mu \sim \mathcal{B}(0, \sqrt{h})} [f(X_1) > \beta, \dots, f(X_\mu) > \beta] \\ &= \mathbb{P}_{X \sim \mathcal{B}(0, \sqrt{h})} [f(X) > \beta]^\mu. \end{aligned}$$

Recall that the volume of a  $d$ -dimensional ball of radius  $r$  is proportional to  $r^d$ . Thus, we get:

$$\mathbb{P}_{X \sim \mathcal{B}(0, \sqrt{h})} [f(X) < \beta] = \frac{\sqrt{\beta}^d}{\sqrt{h}^d} = \left(\frac{\beta}{h}\right)^{\frac{d}{2}}.$$



It is known that for every positive random variable  $X$ ,  $\mathbb{E}(X) = \int_0^\infty \mathbb{P}(X > \beta) d\beta$ . Therefore:

$$\begin{aligned} \mathbb{E}_S [f(X_{(1)}) \mid f(X_{(\mu+1)}) = h] &= \int_0^h \mathbb{P} [f(X_{(1)}) > \beta \mid f(X_{(\mu+1)}) = h] d\beta \\ &= \int_0^h \left(1 - \left(\frac{\beta}{h}\right)^{\frac{d}{2}}\right)^\mu d\beta \\ &= h \int_0^1 \left(1 - u^{\frac{d}{2}}\right)^\mu du \\ &= h \frac{2}{d} \int_0^1 (1-t)^\mu t^{2/d-1} dt = h \frac{\Gamma(\frac{d+2}{d})\Gamma(\mu+1)}{\Gamma(\mu+1+2/d)}. \end{aligned}$$

To obtain the last equality, we identify the integral with the beta function of parameters  $\mu + 1$  and  $\frac{2}{d}$ . □

We now directly compute  $\mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0,r)} [f(X_{(1)})]$ .

**Lemma 5.** *For all  $d > 0$ ,  $\lambda > 0$  and  $r > 0$ :*

$$\mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0,r)} [f(X_{(1)})] = r^2 \frac{\Gamma(\frac{d+2}{d})\Gamma(\lambda+1)}{\Gamma(\lambda+1+2/d)}.$$

*Proof.* As in Lemma 4, we have for any  $\beta \in (0, r^2]$ :

$$\begin{aligned} \mathbb{P}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(X_{(1)}) > \beta] &= \mathbb{P}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(X_1) > \beta, \dots, f(X_\lambda) > \beta] \\ &= \mathbb{P}_{X \sim \mathcal{B}(0,r)} [f(X) > \beta]^\lambda \\ &= \left(\frac{\sqrt{\beta}}{r}\right)^d. \end{aligned}$$

The result then follows by reasoning as in the proof of Lemma 4. □

By combining the results above, we obtain the exact formula for  $\mathbb{E} [f(\bar{X}_{(\mu)})]$ .

**Theorem 6.** *For all  $d > 0$ ,  $r > 0$  and  $\lambda > \mu \geq 1$ :*

$$\mathbb{E}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(\bar{X}_{(\mu)})] = \frac{r^2 d \times \Gamma(\lambda+1)\Gamma(\mu+1+2/d)}{\mu(d+2)\Gamma(\mu+1)\Gamma(\lambda+1+2/d)}.$$

*Proof.* The proof follows by applying our various lemmas and integrating over all possible values for  $h$ . We have:

$$\begin{aligned}
 & \mathbb{E}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(\bar{X}(\mu))] \\
 &= \mathbb{E} [\mathbb{E} [f(\bar{X}(\mu)) \mid f(X_{(\mu+1)})]] \\
 &= \frac{1}{\mu} \frac{d}{d+2} \mathbb{E} [f(X_{(\mu+1)})] \text{ by Lemma 3} \\
 &= \frac{1}{\mu} \frac{d}{d+2} \frac{\Gamma(\mu+1+2/d)}{\Gamma(\mu+1)\Gamma(\frac{d+2}{d})} \mathbb{E} [\mathbb{E} [f(X_{(1)}) \mid f(X_{(\mu+1)})]] \text{ by Lemma 4} \\
 &= \frac{1}{\mu} \frac{d}{d+2} \frac{\Gamma(\mu+1+2/d)}{\Gamma(\mu+1)\Gamma(\frac{d+2}{d})} \mathbb{E} [f(X_{(1)})] \\
 &= \frac{r^2 d \times \Gamma(\lambda+1)\Gamma(\mu+1+2/d)}{\mu(d+2)\Gamma(\mu+1)\Gamma(\lambda+1+2/d)} \text{ by Lemma 5.}
 \end{aligned}$$

□

We have checked experimentally the result of Theorem 9 (see Fig. 1): the result of Theorem 1 follows from Theorem 9 since for  $d \geq 2$ ,  $\lambda$  and  $r$  fixed,  $\mathbb{E} [f(\bar{X}(\mu))]$  is strictly decreasing in  $\mu$ . In addition, we can obtain asymptotic progress rates:

**Corollary 7.** *Consider  $d > 0$ . When  $\lambda \rightarrow \infty$ , we have*

$$\mathbb{E}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(\bar{X}(\mu))] \sim \lambda^{-\frac{2}{d}} \frac{r^2 d \times \Gamma(\mu+1+2/d)}{\mu(d+2)\Gamma(\mu+1)},$$

while if  $\lambda \rightarrow \infty$  and  $\mu(\lambda) \rightarrow \infty$ ,  $\mathbb{E}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(\bar{X}(\mu(\lambda)))] \sim r^2 \frac{d}{d+2} \frac{\mu(\lambda)^{\frac{2}{d}-1}}{\lambda^{\frac{2}{d}}}$ .

As a result,  $\forall c \in (0, 1)$ ,  $\mathbb{E} (f(\bar{X}_{\lfloor c\lambda \rfloor})) \in \Theta(\frac{1}{\lambda})$  and  $\mathbb{E} (f(X_{(1)})) \in \Theta(\frac{1}{\lambda^{2/d}})$ .

*Proof.* We recall the Stirling equivalent formula for the gamma function: when  $z \rightarrow \infty$ ,

$$\Gamma(z) = \sqrt{\frac{2\pi}{z}} \left(\frac{z}{e}\right)^z \left(1 + O\left(\frac{1}{z}\right)\right).$$

Using this approximation, we get the expected results. □

This result shows that by keeping a single parent, we lose more than a constant factor: the progress rate is significantly impacted. Therefore it is preferable to use more than one parent.

### 2.4 Convergence When the Sampling is not Centered on the Optimum

So far we treated the case where the center of the distribution and the optimum are the same. We now assume that we sample from the distribution  $\mathcal{B}(0, r)$  and that the function  $f$  is  $f(x) = \|x - y\|^2$  with  $\|y\| \leq r$ . We define  $\epsilon = \frac{\|y\|}{r}$ .

**Lemma 8.** *Let  $r > 0, d > 0, \lambda > \mu \geq 1$ , we have:*

$$\mathbb{P}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)}(f(X_{(\mu+1)}) > (1 - \epsilon)^2 r^2) = \mathbb{P}_{U \sim \mathcal{B}(\lambda, (1-\epsilon)^d)}(U \leq \mu),$$

where  $B(\lambda, p)$  is a binomial law of parameters  $\lambda$  and  $p$ .

*Proof.* We have  $f(X_{(\mu+1)}) > (1 - \epsilon)r \iff \sum_{i=1}^\lambda \mathbb{1}_{\{f(X_i) \leq (1-\epsilon)^2 r^2\}} \leq \mu$  since  $\mathbb{1}_{\{f(X_i) \leq (1-\epsilon)^2 r^2\}}$  are independent Bernoulli variables of parameter  $(1-\epsilon)^d$ , hence the result.  $\square$

Using Lemma 8, we now get lower and upper bounds on  $\mathbb{E}[f(X_{(\mu+1)})]$ :

**Theorem 9.** *Consider  $d > 0, r > 0, \lambda > \mu \geq 1$ . The expected value of  $f(\bar{X}_\mu)$  satisfies both*

$$\begin{aligned} \mathbb{E}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(\bar{X}_\mu)] &\leq 4r^2 \mathbb{P}_{U \sim \mathcal{B}(\lambda, (1-\epsilon)^d)}(U \leq \mu) \\ &\quad + \frac{r^2 d \times \Gamma(\lambda + 1) \Gamma(\mu + 1 + 2/d)}{\mu(d + 2) \Gamma(\mu + 1) \Gamma(\lambda + 1 + 2/d)} \\ \text{and } \mathbb{E}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(\bar{X}_\mu)] &\geq \frac{r^2 d \times \Gamma(\lambda + 1) \Gamma(\mu + 1 + 2/d)}{\mu(d + 2) \Gamma(\mu + 1) \Gamma(\lambda + 1 + 2/d)}. \end{aligned}$$

*Proof.*

$$\begin{aligned} \mathbb{E}[f(\bar{X}_\mu)] &= \mathbb{E}(f(\bar{X}_\mu) | f(X_{(\mu+1)}) \geq (1 - \epsilon)^2 r^2) \mathbb{P}(f(X_{(\mu+1)}) \geq (1 - \epsilon)^2 r^2) \\ &\quad + \mathbb{E}(f(\bar{X}_\mu) | f(X_{(\mu+1)}) < (1 - \epsilon)^2 r^2) \mathbb{P}(f(X_{(\mu+1)}) < (1 - \epsilon)^2 r^2). \end{aligned}$$

In this Bayes decomposition, we can bound the various terms as follows:

$$\begin{aligned} \mathbb{E}(f(\bar{X}_\mu) | f(X_{(\mu+1)}) \geq (1 - \epsilon)^2 r^2) &\leq 4r^2, \\ \mathbb{P}(f(X_{(\mu+1)}) \geq (1 - \epsilon)^2 r^2) &\leq 1, \\ \mathbb{E}[f(\bar{X}_\mu) | f(X_{(\mu+1)}) < (1 - \epsilon)^2 r^2] &\leq \frac{r^2 d \times \Gamma(\lambda + 1) \Gamma(\mu + 1 + 2/d)}{\mu(d + 2) \Gamma(\mu + 1) \Gamma(\lambda + 1 + 2/d)}. \end{aligned}$$

Combining these equations yields the first (upper) bound. The second (lower) bound is deduced from the centered case (i.e. when the distribution is centered on the optimum) as in the previous section.  $\square$

Figure 2 gives an illustration of the bounds. Until  $\mu \simeq (1 - \epsilon)^d \lambda$ , the centered and non centered case coincide when  $\lambda \rightarrow \infty$ : in this case, we can have a more precise asymptotic result for the choice of  $\mu$ .

**Theorem 10.** *Consider  $d > 0, r > 0$  and  $y \in \mathbb{R}^d$ . Let  $\epsilon = \frac{\|y\|}{r} \in [0, 1)$  and  $f(x) = \|x - y\|^2$ . When using  $\mu = \lfloor c\lambda \rfloor$  with  $0 < c < (1 - \epsilon)^d$ , we get as  $\lambda \rightarrow \infty$ , for a fixed  $d$ ,*

$$\mathbb{E}_{X_1 \dots X_\lambda \sim \mathcal{B}(0,r)} [f(\bar{X}_\mu)] = \frac{dr^2 c^{2/d-1}}{(d + 2)\lambda} + o\left(\frac{1}{\lambda}\right).$$

*Proof.* Let  $\mu_\lambda = \lfloor c\lambda \rfloor$  with  $0 < c < (1 - \epsilon)^d$ . We immediately have from Hoeffding’s concentration inequality:

$$\mathbb{P}_{U \sim B(\lambda, (1-\epsilon)^d)} (U \leq \mu_\lambda) \in o\left(\frac{1}{\lambda}\right)$$

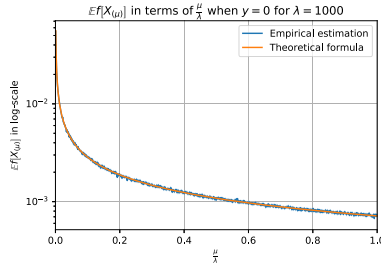
when  $\lambda \rightarrow \infty$ . From Corollary 7, we also get:

$$\frac{r^2 d \times \Gamma(\lambda + 1)\Gamma(\mu_\lambda + 1 + 2/d)}{\mu_\lambda(d + 2)\Gamma(\mu_\lambda + 1)\Gamma(\lambda + 1 + 2/d)} \sim \frac{dr^2 c^{2/d-1}}{(d + 2)\lambda}.$$

Using the inequalities of Theorem 9, we obtain the desired result. □

The result of Theorem 10 shows that a convergence rate  $O(\lambda^{-1})$  can be attained for the  $\mu$ -best approach with  $\mu > 1$ . The rate for  $\mu = 1$  is  $\Theta(\lambda^{-2/d})$ , proving that the  $\mu$ -best approach leads asymptotically to a better estimation of the optimum. If we consider the problem  $\min_\mu \max_{y: \|y\| \leq \epsilon r} \mathbb{E} [f_y(\bar{X}(\mu))]$  with  $f_y$  the objective function  $x \mapsto \|x - y\|^2$ , then  $\mu = \lfloor c\lambda \rfloor$  with  $0 < c < (1 - \epsilon)^d$  achieves the  $O(\lambda^{-1})$  progress rate.

All the results we proved in this section are easily extendable to strongly convex quadratic functions. For larger class of functions, it is less immediate, and left as future work.



**Fig. 1.** Centered case: validation of the theoretical formula for  $\mathbb{E}_{X_1 \dots X_\lambda \sim B(0,r)} [f(\bar{X}(\mu))]$  when  $y = 0$  from Theorem 6 for  $d = 5$ ,  $\lambda = 1000$  and  $R = 1$ . 1000 samples have been drawn to estimate the expectation. The two curves overlap, showing agreement between theory and practice.

### 2.5 Using Quasi-convexity

The method above was designed for the sphere function, yet its adaptation to other quadratic convex functions is straightforward. On the other hand, our reasoning might break down when applied to multimodal functions. We thus consider an adaptive strategy to define  $\mu$ . A desirable property to a  $\mu$ -best approach is that the level-sets of the functions are convex. A simple workaround is to choose  $\mu$  maximal such that there is a quasi-convex function which is identical

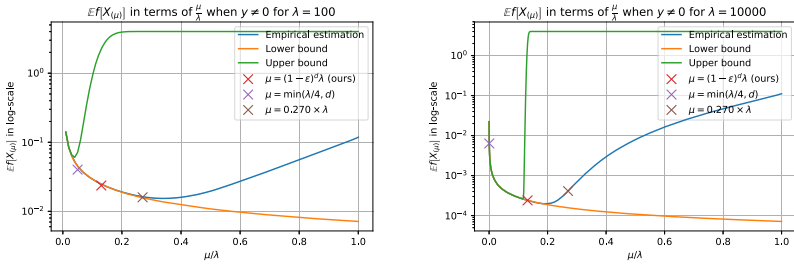
to  $f$  on  $\{X_{(1)}, \dots, X_{(\mu)}\}$ . If the objective function is quasi-convex on the convex hull of  $\{X_{(1)}, \dots, X_{(\tilde{\mu})}\}$  with  $\tilde{\mu} \leq \lambda$ , then: for any  $i \leq \tilde{\mu}$ ,  $X_{(i)}$  is on the frontier (denoted  $\partial$ ) of the convex hull of  $\{X_{(1)}, \dots, X_{(i)}\}$  and the value

$$h = \max \{i \in [1, \lambda], \forall j \leq i, X_{(j)} \in \partial [\text{ConvexHull}(X_{(1)}, \dots, X_{(j)})]\}$$

verifies  $h \geq \tilde{\mu}$  so that  $\mu = \min(h, \tilde{\mu})$  is actually equal to  $\tilde{\mu}$ . As a result:

- in the case of the sphere function, or any quasi-convex function, if we set  $\tilde{\mu} = \lfloor \lambda(1-\epsilon)^d \rfloor$ , using  $\mu = \min(h, \tilde{\mu})$  leads to the same value of  $\mu = \tilde{\mu} = \lfloor \lambda(1-\epsilon)^d \rfloor$ . In particular, we preserve the theoretical guarantees of the previous sections for the sphere function  $x \mapsto \|x - y\|^2$ .
- if the objective function is not quasi-convex, we can still compute the quantity  $h$  defined above, but we might get a  $\mu$  smaller than  $\tilde{\mu}$ . However, this strategy remains meaningful as it prevents from keeping too many points when the function is “highly” non-quasi-convex.

### 3 Experiments



**Fig. 2.** Non centered case: validation of the theoretical bounds for  $\mathbb{E}_{X_1, \dots, X_\lambda \sim \mathcal{B}(0, r)} [f(\bar{X}_\mu)]$  when  $\|y\| = \frac{R}{3}$  (i.e.  $\epsilon = \frac{1}{3}$ ) from Theorem 9 for  $d = 5$  and  $R = 1$ . We implemented  $\lambda = 100$  and  $\lambda = 10000$ . 10000 samples have been drawn to estimate the expectation. We see that such a value for  $\mu$  is a good approximation of the minimum of the empirical values: we can thus recommend  $\mu = \lfloor \lambda(1 - \epsilon)^d \rfloor$  when  $\lambda \rightarrow \infty$ . We also added some classical choices of values for  $\mu$  from literature: when  $\lambda \rightarrow \infty$ , our method performs the best.

To validate our theoretical findings, we first compare the formulas obtained in Theorems 6 and 9 with their empirical estimates. We then perform larger scale experiments in a one-shot optimization setting.

#### 3.1 Experimental Validation of Theoretical Formulas

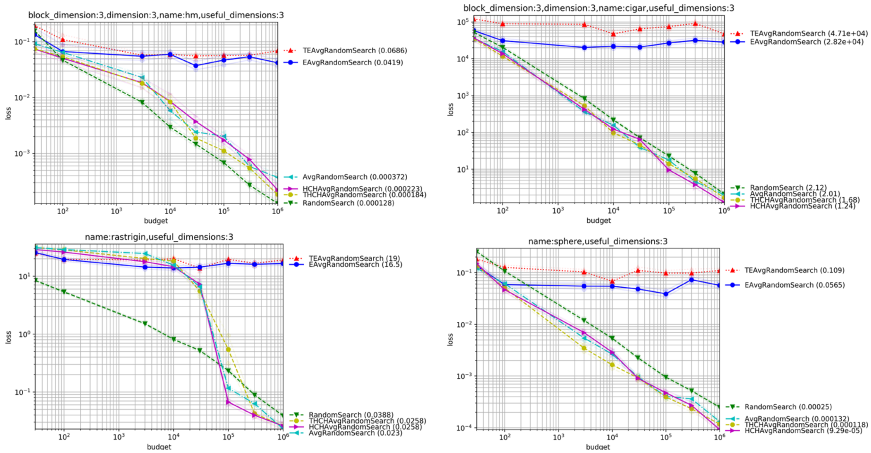
Figure 1 compares the theoretical formula from Theorem 6 and its empirical estimation: we note that the results coincide and validate our formula. Moreover,

the plot confirms that taking the  $\mu$ -best points leads to a lower regret than the 1-best approach.

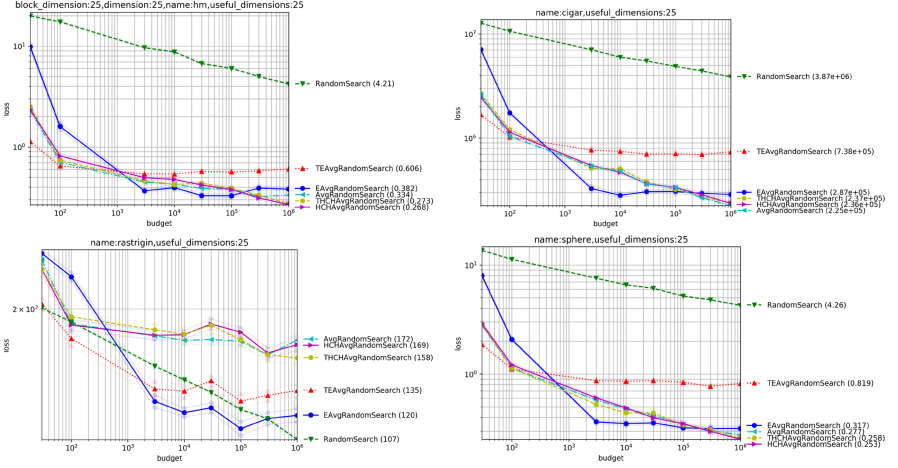
We also compare in Fig. 2 the theoretical bounds from Theorem 9 with their empirical estimates. We remark that for  $\mu \leq (1-\epsilon)^d \lambda$  the convergence of the two bounds to  $\mathbb{E}(f(\bar{X}_{(\mu)}))$  is fast. There exists a transition phase around  $\mu \simeq (1-\epsilon)^d \lambda$  on which the regret is reaching a minimum: thus, one needs to choose  $\mu$  both small enough to reduce bias and large enough to reduce variance. We compared to other empirically estimated values for  $\mu$  from [4, 5, 10]. It turns out that if the population is large, our formula for  $\mu$  leads to a smaller regret. Note that our strategy assumes that  $\epsilon$  is known, which is not the case in practice. It is interesting to note that if the center of the distribution and the optimum are close (i.e.  $\epsilon$  is small), one can choose a larger  $\mu$  to get a lower variance on the estimator of the optimum.

### 3.2 One-Shot Optimization in Nevergrad

In this section we test different formulas and variants for the choice of  $\mu$  for a larger scale of experiments in the one-shot setting. Equations 1–6 present the different formulas for  $\mu$  used in our comparison.



**Fig. 3.** Experimental curves comparing various methods for choosing  $\mu$  as a function of  $\lambda$  in dimension 3. Standard deviations are shown by lighter lines (close to the average lines). Each x-axis value is computed independently. Our proposed formulas HCHAVg and THCHAVg perform well overall. See Fig. 4 for results in dimension 25.



**Fig. 4.** Experimental curves comparing various methods for choosing  $\mu$  as a function of  $\lambda$  in dimension 25 (Fig. 3, continued for dimension 25; see Fig. 5 for dimension 200). Our proposals lead to good results but we notice that they are outperformed by TEAvg and EAvg for Rastrigin: it is better to not take into account non-quasi-convexity because the overall shape is more meaningful than local ruggedness. This phenomenon does not happen for the more rugged HM (Highly Multimodal) function. It also does not happen in dimension 3 or dimension 200 (previous and next figures): in those cases, THCH performed best. Confidence intervals shown in lighter color (they are quite small, and therefore they are difficult to notice).

$$\mu = 1 \quad \text{No prefix} \quad (1)$$

$$\mu = \text{clip} \left( 1, d, \frac{\lambda}{4} \right) \quad \text{Prefix: Avg (averaging)} \quad (2)$$

$$\mu = \text{clip} \left( 1, \infty, \frac{\lambda}{1.1d} \right) \quad \text{Prefix: EAvg (Exp. Averaging)} \quad (3)$$

$$\mu = \text{clip} \left( 1, \min \left( h, \frac{\lambda}{4} \right), d + \frac{\lambda}{1.1d} \right) \quad \text{Prefix: HCAvg (} h \text{ from Convex Hull)} \quad (4)$$

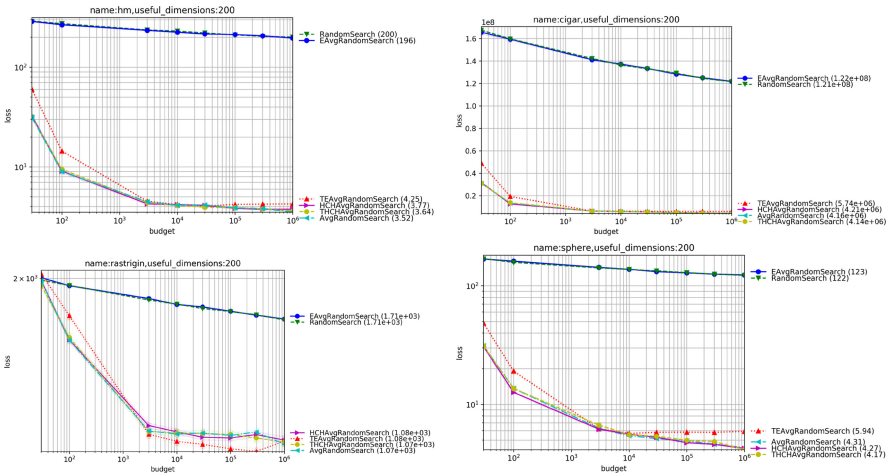
$$\mu = \text{clip} \left( 1, \infty, \frac{\lambda}{1.01d} \right) \quad \text{Prefix: TEAvg (Tuned Exp. Avg)} \quad (5)$$

$$\mu = \text{clip} \left( 1, \min \left( h, \frac{\lambda}{4} \right), d + \frac{\lambda}{1.01d} \right) \quad \text{Prefix: THCAvg (Tuned HCH Avg)} \quad (6)$$

where  $\text{clip}(a, b, c) = \max(a, \min(b, c))$  is the projection of  $c$  in  $[a, b]$  and  $h$  is the maximum  $i$  such that, for all  $j \leq i$ ,  $X_{(j)}$  is on the frontier of the convex hull of  $\{X_{(1)}, \dots, X_{(j)}\}$  (Sect. 2.5). Equation 1 is the naive recommendation “pick up the best so far”. Equation 2 existed before the present work: it was, until now, the best rule [16], overall, in the Nevergrad platform. Equations 3 and 5 are the proposals we deduced from Theorem 10: asymptotically on the sphere, they should have a better rate than Eq. 1. Equations 4 and 6 are counterparts of Eqs. 3 and 5 that combine the latter formulas with ideas from [16]. Theorem

10 remains true if we add to  $\mu$  some constant depending on  $d$  so we fine tune our theoretical equation (Eq. 3) with the one provided by [16], so that  $\mu$  is close to the value in Eq. 2 for moderate values of  $\lambda$ . We perform experiments in the open source platform Nevergrad [15].

While previous experiments (Figs. 1 and 2) were performed in a controlled ad hoc environment, we work here with more realistic conditions: the sampling is Gaussian (i.e. not uniform in a ball), the objective functions are not all sphere-like, and budgets vary but are not asymptotic. Figures 3, 4, 5 present our results in dimension 3, 25 and 200 respectively. The objective functions are randomly translated using  $\mathcal{N}(0, 0.2I_d)$ . The objective functions are defined as  $f_{Sphere}(x) = \|x\|^2$ ,  $f_{Cigar}(x) = 10^6 \sum_{i=2}^d x_i^2 + x_1^2$ ,  $f_{HM}(x) = \sum_{i=1}^d x_i^2 \times (1.1 + \cos(1/x_i))$ ,  $f_{Rastrigin}(x) = 10d + f_{sphere}(x) - 10 \sum_i \cos(2\pi x_i)$ . Our proposed equations TEAvg and EAvg are unstable: they sometimes perform excellently (e.g. everything in dimension 25, Fig. 4), but they can also fail dramatically (e.g. dimension 3, Fig. 3). Our combinations THCHAVg and HCHAVg perform well: in most settings, THCHAVg performs best. But the gap with the previously proposed Avg is not that big. The use of quasi-convexity as described in Sect. 2.5 was usually beneficial: however, in dimension 25 for the Rastrigin function, it prevented the averaging from benefiting from the overall “approximate” convexity of Rastrigin. This phenomenon did not happen for the “more” multimodal function HM, or in other dimensions for the Rastrigin function.



**Fig. 5.** Experimental curves comparing various methods for choosing  $\mu$  as a function of  $\lambda$  in dimension 200 (Fig. 3 and 4, continued for dimension 200). Confidence intervals shown in lighter color (they are quite small, and therefore they are difficult to notice). Our proposed methods THCHAVg and HCHAVg perform well overall.



## 4 Conclusion

We have proved formally that the average of the  $\mu$  best is better than the single best in the case of the sphere function (simple regret  $O(1/\lambda)$  instead of  $O(1/\lambda^{2/d})$ ) with uniform sampling. We suggested a value  $\mu = \lfloor c\lambda \rfloor$  with  $0 < c < (1 - \epsilon)^d$ . Even better results can be obtained in practice using quasi-convexity, without losing the theoretical guarantees of the convex case on the sphere function. Our results have been successfully implemented in [15]. The improvement compared to the state of the art, albeit moderate, is obtained without any computational overhead in our method, and supported by a theoretical result.

**Further Work.** Our theorem is limited to a single iteration, i.e. fully parallel optimization, and to the sphere function. Experiments are positive in the convex case, encouraging more theoretical developments in this setting. We did not explore approaches based on surrogate models. Our experimental methods include an automatic choice of  $\mu$  in the multimodal case using quasi-convexity, for which the theoretical analysis has yet to be fully developed - we show that this is not detrimental in the convex setting, but not that it performs better in a non-convex setting. We need an upper bound on the distance between the center of the sampling and the optimum for our results to be applicable (see parameter  $\epsilon$ ): removing this need is a worthy consideration, as such a bound is rarely available in real life.

## References

1. Arnold, D.V.: Optimal weighted recombination. In: Wright, A.H., Vose, M.D., De Jong, K.A., Schmitt, L.M. (eds.) FOGA 2005. LNCS, vol. 3469, pp. 215–237. Springer, Heidelberg (2005). [https://doi.org/10.1007/11513575\\_12](https://doi.org/10.1007/11513575_12)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. *JMLR* **13**, 281–305 (2012)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **13**, 281–305 (2012)
4. Beyer, H.G., Schwefel, H.P.: Evolution strategies -a comprehensive introduction. *Natural Comput. Int. J.* **1**(1), 3–52 (2002). <https://doi.org/10.1023/A:1015059928466>
5. Beyer, H.-G., Sendhoff, B.: Covariance matrix adaptation revisited – the CMSA evolution strategy -. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 123–132. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-87700-4\\_13](https://doi.org/10.1007/978-3-540-87700-4_13)
6. Bousquet, O., Gelly, S., Karol, K., Teytaud, O., Vincent, D.: Critical hyper-parameters: No random, no cry (2017, preprint). <https://arxiv.org/pdf/1706.03200.pdf>
7. Bubeck, S., Munos, R., Stoltz, G.: Pure exploration in multi-armed bandits problems. In: Gavaldà, R., Lugosi, G., Zeugmann, T., Zilles, S. (eds.) ALT 2009. LNCS (LNAI), vol. 5809, pp. 23–37. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-04414-4\\_7](https://doi.org/10.1007/978-3-642-04414-4_7)
8. Escalante, H., Reyes, A.M.: Evolution strategies. CCC-INAOE tutorial (2013)

9. Fournier, H., Teytaud, O.: Lower bounds for comparison based evolution strategies using VC-dimension and sign patterns. *Algorithmica* (2010)
10. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* **11**(1), 159–195 (2003)
11. 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)
12. Jebalia, M., Auger, A.: Log-linear convergence of the scale-invariant  $(\mu/\mu_{w\lambda})$ -ES and optimal  $\mu$  for intermediate recombination for large population sizes. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) *PPSN 2010. LNCS*, vol. 6238, pp. 52–62. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_6](https://doi.org/10.1007/978-3-642-15844-5_6)
13. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21**(2), 239–245 (1979)
14. Niederreiter, H.: *Random Number Generation and quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia (1992)
15. Rapin, J., Teytaud, O.: Nevergrad - A gradient-free optimization platform. <https://GitHub.com/FacebookResearch/Nevergrad> (2018)
16. Teytaud, F.: A new selection ratio for large population sizes. In: Di Chio, C., et al. (eds.) *EvoApplications 2010. LNCS*, vol. 6024, pp. 452–460. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-12239-2\\_47](https://doi.org/10.1007/978-3-642-12239-2_47)
17. Teytaud, F., Teytaud, O.: Why one must use reweighting in estimation of distribution algorithms. In: *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, 8–12 July 2009*, pp. 453–460 (2009)



# Filter Sort Is $\Omega(N^3)$ in the Worst Case

Sumit Mishra<sup>1</sup> and Maxim Buzdalov<sup>2</sup>(✉)

<sup>1</sup> IIIT Guwahati, Guwahati, India  
sumit@iiitg.ac.in

<sup>2</sup> ITMO University, Saint Petersburg, Russia  
mbuzdalov@gmail.com

**Abstract.** Non-dominated sorting is a crucial operation used in many popular evolutionary multiobjective algorithms. The problem of non-dominated sorting, although solvable in polynomial time, is surprisingly difficult, and no algorithm is yet known which solves any instance on  $N$  points and  $M$  objectives in time asymptotically smaller than  $MN^2$ .

For this reason, many algorithm designers concentrate on reducing the leading constant and on (implicitly) tailoring their algorithms to inputs typical to evolutionary computation. While doing that, they sometimes forget to ensure that the worst-case running time of their algorithm is still  $O(MN^2)$ . This is undesirable, especially if the inputs which make the algorithm work too slow can occur spontaneously. However, even if a counterexample is hard to find, the fact that it exists is still a weak point, as this can be exploited and lead to denial of service and other kinds of misbehaving.

In this paper we prove that a recent algorithm for non-dominated sorting, called Filter Sort, has the worst-case complexity of  $\Omega(N^3)$ . In particular, we present a scenario which requires Filter Sort to perform  $\Theta(N^3)$  dominance comparisons, where each comparison, however, needs only  $O(1)$  elementary operations. Our scenario contains  $\Theta(N)$  non-domination layers, which is a necessary, but by no means a sufficient condition for being difficult for Filter Sort.

**Keywords:** Non-dominated sorting · Filter sort · Time complexity

## 1 Introduction

Optimizers that rank solutions based on the Pareto dominance relation arguably prevail in evolutionary multiobjective optimization for already more than two decades, and only relatively recently they started to partially lose many-objective ground to decomposition-based methods. In turn, among the ranking methods that employ the Pareto dominance relation, non-dominated sorting is probably one of the most frequently used. With a relatively small computation cost and a possibility of writing a relatively easy implementation, it is now used not only in the algorithm NSGA-II [9] that popularized it, but in a wide range of algorithms

belonging to different paradigms, such as Strength Pareto Evolutionary Algorithm (SPEA), Pareto Envelope-Based Selection Algorithm (PESA) [6], Pareto Archived Evolution Strategy (PAES) [14], M-Pareto Archived Evolution Strategy (M-PAES) [13], Micro-GA [5], KnEA [38], NSGA-III [8] and many others.

Assume that there are  $M$  objectives and, without loss of generality, that we are required to minimize all objectives. A solution  $p$  is said to dominate, in Pareto sense, another solution  $q$ , which is written as  $p \prec q$ , if the following conditions are satisfied:

1.  $\forall i \in [1..M]$  it holds that  $p_i \leq q_i$ ;
2.  $\exists j \in [1..M]$  such that  $p_j < q_j$ ;

where the notation is as follows:  $[a..b]$  is the set of integers  $\{a, a + 1, \dots, b - 1, b\}$ .

Non-dominated sorting can then be defined as follows. Let  $\mathbb{P} = \{\mathbb{P}_1, \dots, \mathbb{P}_N\}$  be a population of  $N$  evaluated solutions. The problem is to divide  $\mathbb{P}$  into several fronts  $\mathbb{F} = \{\mathbb{F}_1, \mathbb{F}_2, \dots\}$ , such that the following conditions are satisfied:

1. The fronts constitute a partition, that is:
  - $\bigcup_i \mathbb{F}_i = \mathbb{P}$ ;
  - if  $i \neq j$ , then  $\mathbb{F}_i \cap \mathbb{F}_j = \emptyset$ ;
2. For any two solutions  $s, t \in \mathbb{F}_i$ , neither of them dominates another one;
3. For  $i > 1$ , for any  $s \in \mathbb{F}_i$  there exists some  $t \in \mathbb{F}_{i-1}$  such that  $t \prec s$ .

We shall explicitly state here that, for the soundness of the definition above, a solution does not dominate itself (as well as it does not dominate any other solution with identical objective values), as otherwise the ordering of the solutions would affect the results of the procedure in an implementation-dependent way. However, it is often desirable, that, whenever there are three solutions  $p_1, p_2, q$  such that  $p_1 = p_2$  and  $p_1 \prec q$ , the solution  $q$  gets a worse rank than it would have without  $p_2$ . The reader is welcome to an extended discussion about possible sound extensions of this version of non-dominated sorting, as well as other similar problems, in a recent paper [2].

Outside evolutionary computation, non-dominated sorting is often known under other names (such as *layers of maxima* or *longest chains*) and has applications in various domains like data clustering [11], graph theory [17], computer vision, economics and game theory [16], database [1] and others [7, 28, 29].

Since in this paper we are interested in algorithms for non-dominated sorting, and do not investigate its applications, we do not differentiate between solutions and their objective vectors, treat them as points in an  $M$ -dimensional space and use “points” and “solutions” interchangeably.

The apparent simplicity of the non-dominated sorting problem makes it surprising that, despite a huge effort, no algorithms are still known that solve this problem in time  $o(MN^2)$  for any input of  $N$  points and  $M$  objectives. Below, we give a quick summary of the basic ideas of these algorithms.

There are plenty of algorithms that run in  $\Theta(MN^2)$  time in the worst case, beginning with the algorithm called “fast non-dominated sorting” that accompanied NSGA-II [9], as well as more advanced approaches [10, 23–27, 30, 31, 33, 36, 37]

and some others. All these algorithms focus on the running time on inputs with rather small values of  $N$ , and on those that are distributed similar to typical populations in evolutionary multiobjective optimization. The basic desire driving most of these designs is to somehow “reduce the number of unnecessary comparisons” with certain heuristics that work reasonably well under uniform or other similar distributions. Most of these algorithms cannot cope with  $N = 10^5$  points, however, there are notable exceptions, such as the kd-tree-based algorithm called ENS-NDT [10] and, to some extent, the flavours of Best Order Sort [23, 25, 30, 31].

The algorithms belonging to a different group apply the divide-and-conquer paradigm in a particular manner that allows an asymptotically better upper bound of  $O(N(\log N)^{M-1})$  to be proven. This expression holds when  $M$  is constant with regards to  $N$ , and all these algorithms also satisfy the  $O(MN^2)$  upper bound, which they quickly reach with large enough  $M$ . The divide-and-conquer flavour in question is suggested long time ago in [15], and it was applied for the first time to non-dominated sorting in [12]. Subsequent development involved modifications to reliably work on every input [4] and various practical runtime improvements typically involving hybridization with other algorithms [18, 19], and word-RAM data structures [3].

However, some algorithms have even worse running time guarantees. The original NSGA algorithm [32] featured a particularly naive algorithm that works in  $O(MN^3)$  time, where one of the  $N$  factors is the number of fronts in the output. Unfortunately, some of the algorithms are also that slow. On some occasions it is trivial to show, as it was the case with the DDA sorting [22, 39], however, sometimes the original paper features a wrong optimistic bound, such which can be non-obvious to prove, and even more difficult to persuade the scientific community that it is true [20, 21].

In this paper, we focus on a fairly recent algorithm called Filter Sort [34]. Although this algorithm bears a large resemblance with Best Order Sort, which is  $O(MN^2)$ , it loses this worst-case bound in an attempt to speed-up. We present the test scenario which requires this algorithm to perform  $\Omega(N^3)$  dominance comparisons between the points; however, because of a particular property of this algorithm, we cannot guarantee that even a constant fraction of these comparisons will take  $\Omega(M)$  time, so our running time lower bound is as well  $\Omega(N^3)$ .

The rest of the paper is structured as follows. Section 2 explains Filter Sort in necessary detail. Then we present our test scenario in Sect. 3 and show that Filter Sort runs in  $\Omega(N^3)$  on this scenario. Finally, we conclude the paper and discuss the consequences of our results in Sect. 4.

## 2 Filter Sort

In this section, we shortly discuss Filter Sort, which is outlined in Algorithm 1. This algorithm is based on an idea that a solution that minimizes any linear combination of objectives, or even of functions that grow monotonically with an index of an objective, cannot be dominated. Furthermore, if such a function is chosen to be noticeably different from each objective, one can efficiently filter

(hence the name) the solutions that can be non-dominated assuming there is a pre-sorted list of solutions for each objective. In Filter Sort, the sum of ranks of

---

**Algorithm 1.** Filter Sort
 

---

**Require:**  $\mathbb{P} = \{\mathbb{P}_1, \mathbb{P}_2, \dots, \mathbb{P}_N\}$ : point in  $M$ -dimensional space

**Ensure:**  $\mathbb{F} = \{\mathbb{F}_1, \mathbb{F}_2, \dots\}$ : points from  $\mathbb{P}$  split into fronts

```

1: for  $j \in [1..M]$  do ▷ Phase 1: Pre-sorting
2:    $\mathbb{O}_j \leftarrow \mathbb{P}$  sorted by objective  $j$ , compared lexicographically if equal
3:   for  $i \in [1..N]$  do
4:      $\mathbb{I}_{ij} \leftarrow$  index of  $\mathbb{P}_i$  in  $\mathbb{O}_j$ 
5:   end for
6: end for
7: for  $i \in [1..N]$  do ▷ Phase 2: Finding objective statistics
8:    $\mathbb{B}_{\mathbb{P}_i} \leftarrow \arg \min_j \mathbb{I}_{ij}$  ▷ Find the best objective of  $\mathbb{P}_i$  according to its index
9:    $\mathbb{W}_{\mathbb{P}_i} \leftarrow \arg \max_j \mathbb{I}_{ij}$  ▷ Find the worst objective of  $\mathbb{P}_i$  according to its index
10:   $\mathbb{S}_{\mathbb{P}_i} \leftarrow \sum_j \mathbb{I}_{ij}$  ▷ Find the sum of objective indices
11: end for
12:  $\mathbb{T} \leftarrow \mathbb{P}$  sorted by  $\mathbb{S}$  ▷ Phase 3: Creating filters
13: for  $r \in \{1, 2, \dots\}$  do ▷ Phase 4: Actual sorting
14:   if  $|\mathbb{T}| = 0$  then
15:     break ▷ No more solutions left
16:   end if
17:    $t \leftarrow \mathbb{T}_1$  ▷ Choose filter solution with the smallest index sum
18:    $C \leftarrow \emptyset$  ▷ Candidate solutions, initially empty
19:   for  $j \in [1..M]$  do
20:      $k \leftarrow 1$ 
21:     while  $\mathbb{O}_{jk} \neq t$  do ▷ Add all solutions from  $\mathbb{O}_j$  preceding  $t$  to candidates
22:        $C \leftarrow C \cup \{\mathbb{O}_{jk}\}, k \leftarrow k + 1$ 
23:     end while
24:   end for
25:    $\mathbb{F}_r \leftarrow \mathbb{F}_r \cup \{t\}$ , remove  $t$  from  $\mathbb{T}$  and  $\mathbb{O}_j, j \in [1..M]$  ▷ Rank and remove  $t$ 
26:   for  $c \in C$  do ▷ Try each candidate for being non-dominated
27:     isDominated  $\leftarrow$  FALSE,  $k \leftarrow 1, b \leftarrow \mathbb{B}_c, w \leftarrow \mathbb{W}_c$ 
28:      $L \leftarrow \mathbb{O}_b$  ▷ Compare  $c$  with the shortest list of maybe-dominating points
29:     while  $L_k \neq c_b$  do ▷ When  $c$  is hit, the rest cannot dominate
30:       if  $(L_k)_w \leq c_w$  and  $L_k < c$  then ▷ Check the worst objective first
31:         isDominated  $\leftarrow$  TRUE, break
32:       end if
33:        $k \leftarrow k + 1$ 
34:     end while
35:     if isDominated then
36:        $C \leftarrow C \setminus \{c\}$  ▷ Remove  $c$  if it was dominated
37:     end if
38:   end for
39:   for  $c \in C$  do
40:      $\mathbb{F}_r \leftarrow \mathbb{F}_r \cup \{c\}$ , remove  $c$  from  $\mathbb{T}$  and
41:   end for
42: end for

```

---

solution's objectives is chosen as such a linear combination, which is arguably a choice that requires as few assumptions as possible.

The first three phases are rather straightforward. Phase 1 (lines 1–6 in Algorithm 1) performs sorting of the population by each of the objectives, using lexicographical sorting in the case of ties. This phase can be done in  $O(MN \log N)$  using a quicksort-like  $O(MN + N \log N)$  algorithm for lexicographical sorting, that also sorts the points by the first objective, and  $M - 1$  runs of any efficient sorting algorithm in  $O(N \log N)$ . The points sorted by the  $j$ -th objective are stored in  $\mathbb{O}_j$ . During this phase, the indices of each point  $i$  in the sorted order along each objective  $j$  are stored in  $\mathbb{I}_{ij}$ , which can easily be done from within the sorting algorithms.

Phase 2 (lines 7–11) computes, using the indices  $\mathbb{I}_{ij}$  from the previous stage, the best objective of each solution (that is, the objective, for which this point comes earlier in the corresponding list  $\mathbb{O}_j$ ), the worst objective, and the sum of objective indices. This phase is done in  $O(MN)$ . Next, Phase 3 (line 12) sorts the population according to the sum of objective indices, again in  $O(N \log N)$ , and stores the result in a list  $\mathbb{T}$ .

Note that the lists  $\mathbb{O}_j$  and  $\mathbb{T}$  subsequently require fast removal of elements from arbitrary locations. One of the possible choices is to create them as doubly linked lists, or to convert them to such lists soon after creation, for which the most efficient implementation would probably be to store the next/previous pointers in the point itself. An alternative solution would be to use auxiliary Boolean arrays that store whether a solution was deleted, and to compact the arrays and the (non-linked) lists when enough solutions are removed.

Finally, the actual non-dominated sorting happens in Phase 4 (lines 13–42). If there are any remaining solutions, the filter solution  $t$  is first chosen as the first solution in the list  $\mathbb{T}$ . As no other solutions have a smaller sum of objective ranks,  $t$  is guaranteed to be non-dominated. Then, in lines 18–24, the algorithm collects the solutions which precede  $t$  in at least one objective by joining the corresponding prefixes of all  $\mathbb{O}_j$  for each objective  $j$ , effectively filtering out all the solutions that are dominated by  $t$ . Next, the filter solution  $t$  is removed from all the lists and is added to the currently populated front  $\mathbb{F}_r$ . Finally, each of the candidates  $c$  is tested for non-dominance. For that,  $c$  is compared with all the solutions that come before  $c$  in the objective list  $\mathbb{O}_b$ , where  $b = \mathbb{B}_c$  is the best objective of  $c$  (populated in line 8 in Algorithm 1). To further speed-up the comparison, first the comparison in the worst objective of  $c$  is performed, as if  $c$  is not dominated in this objective, then it is not dominated at all. All the candidate points that passed the non-dominance checks are also added to  $\mathbb{F}_r$ , after which this front is declared complete.

### 3 Worst-Case Running Time Analysis

Now we turn to the worst-case running time analysis of Filter Sort. Our analysis consists of a nearly-trivial upper bound and a much more involved lower bound, which we state as two separate lemmas.

**Lemma 1.** *The worst-case running time of Filter Sort is  $O(MN^3)$ .*

*Proof.* This follows from the simple static analysis of Algorithm 1. Indeed, Phases 1–3 require  $O(MN \log N)$  time in common. The number of iterations of the main loop (lines 13–41) in Phase 4 coincides with the number of reported fronts, which is  $O(N)$ . In each iteration, the time spent in lines 14–25, as well as 39–41, cannot exceed  $O(MN)$ . The size of the candidate set  $C$  is at most  $N - 1$ , the number of iterations of the while-loop in lines 29–33 is at most  $N - 1$  since  $\mathbb{O}_b$  cannot contain more than  $N$  points, and the dominance comparison in line 30 cannot take more than  $O(M)$  time.

In total, each loop in lines 29–34 is at most  $O(MN)$ , each iteration in lines 26–38 is at most  $O(MN^2)$ , and the whole algorithm cannot take more than  $O(MN^3)$  time. □

We proceed with the lower bound. We first present the analysis for  $M = 2$  and then we produce a hard input for any  $M > 1$  based on this analysis.

**Lemma 2.** *There exists an input  $\mathbb{P}$  with  $N$  two-dimensional points which requires Filter Sort to run for  $\Omega(N^3)$  time.*

*Proof.* In the proof, we use the notation  $(x, y)$  to denote a two-dimensional point with objective values  $x$  and  $y$ . We assume  $N_3 = \lfloor \frac{N-1}{3} \rfloor$  and use the test consisting of three sets of points as below, depicted on Fig. 1:

- left points:  $(i, 2N_3 + i)$  for  $i = 1, \dots, N - 2N_3$ ;
- middle points:  $(N - 2N_3 + i, N_3 + i)$  for  $i = 1, \dots, N_3$ .
- right points:  $(N - N_3 + i, i)$  for  $i = 1, \dots, N_3$ ;

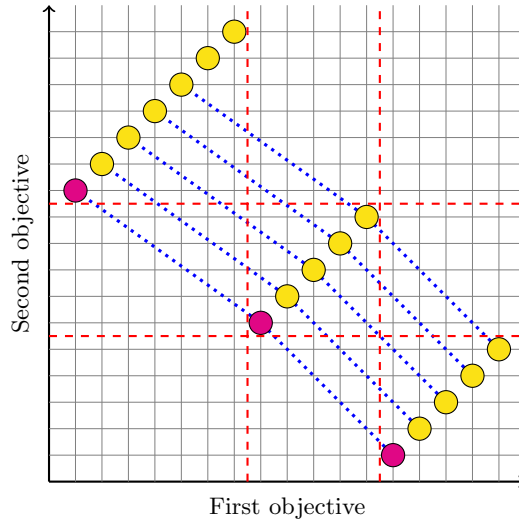
We chose  $N_3$  this way so that  $N - N_3 > 2N_3$ , that is, the number of left points is always greater than the number of middle points and of right points, which is crucial in our analysis.

Note that this test example has exactly  $N - 2N_3$  fronts, however, only the first  $N_3$  of them are the complete fronts that consist of three points each. What is more, as long as  $N \geq 4$ , when we compute and remove the first front, the remaining test would be essentially the same test for  $N' = N - 3$  points and larger gaps between the point groups, which does not influence the way Filter Sort works. This consideration makes our analysis much simpler.

The three points that compete for being a filter element are the first left point  $(1, 2N_3 + 1)$ , the first middle point  $(N - 2N_3 + 1, N_3 + 1)$  and the first right point  $(N - N_3 + 1, 1)$ , which are highlighted in Fig. 1. With our choice of objective values, the sum of ranks is the same as the sum of objectives themselves, hence the best middle and right points have these sums equal to  $N - N_3 + 2$  and the best left point has the sum equal to  $2N_3 + 2$ . As, per our choice,  $N - N_3 > 2N_3$ , the best left point is unambiguously chosen as the filter point  $t$ .

Next, Filter Sort constructs the set of candidate solutions. By our construction, every middle and every right point has the smaller second objective than  $t$ , so these points constitute the candidate set  $C$ .





**Fig. 1.** Test example for  $N = 17, M = 2$ . Points from the same front are connected with blue lines (Color figure online)

To prove that the loop at lines 26–38 requires  $\Theta(N^2)$  point comparisons, we note that the best objective of every middle point is the second objective, since the offset in the second objective is  $N_3$  and in the first objective it is  $N - 2N_3$ , which is greater. For this reason, every middle point would necessary be compared with every right point, which yields  $N_3^2 = \Theta(N^2)$  comparisons. Note that each such comparison terminates early and costs  $O(1)$ , because the worst objective of each middle point is the first objective, and in this objective every middle point is better than every right point.

As a result, sorting of the entire input of this sort would require at least

$$\sum_{i=1}^{\lfloor \frac{N-1}{3} \rfloor} i^2 = \frac{\lfloor \frac{N-1}{3} \rfloor (\lfloor \frac{N-1}{3} \rfloor + 1) (2\lfloor \frac{N-1}{3} \rfloor + 1)}{6} = \frac{N^3}{81} + O(N^2)$$

point comparisons and running time. □

**Lemma 3.** *There exists an input  $\mathbb{P}$  with  $N$  points of dimension  $M$  which requires Filter Sort to run for  $\Omega(N^3)$  time.*

*Proof.* We first construct an auxiliary set of  $N$  two-dimensional points  $\mathbb{Q}$  using the method provided in Lemma 2. Next, we define each point  $\mathbb{P}_i$  as follows:

- for  $1 \leq j \leq M - 1, \mathbb{P}_{ij} \leftarrow \mathbb{Q}_{i1}$ ;
- for  $j = M, \mathbb{P}_{ij} \leftarrow \mathbb{Q}_{i2}$ .

In this case, Filter Sort will still select the filter element from the equivalent of left points, since the objective index sum would be the smallest for such a point.

For every equivalent of a middle point, the best objective would be the last one, and the worst objective would be any objective except the last one. As a result, Filter Sort would make exactly the same choices for  $\mathbb{P}$  as it would do for  $\mathbb{Q}$ , hence it will also make  $\Omega(N^3)$  point comparisons for the input  $\mathbb{P}$ .  $\square$

Now we can formulate and prove the main theorem of the paper.

**Theorem 1.** *The worst-case running time of Filter Sort is  $\Omega(N^3)$  and  $O(MN^3)$ .*

*Proof.* The upper bound is proven in Lemma 1 and the lower bound is proven in Lemma 3.  $\square$

## 4 Conclusion and Discussion

We have proven that Filter Sort, despite the reports on its wall-clock time efficiency compared to some other algorithms, can be forced to perform  $\Omega(N^3)$  dominance comparisons, which is much worse than  $O(MN^2)$  ensured by many other algorithms.

As a result, we suggest that the authors of evolutionary multiobjective software use Filter Sort with caution (if at all) even if they like its typical performance. One recipe would be to track the number of dominance comparisons and switch to any algorithm that is less efficient in average, but has asymptotically better worst-case running time, for example, from the ENS family [35]. The availability of the non-modified algorithm that can be forced to work much slower than expected is, in fact, a security breach that can expose a DoS-attack in the case the evolutionary multiobjective software is accessible as a service.

Concerning the possible improvements of Filter Sort, we do not currently have much to propose. One of the important weaknesses is that the list  $L$  of the points which may dominate the current candidate  $c$ , as in line 28 of Algorithm 1, may contain former candidate solutions that have already been dominated by some other candidate solution. One can get rid of that by making deep copies of all the lists  $\mathbb{O}_j$  before line 26, using these copies in line 28 instead of the originals, and removing the former candidate solutions from these copies in line 36 together with the removal from the set of candidate solutions. However, just making these copies, although taking at most  $O(MN^2)$  total time, may introduce a huge overhead in typical scenarios, essentially destroying the “average” benefits of Filter Sort.

It is currently an open question whether  $\Omega(N^3)$  is the best lower bound we can prove (e.g. there is a matching  $O(N^3 + MN^2)$  bound), or our principle of constructing hard test cases is not the best one, and a strictly better lower bound holds. It appears now that tracking the worst objective and using it first to compare points is a crucial component of Filter Sort that makes it harder to propose  $\Omega(MN^3)$  tests. However, we find it difficult now to prove or disprove that the  $O(N^3 + MN^2)$  bound actually holds.

**Acknowledgment.** This research is financially supported by The Russian Science Foundation, Agreement No. 17-71-30029 with co-financing of Bank Saint Petersburg.

## References

1. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of 17th International Conference on Data Engineering, pp. 421–430. IEEE (2001)
2. Buzdalov, M.: Generalized offline orthant search: one code for many problems in multiobjective optimization. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 593–600. ACM (2018)
3. Buzdalov, M.: Make evolutionary multiobjective algorithms scale better with advanced data structures: van emde boas tree for non-dominated sorting. In: Deb, K., et al. (eds.) EMO 2019. LNCS, vol. 11411, pp. 66–77. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-12598-1\\_6](https://doi.org/10.1007/978-3-030-12598-1_6)
4. 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)
5. Coello 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)
6. Corne, D.W., Knowles, J.D., Oates, M.J.: The pareto envelope-based selection algorithm for multiobjective optimization. In: Schoenauer, M., et al. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 839–848. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45356-3\\_82](https://doi.org/10.1007/3-540-45356-3_82)
7. Deb, K., Hussein, R., Roy, P., Toscano, G.: Classifying metamodeling methods for evolutionary multi-objective optimization: first results. In: Trautmann, H., et al. (eds.) EMO 2017. LNCS, vol. 10173, pp. 160–175. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-54157-0\\_12](https://doi.org/10.1007/978-3-319-54157-0_12)
8. 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)
9. 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)
10. Gustavsson, P., Syberfeldt, A.: A new algorithm using the non-dominated tree to improve non-dominated sorting. *Evol. Comput.* **26**(1), 89–116 (2018)
11. Handl, J., Knowles, J.: Exploiting the trade-off — the benefits of multiple objectives in data clustering. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 547–560. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-31880-4\\_38](https://doi.org/10.1007/978-3-540-31880-4_38)
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., Corne, D.: M-PAES: a memetic algorithm for multiobjective optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, vol. 1, pp. 325–332. IEEE (2000)
14. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. *Evol. Comput.* **8**(2), 149–172 (2000)
15. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. *J. ACM* **22**(4), 469–476 (1975)
16. Leyton-Brown, K., Shoham, Y.: Essentials of game theory: a concise multidisciplinary introduction. *Synthesis Lect. Artif. Intell. Mach. Learn.* **2**(1), 1–88 (2008)

17. Lou, R.D., Sarrafzadeh, M.: An optimal algorithm for the maximum three-chain problem. *SIAM J. Comput.* **22**(5), 976–993 (1993)
18. Markina, M., Buzdalov, M.: Hybridizing non-dominated sorting algorithms: divide-and-conquer meets best order sort. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 153–154. ACM (2017)
19. Markina, M., Buzdalov, M.: Towards large-scale multiobjective optimisation with a hybrid algorithm for non-dominated sorting. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) *PPSN 2018. LNCS*, vol. 11101, pp. 347–358. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99253-2\\_28](https://doi.org/10.1007/978-3-319-99253-2_28)
20. McClymont, K., Keedwell, E.: Deductive sort and climbing sort: new methods for non-dominated sorting. *Evol. Comput.* **20**(1), 1–26 (2012)
21. Mishra, S., Buzdalov, M.: If unsure, shuffle: deductive sort is  $\Theta(MN^3)$ , but  $O(MN^2)$  in expectation over input permutations. In: *Proceedings of Genetic and Evolutionary Computation Conference*. ACM (2020). <https://doi.org/10.1145/3377930.3390246>. Accepted for publication
22. Mishra, S., Buzdalov, M., Senwar, R.: Time complexity analysis of the dominance degree approach for non-dominated sorting. In: *Proceedings of Genetic and Evolutionary Computation Conference Companion*. ACM (2020). <https://doi.org/10.1145/3377929.3389900>. Accepted for publication
23. Mishra, S., Mondal, S., Saha, S., Coello Coello, C.A.: GBOS: generalized best order sort algorithm for non-dominated sorting. *Swarm Evol. Comput.* **43**, 244–264 (2018)
24. Mishra, S., Saha, S., Mondal, S.: Divide and conquer based non-dominated sorting for parallel environment. In: *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 4297–4304. IEEE (2016)
25. Mishra, S., Saha, S., Mondal, S.: MBOS: modified best order sort algorithm for performing non-dominated sorting. In: *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 725–732. IEEE (2018)
26. Mishra, S., Saha, S., Mondal, S., Coello Coello, C.A.: A divide-and-conquer based efficient non-dominated sorting approach. *Swarm Evol. Comput.* **44**, 748–773 (2019)
27. Moreno, J., Rodriguez, D., Nebro, A.J., Lozano, J.A.: Merge nondominated sorting algorithm for many-objective optimization. *IEEE Trans. Cybern.* (2020). <https://doi.org/10.1109/TCYB.2020.2968301>. Accepted for publication
28. Roy, P., Hussein, R., Deb, K.: Metamodeling for multimodal selection functions in evolutionary multi-objective optimization. In: *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 625–632. ACM (2017)
29. Roy, P.C., Deb, K.: High dimensional model representation for solving expensive multi-objective optimization problems. In: *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2490–2497. IEEE (2016)
30. Roy, P.C., Deb, K., Islam, M.M.: An efficient nondominated sorting algorithm for large number of fronts. *IEEE Trans. Cybern.* **49**(3), 859–869 (2019)
31. 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. ACM (2016)
32. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.* **2**(3), 221–248 (1994)

33. Tang, S., Cai, Z., Zheng, J.: A fast method of constructing the non-dominated set: arena's principle. In: 4th International Conference on Natural Computation, pp. 391–395. IEEE (2008)
34. Wang, J., Li, C., Diao, Y., Zeng, S., Wang, H.: An efficient nondominated sorting algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 203–204. ACM (2018)
35. 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)
36. Zhang, X., Tian, Y., Cheng, R., Jin, Y.: A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Trans. Evol. Comput.* **22**(1), 97–112 (2018)
37. Zhang, X., Tian, Y., Cheng, R., Yaochu, J.: An efficient approach to nondominated sorting for evolutionary multiobjective optimization. *IEEE Trans. Evol. Comput.* **19**(2), 201–213 (2015)
38. 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)
39. Zhou, Y., Chen, Z., Zhang, J.: Ranking vectors by means of the dominance degree matrix. *IEEE Trans. Evol. Comput.* **21**(1), 34–51 (2017)



# Approximation Speed-Up by Quadratzation on LeadingOnes

Andrew M. Sutton<sup>1</sup>(✉) and Darrell Whitley<sup>2</sup>

<sup>1</sup> Department of Computer Science, University of Minnesota Duluth, Duluth, USA  
amsutton@umn.edu

<sup>2</sup> Department of Computer Science, Colorado State University, Fort Collins, USA  
whitley@cs.colostate.edu

**Abstract.** We investigate the quadratzation of LEADINGONES in the context of the landscape for local search. We prove that a standard quadratzation (i.e., its expression as a degree-2 multilinear polynomial) of LEADINGONES transforms the search space for local search in such a way that faster progress can be made. In particular, we prove there is a  $\Omega(n/\log n)$  speed-up for constant-factor approximations by RLS when using the quadratzated version of the function. This suggests that well-known transformations for classical pseudo-Boolean optimization might have an interesting impact on search heuristics. We derive and present numerical results that investigate the difference in correlation structure between the untransformed landscape and its quadratzation. Finally, we report experiments that provide a detailed glimpse into the convergence properties on the quadratzated function.

## 1 Introduction

The transformation of higher order pseudo-Boolean functions into quadratic functions has been studied in the context of classical mathematical optimization [2]. Such transformations are useful because they allow for faster exact maximization techniques. However, in the context of evolutionary optimization and local search, it is not immediately clear that such transformations could be beneficial. Indeed, it seems likely that an arbitrary transformation of a pseudo-Boolean function could be detrimental to local search methods by introducing auxiliary variables with uncontrolled dependencies and obscuring the “fitness” signal within the search landscape. This raises a question as to whether quadratic transformations could actually be beneficial to local search. This paper answers that question in the affirmative. In particular, we show that a standard transformation of the LEADINGONES function to a quadratic form yields a  $\Omega(n/\log n)$  speed-up for constant factor approximations by RLS. Moreover, we show that instead of obscuring the fitness signal, the transform supplies a more favorable correlation structure to the search landscape.

Let  $f: \{0, 1\}^n \rightarrow \mathbb{R}$  be a pseudo-Boolean function. Then  $f$  has a unique multilinear form

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{j \in S} x_j,$$

where  $c_S$  is a real coefficient. We refer to  $c_S \prod_{j \in S} x_j$  as the *monomial* corresponding to  $S$ . The *degree* of  $f$  is the maximum cardinality of  $S$  such that the coefficient  $c_S$  of the monomial corresponding to  $S$  is nonzero. In particular,

$$\text{deg}(f) = \max_{S \subseteq [n]} \{|S| : c_S \neq 0\}$$

For an arbitrary pseudo-Boolean function  $f: \{0, 1\}^n \rightarrow \mathbb{R}$ , a *quadratzation* [1] of  $f$  is a quadratic function  $g(x, y)$  where  $x \in \{0, 1\}^n$  and  $y \in \{0, 1\}^m$  for some  $m = \text{poly}(n)$  such that

$$f(x) = \max_{y \in \{0, 1\}^m} \{g(x, y)\}.$$

Here we refer to the  $m$  additional  $y_i$ -variables as *auxiliary* variables. Writing  $f$  as a maximum over auxiliary variables is useful in the context of maximization. In particular, if we seek to maximize  $f$ , we have the correspondence

$$\max_{x \in \{0, 1\}^n} f(x) = \max\{g(x; y) : x \in \{0, 1\}^n, y \in \{0, 1\}^n\}.$$

In order to construct  $g$ , we compute a quadratzation for each monomial in  $f$  with degree at least three by using the following observations.

A positive monomial (i.e.,  $c_S > 0$ ) can be written as

$$c_S \prod_{j \in S} x_j = c_S \max_{y \in \{0, 1\}} \left\{ y \left( \sum_{j \in S} x_j - (|S| - 1) \right) \right\}, \tag{1}$$

where  $y$  is a new auxiliary variable [6].

### 1.1 LeadingOnes

The well-known LEADINGONES [4, 8] function counts the number of one bits appearing as a prefix in a bit string before the first zero. LEADINGONES is defined as follows.

$$f(x) = \sum_{i=1}^n \prod_{j=1}^i x_j = x_1 + x_1x_2 + \sum_{i=3}^n \prod_{j=1}^i x_j$$

Hence there are exactly  $n$  positive monomials with unit coefficients. Applying (1) to each positive monomial of degree greater than two, we arrive at the quadratzation of LEADINGONES  $f(x) = \max\{g(x, y) : y \in \{0, 1\}^{n-2}\}$  where

$$g(x, y) = x_1 + x_1x_2 + \sum_{i=3}^n \sum_{j=1}^i x_j y_{i-2} - \sum_{i=3}^n (i-1)y_{i-2}. \tag{2}$$

The maximum of  $g$  is  $g(1^n, 1^{n-2}) = n$ . The minimum of  $g$  lies at  $g(0^n, 1^{n-2}) = -(n^2 - n - 2)/2$ .

## 2 Quadratzation Can Improve Approximation Speed

It is known that the quadratzation of general pseudo-Boolean functions can render them easier to solve by classical computational approaches [2]. In this section, we will show that in the case of LEADINGONES, the quadratzation can also improve the search space for local search algorithms. In particular, the quadratzation changes the landscape to permit “shortcuts” that speed up the time to reach higher quality solutions.

One of the simplest forms of local search is so-called *random local search* (RLS) in which we iteratively hillclimb in the space of bitstrings to try to locate an optimal solution. The traditional RLS algorithm is listed in Algorithm 1.

---

**Algorithm 1:** Classical Random Local Search to maximize  $f$

---

**input** : A function  $f: \{0, 1\}^n \rightarrow \mathbb{R}$   
**output**: A proposed maximum to  $f$

- 1 Choose  $x \in \{0, 1\}^n$  uniformly at random;
- 2 **while** *termination criteria not met* **do**
- 3     Create  $x'$  by flipping exactly one of the  $n$  bits of  $x$ , chosen u.a.r.;
- 4     **if**  $f(x') \geq f(x)$  **then**  $x \leftarrow x'$
- 5 **return**  $x$ ;

---

We adapt local search on the quadratzation of LEADINGONES as follows. We employ the foregoing quadratzation transformation  $g(x, y)$  as the evaluation function for local search over  $(x, y) \in \{0, 1\}^n \times \{0, 1\}^{n-2}$ , where  $g$  is to be maximized. After iteratively hillclimbing on  $g$  for a prescribed number of steps, we obtain a proposed solution. We then interpret the first argument of  $g$  as the proposed solution for  $f$ . The general algorithm for RLS using a quadratzation as a surrogate is listed in Algorithm 2.

---

**Algorithm 2:** Random Local Search to maximize  $f$  using a quadratzation  $g$

---

**input** : A function  $f: \{0, 1\}^n \rightarrow \mathbb{R}$  and its quadratzation  
 $g: \{0, 1\}^n \times \{0, 1\}^m \rightarrow \mathbb{R}$   
**output**: A proposed maximum to  $f$

- 1 Choose  $x \in \{0, 1\}^n, y \in \{0, 1\}^m$  uniformly at random;
- 2 **while** *termination criteria not met* **do**
- 3     Create  $(x', y')$  by flipping exactly one of the  $n + m$  bits in  $(x, y)$ , chosen u.a.r.;
- 4     **if**  $g(x', y') \geq g(x, y)$  **then**  $(x, y) \leftarrow (x', y')$
- 5 **return**  $x$ ;

---

We begin by stating the claim that RLS requires  $\Omega(n^2)$  steps to find any constant factor approximation for LEADINGONES with high probability.



**Theorem 1.** *Let  $0 < \rho < 1$  be an arbitrary constant. With probability  $1 - o(1)$ , Random Local Search (Algorithm 1) requires  $\Omega(n^2)$  iterations until it generates a solution  $x$  with at least  $\rho n$  leading ones.*

The proof of Theorem 1 follows easily from the fact that the time to find a string in  $\{0, 1\}^n$  with at least  $\rho n$  leading ones (for any constant  $0 < \rho < 1$ ) is no faster than the time to solve LEADINGONES on  $\{0, 1\}^{\lfloor \rho n \rfloor}$ . The tail bound follows by adapting the argument in [5, Theorem 17].

Our main result is that the transformed search space allows for the following probabilistic performance guarantee.

**Theorem 2.** *Let  $0 < \rho < 1$  be an arbitrary constant. With probability  $\Omega(1)$ , Random Local Search (Algorithm 2) using the quadratzation  $g$  of LEADINGONES requires  $\Theta(n \log n)$  iterations until it generates a solution  $x$  with at least  $\rho n$  leading ones.*

Before proving Theorem 2, it will be useful to prove the following two technical lemmas that shed light on the properties of the transformed search space.

**Lemma 1.** *Let  $x \in \{0, 1\}^n$  and  $y \in \{0, 1\}^{n-2}$ . For arbitrary  $k \leq 2 + \max\{i : y_i = 1\}$ , let  $x'$  be the Hamming neighbor of  $x$  produced by flipping the  $k$ -th bit of  $x$ . Then  $g(x, y) < g(x', y) \iff x_k < x'_k$ , where  $g$  is the quadratzation of LEADINGONES.*

*Proof.* By Eq. (2),  $g(x, y) - g(x', y) = (x_k - x'_k) \sum_{i=k}^n y_{i-2}$ . Since we assume  $k \leq 2 + \max\{i : y_i = 1\}$ , it holds that

$$\sum_{i=k}^n y_{i-2} > 0,$$

and thus the claim holds. □

**Lemma 2.** *Let  $x \in \{0, 1\}^n$  and  $y \in \{0, 1\}^{n-2}$ . For arbitrary  $k \in \{1, \dots, n - 2\}$ , let  $y'$  be the Hamming neighbor of  $y$  produced by flipping the  $k$ -th bit of  $y$ . Then the following properties hold.*

1. *If  $\text{LEADINGONES}(x) \geq k + 2$ , then  $g(x, y) < g(x, y') \iff y_k < y'_k$ .*
2. *If  $\text{LEADINGONES}(x) = k + 1$ , or more generally, there is at most one index  $1 \leq i \leq k + 2$  such that  $x_i = 0$ , then  $g(x, y) = g(x, y')$ .*
3. *Otherwise,  $g(x, y) < g(x, y') \iff y_k > y'_k$ .*

*Proof.* By Eq. (2), we have

$$g(x, y) - g(x, y') = \sum_{j=1}^{k+2} x_j (y_k - y'_k) - (y_k - y'_k)(k + 1). \tag{3}$$

Suppose first that  $\text{LEADINGONES}(x) \geq k + 2$ . Then property 1. must hold, since Eq. (3) yields  $g(x, y) - g(x, y') = (y_k - y'_k)(k + 2 - (k + 1)) = (y_k - y'_k)$ .

Similarly, when there is only a single zero in  $x$  between indexes 1 and  $k + 2$ , then  $\sum_{j=1}^{k+2} x_j = k + 1$  and Eq. (3) is zero, yielding property 2. Otherwise,  $g(x, y) - g(x, y') = (y_k - y'_k)(a - (k + 1))$  for some  $a \leq k$ , providing property 3.  $\square$

An interesting effect of the quadratization is that at time  $t$ , the substring  $x_1, \dots, x_{\ell(t)}$  where  $\ell(t)$  is two plus the maximum index of  $y$  set to one at time  $t$ , essentially “looks like” a positive linear function to RLS. Thus, as long as high bits of  $y$  remain set to one, a large prefix of  $x$  can be quickly optimized. We then rely on Lemmas 1 and 2 to show that the correct combination of  $x$  and  $y$  bits set to one ensure that a constant fraction of leading ones in  $x$  are protected from switching to zero for the remainder of the process.

*Proof (of Theorem 2).* Fix a constant  $0 < \rho < 1$ . We are interested in the number of iterations of Algorithm 2 until  $\text{LEADINGONES}(x) \geq \rho n$ .

We begin by proving the upper bound of  $O(n \log n)$ . Let  $\rho' = (1/2 + \rho/2)$ . We define the random variable  $T$  to be the first iteration in which  $\max\{i : y_i = 1\} < \rho'n$ . Since we choose the initial string at random, by Chernoff bounds, with probability exponentially close to one,

$$z := \sum_{i=\rho'n}^{n-2} y_i > \frac{(1 - \rho')n}{4}.$$

By Lemma 2, as long as  $\text{LEADINGONES}(x) < \rho'n$ , any such  $y_i$  changing from 1 to 0 is accepted, while any  $y_i$  changing from 0 to 1 may be rejected, as the resulting fitness is either smaller or equal. Either of these flips happen with probability  $\frac{1}{2(n-1)}$ . By pessimistically assuming that no zeros are changed to one before  $\max\{i : y_i = 1\} < \rho'n$  (which could only slow the process down),  $T$  is probabilistically bounded below by the hitting time of a coupon-collector process from which we can derive the following tail bound via the Chebyshev inequality

$$\Pr(T \geq 2(n - 1)H_z - 2(n - 1) \ln 3) > 1 - \frac{1}{\ln^2 3}.$$

where  $H_z$  denotes the  $z$ -th Harmonic number. For the remainder of the proof we condition on the event that  $T \geq 2(n - 1)H_z - 2(n - 1) \ln 3$ .

By Lemma 1, for all  $t < T$ , changing an  $x_i$  from 0 to 1 with  $i \leq \rho'n$  is always accepted. Similarly, changing an  $x_i$  from 1 to 0 with  $i \leq \rho'n$  is always rejected. Consider a contiguous block of size  $z/10$  in  $x$ . Let  $S$  be the waiting time until the block is solved. As long as the block is solved before  $T$  steps,  $S$  is again the hitting time of a coupon-collector process with  $E[S] = 2(n - 1)H_{z/10}$ . We thus have,

$$\Pr(S > 2(n - 1)H_{z/10} + 2(n - 1) \ln 3) < \frac{1}{\ln^2 3}.$$

The probability that  $r$  independent contiguous blocks of size  $z/10$  are all solved before  $2(n - 1)H_{z/10} + 2(n - 1) \ln 3$  is at least  $(1 - \frac{1}{\ln^2 3})^r$ . Setting

$r = 10\rho'n/z < 40\rho'/(1 - \rho')$ , we see that the first  $r$  blocks would contribute  $\rho'n$  to the LEADINGONES value of  $x$ . Note that, by assumption,

$$\begin{aligned} T - (2(n - 1)H_{z/10} + 2(n - 1)\ln 3) &\geq 2(n - 1)(H_z - H_{z/10} - 2\ln 3) \\ &= 2(n - 1)\left(\ln \frac{10}{9} - O(1/z)\right) \end{aligned}$$

is positive for sufficiently large  $z$ . Hence with probability  $\Omega(1)$ , there is a point in time before  $T$  iterations when  $\text{LEADINGONES}(x) \geq \rho'n$

We now condition on this occurrence and assume the process has generated at least  $\rho'n$  leading ones in  $x$  at some time  $T' \leq T$ . We claim that in iteration  $T' + 1$ , with at least constant probability, there is an index  $k \in [\rho n, \rho'n - 2]$  such that  $y_k = 1$ . If there is no such bit  $y_k = 1$  in iteration  $T'$ , then one of the bits of  $y$  in the index range  $[\rho n, \rho'n - 2]$  flips from zero to one with probability

$$\frac{\rho'n - \rho n - 2}{2(n - 1)} = (1 - \rho)/4 - O(1/n) = \Omega(1).$$

Conditioning on this event, by Lemma 1, as long as  $y_k = 1$ , we have  $2 + \max\{i : y_i = 1\} > k + 2$ , and so none of the one bits in  $x$  with index at most  $k$  will be lost in the next step. Furthermore, by Lemma 2, since  $\text{LEADINGONES}(x) \geq k + 2$ , any change of  $y_i$  with  $i \leq k$  from one to zero is not accepted, and any change of  $y_i$  with  $i \leq k$  from zero to one is accepted.

As long as these two constant-probability events have occurred, then at time  $T' + 1$  there at least  $k \geq \rho n$  leading ones in  $x$ , and by induction, this condition is maintained in every step beyond  $T'$ , and so the  $k \geq \rho n$  leading ones of  $x$  are never lost for the remainder of the search process.

To prove the lower bound, note that we must collect  $\rho n$  leading ones in  $x$ . With high probability, at least  $\rho n/4$  bits of  $x$  are zero at initialization. Set  $t = (2n - 3)\ln n$ , and note that the probability that after  $t$  steps, a particular one of these zero bits has not been flipped is at least

$$\left(1 - \frac{1}{2(n - 1)}\right)^t = \left(1 - \frac{1}{2(n - 1)}\right)^{(2n-3)\ln n} \geq 1/n.$$

The probability that at time  $t$  there is still a zero bit in the first  $\rho n$  bits of  $x$  is thus at least  $1 - (1 - 1/n)^{\rho n/4} = \Omega(1)$ . Thus with at least constant probability, Algorithm 2 needs at least  $\Omega(n \log n)$  steps to reach a  $\rho$ -approximation for  $x$ .  $\square$

The proof of Theorem 2 relies on the fact that enough leading bits of  $x$  become fixed to one before too many bits in  $y$  are changed to zero. We can also translate this to an exact result on a somewhat more ad-hoc function. Define  $\text{HALFONEMAX}(x) := x \rightarrow \sum_{i=\lfloor n/2 \rfloor}^n x_i$ . We can adapt the proof of Theorem 2 to prove the following.

**Theorem 3.** *Algorithm 2 solves  $\text{LEADINGONES}(x) + \text{HALFONEMAX}(x)$  to optimality in  $\Theta(n \log n)$  with constant probability.*

*Proof.* The quadratization of LEADINGONES+HALFONEMAX is

$$g'(x, y) := \sum_{i=\lfloor n/2 \rfloor}^n x_i + g(x, y),$$

where  $g$  is the quadratization of LEADINGONES. In this case, for every  $k \geq n/2$ , flipping  $x_k$  from a zero to a one is always improving, since the flip is counted at least once in the quadratization  $g'$ . For the same reason, any such  $x_k = 1$  is never switched to zero. Therefore, for any positive constant  $\varepsilon$ , the solution to HALFONEMAX is found in at most  $2(n-1) \ln n + \varepsilon 2(n-1)$  steps with probability at least  $1 - e^{-\varepsilon}$  [3, Theorem 1.9.2]. Favoring the rightmost  $\lfloor n/2 \rfloor$  bits of  $x$  does not slow the process for optimizing  $g(x, y)$ , and we apply Theorem 2 to show that a 1/2-approximation of LEADINGONES is attained in  $\Theta(n \log n)$  steps with constant probability. Since this approximation covers the leftmost  $\lceil n/2 \rceil$  bits of  $x$ , the proof is complete.  $\square$

On the other hand, an adaptation of the proof of Theorem 1 establishes that using the standard non quadratized version of this function would require  $\Omega(n^2)$  steps, as it must also solve the LEADINGONES component of size  $\lceil n/2 \rceil$ .

### 3 Experiments

We perform an empirical supplement to the above results to (1) understand numerically the difference in correlation structure between the two search landscapes, and (2) investigate the difference in convergence on a generalization of LEADINGONES.

#### 3.1 Random Walk Autocorrelation

A common measure of the “smoothness” of a search space is the *random walk autocorrelation* for the fitness landscape. This together with the *correlation length* of the fitness landscape characterizes how well fitness values are correlated in the local neighborhood of local search algorithms and evolutionary mutation operators [9]. In this section we derive a method to compute numerically the exact random walk autocorrelation function and correlation length for LEADINGONES and its quadratization introduced in Sect. 2. The random walk autocorrelation function  $r(s)$  is the statistical autocorrelation along a random walk of length  $s$ , whereas the correlation length measures how far along a random walk in the search neighborhood fitnesses tend to be correlated. A larger correlation length corresponds to a search space that is “smoother” on average, as fitnesses tend to be correlated at longer distances. Stadler and Schnabl have even conjectured [10] that the correlation length is intimately related to the number of local optima in the landscape.

On pseudo-Boolean functions, exact statistical quantities describing the landscape can often be extracted using the well-known Walsh transform. Let the function  $S_n : \{0, 1, \dots, n\} \rightarrow 2^{\{0,1,\dots,n\}}$  be defined as

$$S_n(i) := \left\{ b : \left\lfloor \frac{i}{2^b} \right\rfloor \equiv 1 \pmod{2} \right\}$$

that determines which bits are set in the length- $n$  binary representation of  $i$ . When  $n$  is clear from context, we omit the subscript for simplicity. Every pseudo-Boolean function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$  can be represented in the Walsh polynomial basis

$$f(x) = \sum_{i=0}^{2^k-1} w_i \psi_i(x), \quad \text{where } \psi_i(x) = \prod_{j \in S(i)} (-1)^{x_j},$$

and  $w_i$  is a real-valued Walsh coefficient. We define  $|S(i)|$  to be the *order* of the coefficient. The coefficients  $w_i$  are recovered by the Walsh transform [7],

$$w_i = \sum_{x \in \{0,1\}^n} f(x) \psi_i(x). \tag{4}$$

The random walk autocorrelation  $r(s)$  and correlation length  $\ell$  on a landscape can be computed as linear combinations of the ratio of squared Walsh coefficients of each order [11].

$$r(s) := \sum_{p \neq 0} W^{(p)} \left( 1 - \frac{2p}{n} \right)^s, \tag{5}$$

and

$$\ell := n \sum_{p \neq 0} \frac{W^{(p)}}{2^p}, \tag{6}$$

where  $W^{(p)} := \left( \sum_{i: |S(i)|=p} w_i^2 \right) / \left( \sum_{j \neq 0} w_j^2 \right)$  are the normalized order- $p$  amplitudes of the decomposition.

The Walsh transform of a degree- $k$  monomial is

$$\prod_{j=1}^k x_j = \sum_{i=0}^{2^k-1} w_i^{(k)} \psi_i(x),$$

where, by Eq. (4),

$$w_i^{(k)} = \begin{cases} 2^n - 1 & \text{if } i = 0, \\ \psi_i(i) 2^{n-k} & \text{if } S(i) \supseteq \{1, \dots, k\}, \\ 0 & \text{otherwise.} \end{cases}$$

The transform is linear, and hence the transform of LEADINGONES is computed over the monomials  $w_i = \sum_{k=1}^n w_i^{(k)}$ . Collecting the squared Walsh coefficients of order  $p$ , we have

$$\sum_{i:|S(i)|=p} w_i^2 = \binom{k-1}{p-1} (2^{n-k+1} - 1)^2,$$

and we derive the normalized order- $p$  amplitudes for LEADINGONES as follows.

$$W^{(p)} = \frac{\sum_{k=p}^n \sum_{i:|S(i)|=p} w_i^2}{\sum_{j \neq 0} w_j^2} = \frac{1}{s} \sum_{k=p}^n \binom{k-1}{p-1} (2^{n-k+1} - 1)^2, \tag{7}$$

and  $s = \sum_{j \neq 0} w_j^2$ . Note that each  $W^{(p)}$  can be computed as a sum over  $n - p + 1$  terms.

We can also compute the exact correlation for the quadratization in (2). For LEADINGONES on  $n$  bits, we consider the quadratization  $g$  as a function on  $N = 2n - 2$  bits. Again, the Walsh transform is linear, so we can compute the coefficients separately for each term in  $g$ . In particular, let

$$h_1(z) = \sum_{i=3}^n \sum_{j=1}^i z_j z_{n-i-2}, \quad \text{and} \quad h_2 = \sum_{i=3}^n (i-1) z_{n+i-2}.$$

The function  $h_2$  is a simple linear function, and the Walsh coefficients are

$$w_i^{h_2} = \begin{cases} 2^{n-3} \frac{(n+1)(n-2)}{2} & \text{if } i = 0, \\ -(k+1)2^{2n-3} & \text{if } S(i) = \{n+k\}, k > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The function  $h_1$  is quadratic, but the Walsh coefficients are straightforward to extract.

$$w_i^{h_1} = \begin{cases} 2^{2n-4} (n-2)(n+3)/2 & \text{if } i = 0, \\ 2^{2n-4} & \text{if } S(i) = \{b_1, b_2\} \text{ where } 3 < b_1 \leq n < b_2, \\ -(b-n+2)2^{2n-4} & \text{if } S(i) = \{b\} \text{ and } b > n \\ -(n-b+1)2^{2n-4} & \text{if } S(i) = \{b\} \text{ and } 3 < b \leq n \\ -(n-2)2^{2n-4} & \text{if } S(i) = \{b\} \text{ and } b \leq 3 \\ 0 & \text{otherwise.} \end{cases}$$

Combining the above with the Walsh transform for monomials  $x_1$  and  $x_1x_2$ , we compute the set of Walsh coefficients for  $g$  as  $w_i^g = w_i^{(1)} + w_i^{(2)} + w_i^{h_1} - w_i^{h_2}$ .

The squared linear coefficients simplify considerably:

$$\begin{aligned} \sum_{i:|S(i)|=1} w_i^2 &= (-2^{2n-4} - (n-2)2^{2n-4})^2 + (-3 \cdot 2^{2n-4} - (n-2)2^{2n-4})^2 \\ &\quad + ((n-2)2^{2n-4})^2 + \sum_{b=4}^n (-n-b+1)2^{2n-4})^2 \\ &\quad + \sum_{b=n+1}^{2n-2} (-(i-n+2)2^{2n-4} + (i-n+1)2^{2n-3})^2 \\ &= \frac{n(2n^2 - 3n + 13) 2^{4n-8}}{3}. \end{aligned}$$

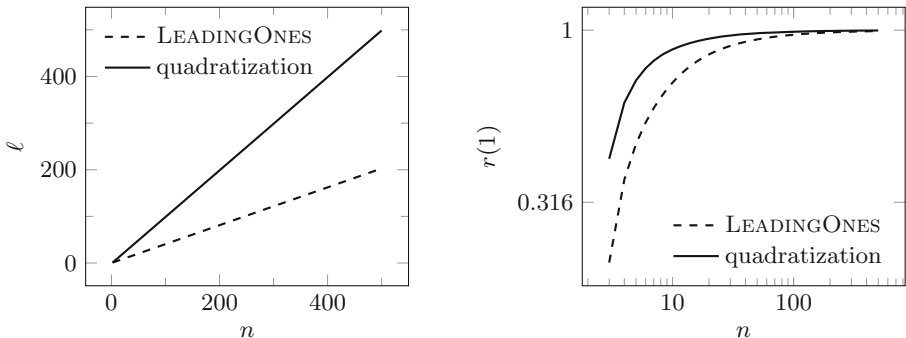
The squared quadratic coefficients can be computed as

$$\sum_{i:|S(i)|=2} w_i^2 = \left( \frac{(n-1)(n+2)}{2} - 1 \right) 2^{4n-8}.$$

Since  $g$  is quadratic,  $w_i = 0$  for all  $i$  with  $|S(i)| > 2$ . Thus we have

$$W^{(1)} = \frac{2n(2n^2 - 3n + 13)}{4n^3 - 3n^2 + 29n - 12}, \quad \text{and} \quad W^{(2)} = \frac{3(n^2 + n - 4)}{4n^3 - 3n^2 + 29n - 12}. \tag{8}$$

Substituting the order- $p$  amplitudes derived in Eq. (7) for LEADINGONES and Eq. (8) for the quadratzation into the formulas for random walk autocorrelation and correlation length (Eqs. (5) and (6)), it is possible to derive numerically these exact quantities, even for large values of  $n$ . Note that in general, one would either require exhaustive enumeration of the search space to obtain the exact correlation structure, or would resort to sampling. We compare the smoothness of LEADINGONES to its quadratzation in Fig. 1.



**Fig. 1.** Exact correlation length  $\ell$  (left) and exact 1-step correlation  $r(1)$  as a function of  $n$  on LEADINGONES and its quadratzation.

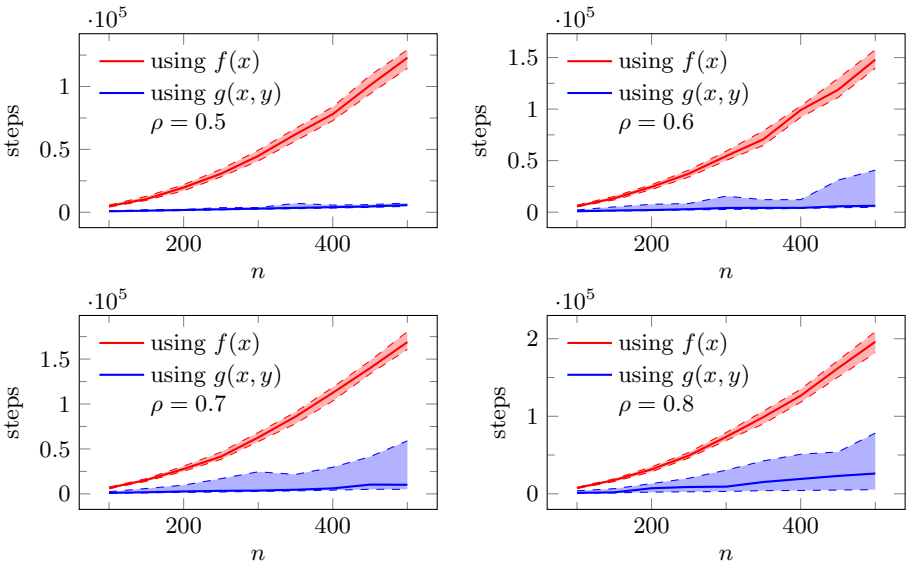
### 3.2 Leading Ones and Sparse Permutation Problems

A generalization of the LEADINGONES problem is the HIDDENPERMUTATION problem. Here we have some permutation  $\pi: \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ , and we want to optimize the function

$$f(x) = \sum_{i=1}^n c_i \prod_{j=1}^i x_{\pi(j)}. \tag{9}$$

Thus the LEADINGONES function is a special case of HIDDENPERMUTATION when  $\pi$  is the identity permutation, and  $c_i = 1$  for all  $i \in \{1, \dots, n\}$ .

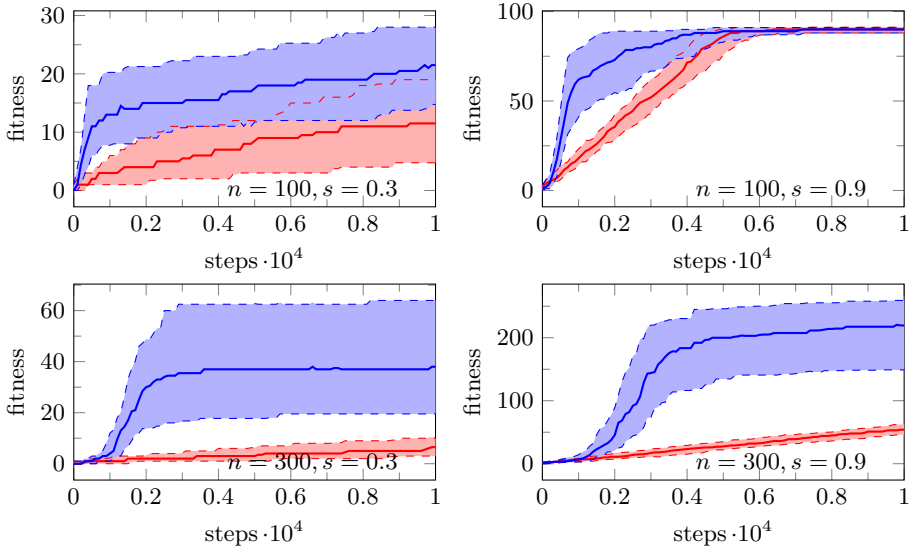
To investigate the tightness of the bounds proved in Sect. 2, and observe the details of the convergence speed for RLS, we perform a number of runs of local search on LEADINGONES and measure the number of steps necessary until a particular approximation factor  $\rho$  is reached. For each  $n \in \{100, 150, \dots, 450, 500\}$  we ran RLS both with LEADINGONES as the evaluation function, and with the quadratization as the surrogate evaluation function. The search was terminated as soon as the true LEADINGONES value reached the target approximation ratio  $\rho \in \{0.5, 0.6, 0.7, 0.9\}$ . For each  $(n, \rho)$  pair, we conducted 100 trials of each search variant. The median number of steps along with interquartile ranges are plotted in Fig. 2. The results suggest the bounds in Sect. 2 are in fact tight.



**Fig. 2.** For each  $\rho = \{0.5, 0.6, 0.7, 0.8\}$ , the plot reports the median number of steps required as a function of  $n$  by RLS until at least  $\rho n$  leading ones are found in  $x$  using either  $f$  (LEADINGONES) directly as an evaluation function, or its quadratization  $g$ . Shaded area denotes interquartile range.



To examine the convergence behavior, and to investigate the generality of our results, we performed a number of experiments in which we generate several random HIDDENPERMUTATION problem instances and sampled the fitness during the search. A problem instance is generated as follows. A permutation  $\pi$  is first drawn uniformly at random, and then the sequence of coefficients  $(c_1, c_2, \dots, c_n)$  is chosen from  $c_i \in \{0, 1\}$ . We parameterize each set of instances with the sparsity parameter  $s$ , and choose  $c_i = 1$  with probability  $s$ , and  $c_i = 0$  with probability  $1 - s$ . For  $n = 100, 200, 300$  and  $s = 0.3, 0.9$ , we generated 100 problems for each parameter combination. For each problem instance, we ran RLS for 10000 steps and sampled the fitness in each step by evaluating the polynomial in Eq. (9). We also ran RLS using the quadratzation of the polynomial as a surrogate evaluation function, but sampled the *true* fitness in each step again by evaluating the polynomial in Eq. (9). The results from these experiments (excluding  $n = 200$ ) are plotted in Fig. 3. On each set, we can observe is a distinct advantage by using the quadratzation.



**Fig. 3.** Median true fitness (Eq. (9)) as a function of local search steps for sparse hidden permutation problems. Search with unmodified function represented by the red line. Search with quadratzation as surrogate evaluation represented by the blue line. Shaded area represents interquartile range.

## 4 Conclusion

We investigated the quadratzation of LEADINGONES in the context of local search behavior. We proved that the quadratzation transforms the search space

in such a way that speeds up local search to find any constant factor approximation. We also derived exact expressions for the random walk autocorrelation on both LEADINGONES and its quadratization, suggesting that the transformation improves the amenability of the landscape to local search algorithms. We conducted experiments to observe the details of the speed-up on both LEADINGONES and the more general class of sparse HIDDENPERMUTATION problems.

## References

1. Boros, E., Gruber, A.: On quadratization of pseudo-Boolean functions. In: International Symposium on Artificial Intelligence and Mathematics (2012)
2. Boros, E., Hammer, P.L.: Pseudo-boolean optimization. *Discrete Appl. Math.* **123**(1–3), 155–225 (2002)
3. Doerr, B.: Probabilistic tools for the analysis of randomized optimization heuristics. *Theory of Evolutionary Computation*. NCS, pp. 1–87. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-29414-4\\_1](https://doi.org/10.1007/978-3-030-29414-4_1)
4. Droste, S., Jansen, T., Wegener, I.: On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 13–22. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0056845>
5. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. *Theor. Comput. Sci.* **276**(1–2), 51–81 (2002)
6. Freedman, D., Drineas, P.: Energy minimization via graph cuts: settling what is possible. In: Computer Society Conference on Computer Vision and Pattern Recognition, pp. 939–946. IEEE Computer Society (2005)
7. Rana, S., Heckendorn, R.B., Whitley, D.: A tractable Walsh analysis of SAT and its implications for genetic algorithms. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI 98), pp. 392–397 (1998)
8. Rudolph, G.: Convergence properties of evolutionary algorithms. Kovac (1997)
9. Stadler, P.F.: Landscapes and their correlation functions. *J. Math. Chem.* **20**(1), 1–45 (1996). <https://doi.org/10.1007/BF01165154>
10. Stadler, P.F., Schnabl, W.: The landscape of the traveling salesman problem. *Phys. Lett. A* **161**(4), 337–344 (1992)
11. Sutton, A.M., Darrell Whitley, L., Howe, A.E.: A polynomial time computation of the exact correlation structure of k-satisfiability landscapes. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2009), pp. 365–372. ACM (2009)



# Benchmarking a $(\mu + \lambda)$ Genetic Algorithm with Configurable Crossover Probability

Furong Ye<sup>1</sup>(✉), Hao Wang<sup>2</sup>, Carola Doerr<sup>2</sup>, and Thomas Bäck<sup>1</sup>

<sup>1</sup> LIACS, Leiden University, Leiden, The Netherlands  
{f.ye,t.h.w.baeck}@liacs.leidenuniv.nl

<sup>2</sup> Sorbonne Université, CNRS, LIP6, Paris, France  
{hao.wang,carola.doerr}@lip6.fr

**Abstract.** We investigate a family of  $(\mu + \lambda)$  Genetic Algorithms (GAs) which creates offspring either from mutation or by recombining two randomly chosen parents. By scaling the crossover probability, we can thus interpolate from a fully mutation-only algorithm towards a fully crossover-based GA. We analyze, by empirical means, how the performance depends on the interplay of population size and the crossover probability.

Our comparison on 25 pseudo-Boolean optimization problems reveals an advantage of crossover-based configurations on several easy optimization tasks, whereas the picture for more complex optimization problems is rather mixed. Moreover, we observe that the “fast” mutation scheme with its power-law distributed mutation strengths outperforms standard bit mutation on complex optimization tasks when it is combined with crossover, but performs worse in the absence of crossover.

We then take a closer look at the surprisingly good performance of the crossover-based  $(\mu + \lambda)$  GAs on the well-known LeadingOnes benchmark problem. We observe that the optimal crossover probability increases with increasing population size  $\mu$ . At the same time, it decreases with increasing problem dimension, indicating that the advantages of the crossover are not visible in the asymptotic view classically applied in runtime analysis. We therefore argue that a mathematical investigation for fixed dimensions might help us observe effects which are not visible when focusing exclusively on asymptotic performance bounds.

**Keywords:** Genetic algorithms · Crossover · Fast mutation

## 1 Introduction

Classic evolutionary computation methods build on two main variation operators: *mutation* and *crossover*. While the former can be mathematically defined as unary operators (i.e., families of probability distributions that depend on a single argument), crossover operators sample from distributions of higher arity, with the goal to “recombine” information from two or more arguments.

There is a long debate in evolutionary computation about the (dis-)advantages of these operators, and about how they interplay with each other [32, 36].

In lack of generally accepted recommendations, the use of these operators still remains a rather subjective decision, which in practice is mostly driven by users' experience. Little guidance is available on which operator(s) to use for which situation, and how to most efficiently interleave them. The question how crossover can be useful can therefore be seen as far from being solved.

Of course, significant research efforts are spent to shed light on this question, which is one of the most fundamental ones that evolutionary computation has to offer. While in the early years of evolutionary computation (see, for example, the classic works [2, 11, 22]) crossover seems to have been widely accepted as an integral part of an evolutionary algorithm, we observe today two diverging trends. Local search algorithms such as GSAT [35] for solving Boolean satisfiability problems, or such as the general-purpose Simulated Annealing [27] heuristic, are clearly very popular optimization methods in practice – both in academic and in industrial applications. These purely mutation-based heuristics are nowadays more commonly studied under the term *stochastic local search*, which forms a very active area of research. Opposed to this is a trend to reduce the use of mutation operators, and to fully base the iterative optimization procedure on recombination operators; see [40] and references therein. However, despite the different recommendations, these opposing positions find their roots in the same problem: we hardly know how to successfully dovetail mutation and crossover.

In addition to large bodies of empirical works aiming to identify useful combinations of crossover and mutation [11, 21, 23, 33],

The question how (or whether) crossover can be beneficial has also always been one of the most prominent problems in *runtime analysis*, the research stream aiming at studying evolutionary algorithms by mathematical means [7, 8, 10, 13–15, 25, 26, 28, 30, 34, 37, 39, 41, 43], most of these results focus on very particular algorithms or problems, and are not (or at least not easily) generalizable to more complex optimization tasks.

*Our Results.* In this work, we study a simple variant of the  $(\mu + \lambda)$  GA which allows us to conveniently scale the relevance of crossover and mutation, respectively, via a single parameter. More precisely, our algorithm is parameterized by a crossover probability  $p_c$ , which is the probability that we generate in the reproduction step an offspring by means of crossover. The offspring is generated by mutation otherwise, so that  $p_c = 0$  corresponds to the mutation-only  $(\mu + \lambda)$  EA, whereas for  $p_c = 1$  the algorithm is entirely based on crossover. Note here that we *either* use crossover *or* mutation, so as to better separate the influence of the two operators.

We first study the performance of different configurations of the  $(\mu + \lambda)$  GA on 25 pseudo-Boolean problems (the 23 functions suggested in [19], a concatenated trap problem, and random NK landscape instances). We observe that the algorithms using crossover perform significantly better on some simple functions as ONEMAX (F1) and LEADINGONES (F2), but also on some problems that are considered hard, e.g., the 1-D Ising model (F19).

We then look more closely into the performance of the algorithm on a benchmark problem intensively studied in runtime analysis: LEADINGONES,

the problem of maximizing the function  $f : \{0, 1\}^n \rightarrow [0..n], x \mapsto \max\{i \in [0..n] \mid \forall j \leq i : x_j = 1\}$ . We observe some very interesting effects, that we believe may motivate the theory community to look at the question of usefulness of crossover from a different angle. More precisely, we find that, against our intuition that uniform crossover cannot be beneficial on LEADINGONES, the performance of the  $(\mu + \lambda)$  GA on LEADINGONES improves when  $p_c$  takes values greater than 0 (and smaller than 1), see Fig. 3. The performances are quite consistent, and we can observe clear patterns, such as a tendency for the optimal value of  $p_c$  (displayed in Table 2) to increase with increasing  $\mu$ , and to decrease with increasing problem dimension. The latter effect may explain why it is so difficult to observe benefits of crossover in theoretical work: they disappear with the asymptotic view that is generally adopted in runtime analysis.

We have also performed similar experiments on ONEMAX (see our project data [44]), but the good performance of the  $(\mu + \lambda)$  GA configurations using crossover is less surprising for this problem, since this benefit has previously been observed for genetic algorithms that are very similar to the  $(\mu + \lambda)$  GA; see [7, 8, 39] for examples and further references. In contrast to a large body of literature on the benefit of crossover for solving ONEMAX, we are not aware of the existence of such results for LEADINGONES, apart from the highly problem-specific algorithms developed and analyzed in [1, 17].

We hope to promote with this work (1) runtime analysis for fixed dimensions, (2) an investigation of the advantages of crossover on LEADINGONES, and (3) the  $(\mu + \lambda)$  GA as a simplified model to study the interplay between problem dimension, population sizes, crossover probability, and mutation rates.

## 2 Algorithms and Benchmarks

We describe in this section our  $(\mu + \lambda)$  GA framework (Sect. 2.1) and the benchmark problems (Sect. 2.2). Since in this paper we can only provide a glimpse on our rich data sets, we also summarize in Sect. 2.3 which data the interested reader can find in our repository [44].

### 2.1 A Family of $(\mu + \lambda)$ Genetic Algorithms

Our main objective is to study the usefulness of crossover for different kinds of problems. To this end, we investigate a meta-model, which allows us to easily transition from a mutation-only to a crossover-only algorithm. Algorithm 1 presents this framework, which, for ease of notation, we refer to as the family of the  $(\mu + \lambda)$  GA in the following.

The  $(\mu + \lambda)$  GA initializes its population uniformly at random (u.a.r., lines 1–2). In each iteration, it creates  $\lambda$  offspring (lines 6–16). For each offspring, we first decide whether to apply crossover (with probability  $p_c$ , lines 8–11) or whether to apply mutation (otherwise, lines 12–15). Offspring that differ from their parents are evaluated, whereas offspring identical to one of their parents inherit this fitness value without function evaluation (see [5] for a discussion).

---

**Algorithm 1: A Family of  $(\mu + \lambda)$  Genetic Algorithms**


---

```

1 Input: Population sizes  $\mu, \lambda$ , crossover probability  $p_c$ , mutation rate  $p$ ;
2 Initialization: for  $i = 1, \dots, \mu$  do sample  $x^{(i)} \in \{0, 1\}^n$  uniformly at random
   (u.a.r.), and evaluate  $f(x^{(i)})$ ;
3 Set  $P = \{x^{(1)}, x^{(2)}, \dots, x^{(\mu)}\}$ ;
4 Optimization: for  $t = 1, 2, 3, \dots$  do
5    $P' \leftarrow \emptyset$ ;
6   for  $i = 1, \dots, \lambda$  do
7     Sample  $r \in [0, 1]$  u.a.r. ;
8     if  $r \leq p_c$  then
9       select two individuals  $x, y$  from  $P$  u.a.r. (w/ replacement);
10       $z^{(i)} \leftarrow \text{Crossover}(x, y)$ ;
11      if  $z^{(i)} \notin \{x, y\}$  then evaluate  $f(z^{(i)})$  else infer  $f(z^{(i)})$  from parent;
12     else
13       select an individual  $x$  from  $P$  u.a.r.;
14        $z^{(i)} \leftarrow \text{Mutation}(x)$ ;
15       if  $z^{(i)} \neq x$  then evaluate  $f(z^{(i)})$  else infer  $f(z^{(i)})$  from parent;
16      $P' \leftarrow P' \cup \{z^{(i)}\}$ ;
17    $P$  is updated by the best  $\mu$  points in  $P \cup P'$  (ties broken u.a.r.);

```

---

The best  $\mu$  of parent and offspring individuals form the new parent population of the next generation (line 17).

Note the unconventional use of *either* crossover *or* mutation. As mentioned, we consider this variant to allow for a better attribution of the effects to each of the operators. Moreover, note that in Alg. 1 we decide for each offspring individually which operator to apply. We call this scheme the  **$(\mu + \lambda)$  GA with offspring-based variator choice**. We also study the performance of the  **$(\mu + \lambda)$  GA with population-based variator choice**, which is the algorithm that we obtain from Alg. 1 by swapping lines 7 and 6.

We study three different crossover operators, *one-point crossover*, *two-point crossover*, and *uniform crossover*, and two different mutation operators, *standard bit mutation* and the *fast mutation* scheme suggested in [16]. These variation operators are briefly described as follows.

- *One-point crossover*: a crossover point is chosen from  $[1..n]$  u.a.r. and an offspring is created by copying the bits from one parent until the crossover point and then copying from the other parent for the remaining positions.
- *Two-point crossover*: similarly, two different crossover points are chosen u.a.r. and the copy process alternates between two parents at each crossover point.
- *Uniform crossover* creates an offspring by copying for each position from the first or from the second parent, chosen independently and u.a.r.
- *Standard bit mutation*: a mutation strength  $\ell$  is sampled from the conditional binomial distribution  $\text{Bin}_{>0}(n, p_m)$ , which assigns to each  $k$  a probability of  $\binom{n}{k} p^k (1-p)^{n-k} / (1 - (1-p)^n)$  [5]. Thereafter,  $\ell$  distinct positions are chosen

- u.a.r. and the offspring is created by first copying the parent and then flipping the bits in these  $\ell$  positions. In this work, we restrict our experiments to the standard mutation rate  $p = 1/n$ . Note, though, that this choice is not necessarily optimal, as in particular the results in [3, 39] and follow-up works demonstrate.
- *Fast mutation* [16]: operates similarly to standard bit mutation except that the mutation strength  $\ell$  is drawn from a power-law distribution:  $\Pr[L = \ell] = (C_{n/2}^\beta)^{-1} \ell^{-\beta}$  with  $\beta = 1.5$  and  $C_{n/2}^\beta = \sum_{i=1}^{n/2} i^{-\beta}$ .

## 2.2 The IOHprofiler Problem Set

To test different configurations of the  $(\mu + \lambda)$  GA, we first perform an extensive benchmarking on the problems suggested in [19], which are available in the IOHprofiler benchmarking environment [18]. This set contains 23 real-valued pseudo-Boolean test problems: **F1 and F4-F10**: ONEMAX (F1) and W-model extensions (F4-10), **F2 and F11-F17**: LEADINGONES (F2) and W-model extensions (F11-17), **F3**: Linear function  $f(\mathbf{x}) = \sum_{i=1}^n ix_i$ , **F18**: Low Autocorrelation Binary Sequences (LABS), **F19-21**: Ising Models, **F22**: Maximum Independent Vertex Set (MIVS), and **F23**: N-Queens (NQP).

We recall that the W-model, originally suggested in [42] and extended in [19], is a collection of perturbations that can be applied to a base problem in order to calibrate its features, such as its neutrality, its epistasis, and its ruggedness. We add to the list of [19] the following two problems:

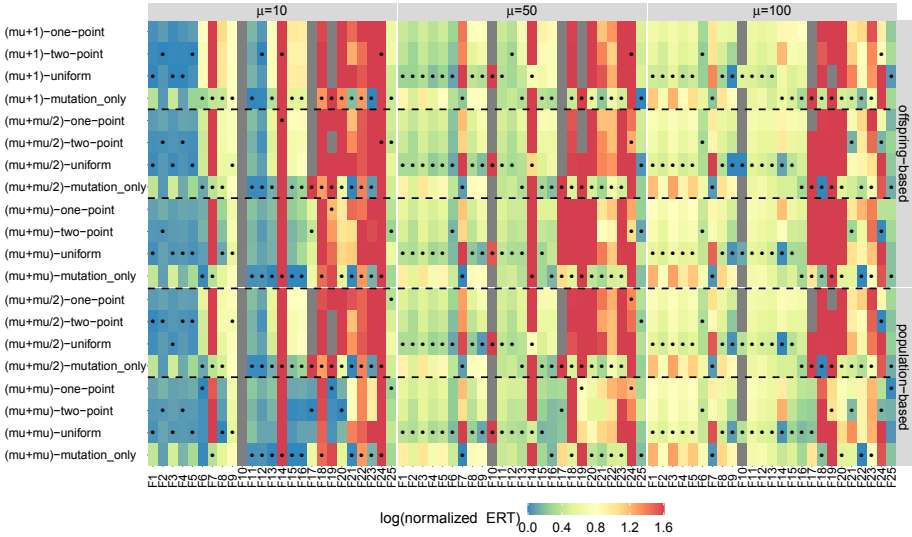
**F24**: Concatenated Trap (CT) is defined by partitioning a bit-string into segments of length  $k$  and concatenating  $m = n/k$  trap functions that takes each segment as input. The trap function is defined as follows:  $f_k^{\text{trap}}(u) = 1$  if the number  $u$  of ones satisfies  $u = k$  and  $f_k^{\text{trap}}(u) = \frac{k-1-u}{k}$  otherwise. We use  $k = 5$  in our experiments.

**F25**: Random NK landscapes (NKL). The function values are defined as the average of  $n$  sub-functions  $F_i: [0..2^{k+1} - 1] \rightarrow \mathbb{R}$ ,  $i \in [1..n]$ , where each component  $F_i$  only takes as input a set of  $k \in [0..n - 1]$  bits that are specified by a neighborhood matrix. In this paper,  $k$  is set to 1 and entries of the neighbourhood matrix are drawn u.a.r. in  $[1..n]$ . The function values of  $F_i$ 's are sampled independently from a uniform distribution on  $(0, 1)$ .

Note that the IOHprofiler problem set provides for each problem several problem instances, which all have isomorphic fitness landscapes, but different problem representations. In our experiments we only use the first instance of each problem (seed 1). For the mutation-based algorithms and the ones using uniform crossover, the obtained results generalize to all other problem instances. For algorithms involving one- or two-point crossover, however, this is not the case, as these algorithms are not unbiased (in the sense of Lehre and Witt [29]).

## 2.3 Data Availability

Detailed results for the different configurations of the  $(\mu + \lambda)$  GA are available in our data repository at [44]. In particular, we host there data for the IOHprofiler



**Fig. 1.** Heat map of normalized ERT values of the  $(\mu + \lambda)$  GA with offspring-based (top part) and population-based (bottom part) variator choice for the 100-dimensional benchmark problems, computed based on the target values specified in Table 1. The crossover probability  $p_c$  is set to 0.5 for all algorithms except the mutation-only ones (which use  $p_c = 0$ ). The displayed values are the the quotient of the ERT and  $ERT_{best}$ , the ERT achieved by the best of all displayed algorithms. These quotients are capped at 40 to increase interpretability of the color gradient in the most interesting region. The three algorithm groups – the  $(\mu + 1)$ , the  $(\mu + \lceil \mu/2 \rceil)$ , and the  $(\mu + \mu)$  GAs – are separated by dashed lines. A dot indicates the best algorithm of each group of four. A grey tile indicates that the  $(\mu + \lambda)$  GA configuration failed, in all runs, to find the target value within the given budget. (Color figure online)

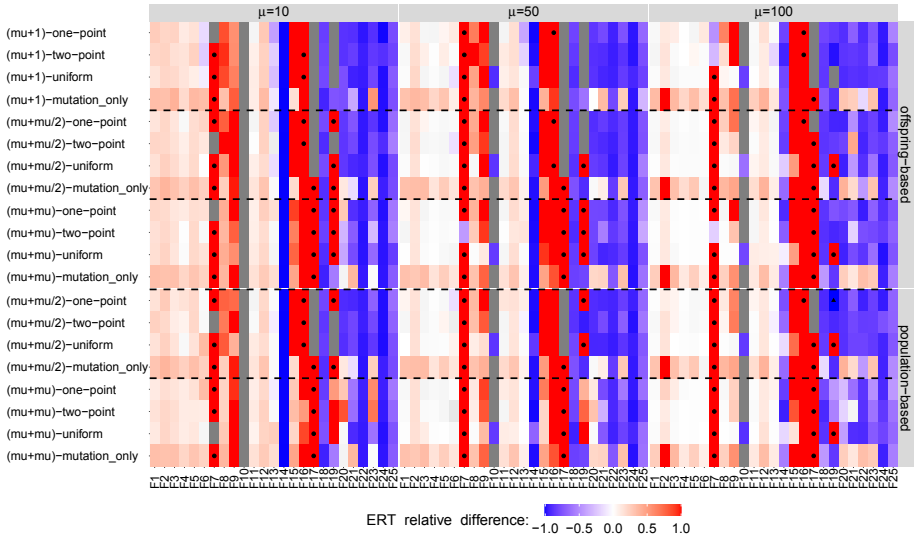
experiments (36 algorithms, 25 functions, 5 dimensions  $\leq 250$ , 100 independent runs) and for the  $(\mu + \lambda)$  GA on ONEMAX and on LEADINGONES for all of the following 5544 parameter combinations:  $n \in \{64, 100, 150, 200, 250, 500\}$  (6 values),  $\mu \in \{2, 3, 5, 8, 10, 20, 30, \dots, 100\}$  (14 values),  $\lambda \in \{1, \lceil \mu/2 \rceil, \mu\}$  (3 values),  $p_c \in \{0.1k \mid k \in [0..9]\} \cup \{0.95\}$  (11 values), two mutation operators (standard bit mutation and fast mutation). In these experiments on ONEMAX and LEADINGONES, the crossover operator is fixed to uniform crossover.

A detailed analysis of these results, for example using IOHprofiler or using HiPlot [9] may give additional insights into the dependence of the overall performance on the parameter setting.

### 3 Results for the IOHprofiler Problems

In order to probe into the empirical performance of the  $(\mu + \lambda)$  GA, we test it on the 25 problems mentioned in Sect. 2.2, with a total budget of  $100n^2$  function evaluations. We perform 100 independent runs of each algorithm on each





**Fig. 2.** Heat map comparing standard bit mutation (sbm) with fast mutation on the 25 problems from Sect. 2.2 in dimensions  $n = 100$ . Plotted values are  $(ERT_{fast} - ERT_{sbm})/ERT_{sbm}$ , for ERTs computed wrt the target values specified in Table 1.  $p_c$  is set to 0.5 for all crossover-based algorithms. Values are bounded in  $[-1, 1]$  to increase visibility of the color gradient in the most interesting region. A black dot indicates that the  $(\mu + \lambda)$  GA with fast mutation failed in all runs to find the target with the given budget; the black triangle signals failure of standard bit mutation, and a gray tile is chosen for settings in which the  $(\mu + \lambda)$  GA failed for both mutation operators. (Color figure online)

**Table 1.** Target values used for computing the ERT value in Fig. 1.

$F_1$	$F_2$	$F_3$	$F_4$	$F_5$	$F_6$	$F_7$	$F_8$	$F_9$	$F_{10}$	$F_{11}$	$F_{12}$	$F_{13}$	$F_{14}$
100	100	5050	50	90	33	100	51	100	100	50	90	33	7
$F_{15}$	$F_{16}$	$F_{17}$	$F_{18}$		$F_{19}$	$F_{20}$	$F_{21}$	$F_{22}$	$F_{23}$	$F_{24}$	$F_{25}$		
51	100	100	4.215852		98	180	260	42	9	17.196	-0.2965711		

problem. A variety of parameter settings are investigated: (1) all three crossover operators described in Sect. 2 (we use  $p_c = 0.5$  for all crossover-based configurations), (2) both mutation variator choices, (3)  $\mu \in \{10, 50, 100\}$ , and (4)  $\lambda \in \{1, \lceil \mu/2 \rceil, \mu\}$ .

In Fig. 1, we highlight a few basic results of this experimentation for  $n = 100$ , where the mutation operator is fixed to the standard bit mutation. More precisely, we plot in this figure the normalized expected running time (ERT), where the normalization is with respect to the best ERT achieved by any of the algorithms for the same problem. Table 1 provides the target values for which we computed the ERT values. For each problem and each algorithm, we first

calculated the 2% percentile of the best function values. We then selected the largest of these percentiles (over all algorithms) as target value.

On the ONEMAX-based problems F1, F4, and F5, the  $(\mu + \lambda)$  GA outperforms the mutation-only GA, regardless of the variator choice scheme, the crossover operator, and the setting of  $\lambda$ . When looking at problem F6, we find out that when  $\mu = 10$  the mutation-only GA surpasses most of  $(\mu + \lambda)$  GA variants except the population-based  $(\mu + \mu)$  GA with one-point crossover. On F8–10, the  $(\mu + \lambda)$  GA takes the lead in general, whereas it cannot rival the mutation-only GA on F7. Also, only the configuration with uniform crossover can hit the optimum of F10 within the given budget.

On the linear function F3 we observe a similar behavior as on ONEMAX. On LEADINGONES (F2), the  $(\mu + \lambda)$  GA outperforms the mutation-only GA again for  $\mu \in \{50, 100\}$  while for  $\mu = 10$  the mutation-only GA becomes superior with one-point and uniform crossovers. On F11–13 and F15–16 (the W-model extensions of LEADINGONES), the mutation-only GA shows a better performance than the  $(\mu + \lambda)$  GA with one-point and uniform crossovers and this advantage becomes more significant when  $\mu = 10$ . On problem F14, that is created from LEADINGONES using the same transformation as in F7, the mutation-only GA is inferior to the  $(\mu + \lambda)$  GA with uniform crossover.

On problems F18 and F23, the mutation-only GA outperforms the  $(\mu + \lambda)$  GA for most parameter settings. On F21, the  $(\mu + \lambda)$  GA with two-point crossover yields a better result when the population size is larger (i.e.,  $\mu = 100$ ) while the mutation-only GA takes the lead for  $\mu = 10$ . On problems F19 and F20, the  $(\mu + \mu)$  GA with the population-based variator choice significantly outperforms all other algorithms, whereas it is substantially worse for the other parameter settings. On problem F24, the  $(\mu + \mu/2)$  GA with two-point crossover achieves the best ERT value when  $\mu = 100$ . None of the tested algorithms manages to solve F24 with the given budget. The target value used in Fig. 1 is 17.196, which is below the optimum 20. On problem F25, the mutation-only GA and the  $(\mu + \lambda)$  GA are fairly comparable when  $\mu \in \{10, 50\}$ . Also, we observe that the population-based  $(\mu + \mu)$  GA outperforms the mutation-only GA when  $\mu = 100$ .

In general, we have made the following observations: (1) on problems F1–6, F8–9, and F11–13, all algorithms obtain better ERT values with  $\mu = 10$ . On problems F7, F14, and F21–25, the  $(\mu + \lambda)$  GA benefits from larger population sizes, i.e.,  $\mu = 100$ ; (2) The  $(\mu + \mu)$  GA with uniform crossover and the mutation-only GA outperform the  $(\mu + \lceil \mu/2 \rceil)$  GA across all three settings of  $\mu$  on most of the problems, except F10, F14, F18, and F22. For the population-based variator choice scheme, increasing  $\lambda$  from one to  $\mu$  improves the performance remarkably on problems F17–24. Such an improvement becomes negligible for the offspring-based scheme; (3) Among all three crossover operators, the uniform crossover often surpasses the other two on ONEMAX, LEADINGONES, and the W-model extensions thereof.

To investigate the impact of mutation operators on GA, we plot in Fig. 2 the relative ERT difference between the  $(\mu + \lambda)$  GA configurations using fast and standard bit mutation, respectively. As expected, fast mutation performs

slightly worse on F1–6, F8, and F11–13. On problems F7, F9, and F15–17, however, fast mutation becomes detrimental to the ERT value for most parameter settings. On problems F10, F14, F18, and F21–25, fast mutation outperforms standard bit mutation, suggesting a potential benefit of pairing the fast mutation with crossover operators to solve more difficult problems. Interestingly, with an increasing  $\mu$ , the relative ERT of the  $(\mu + \lambda)$  GA quickly shrinks to zero, most notably on F1–6, F8, F9, F11–13.

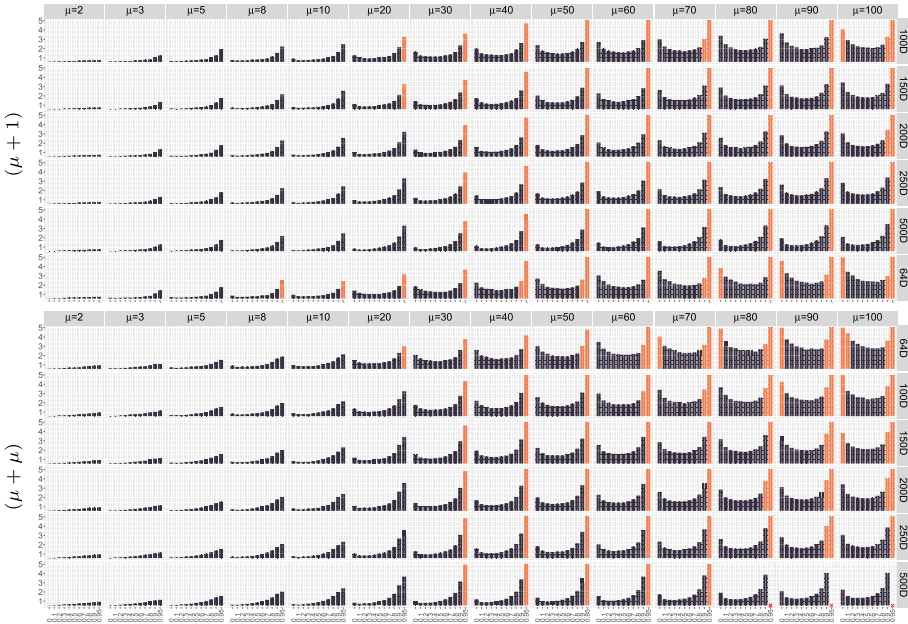
Interestingly, in [31], an empirical study has shown that on a randomly generated maximum flow test generation problem, fast mutation is significantly outperformed by standard bit mutation when combined with uniform crossover. Such an observation seems contrary to our findings on F10, F14, F18, and F21–25. However, it is made on a standard  $(100 + 70)$  GA in which both crossover and mutation are applied to the parent in order to generate offspring. We are planning to investigate the effects of this inter-chaining in future work, but this topic is beyond the focus of this study.

## 4 Case-Study: LeadingOnes

The surprisingly good performance of the  $(\mu + \lambda)$  GA with  $p_c = 0.5$  on LEADINGONES motivates us to investigate this setting in more detail. Before we go into the details of the experimental setup and our results, we recall that for the optimization of LEADINGONES, the fitness values only depend on the first bits, whereas the tail is randomly distributed and has no influence on the selection. More precisely, a search point  $x$  with LEADINGONES-value  $f(x)$  has the following structure: the first  $f(x)$  bits are all 1, the  $f(x) + 1$ st bit equals 0, and the entries in the tail (i.e., in positions  $[f(x) + 2..n]$ ) did not have any influence on the optimization process so far. For many algorithms, it can be shown that these tail bits are uniformly distributed, see [12] for an extended discussion.

**Experimental Setup.** We fix in this section the variator choice to the *offspring-based setting*. We do so because its performance was seen to be slightly better on LEADINGONES than the population-based choice. We experiment with the parameter settings specified in Sect. 2.3. For each of the settings listed there, we perform 100 independent runs, with a maximal budget of  $5n^2$  each.

**Overall Running Time.** We first investigate the impact of the crossover probability on the average running time, i.e., on the average number of function evaluations that the algorithm performs until it evaluates the optimal solution for the first time. The results for the  $(\mu + 1)$  and the  $(\mu + \mu)$  GA using uniform crossover and standard bit mutation are summarized in Fig. 3. Since not all algorithms managed to find the optimum within the given time budget, we plot as red bars the ERT values for such algorithms with success ratio strictly smaller than 1, whereas the black bars are reserved for algorithms with 100 successful runs. All values are normalized by  $n^2$ , to allow for a better comparison. All patterns described below also apply to the  $(\mu + \lceil \mu/2 \rceil)$  GA, whose results we do not display for reasons of space. They are also very similar when we replace the mutation operator by the fast mutation scheme suggested in [16].



**Fig. 3.** By  $n^2$  normalized ERT values for the  $(\mu + \lambda)$  GA using standard bit mutation and uniform crossover on LEADINGONES, for different values of  $\mu$  and for  $\lambda = 1$  (top) and for  $\lambda = \mu$  (bottom). Results are grouped by the value of  $\mu$  (main columns), by the crossover probability  $p_c$  (minor columns), and by the dimension (rows). The ERTs are computed from 100 independent runs for each setting, with a maximal budget of  $5n^2$  fitness evaluations. ERTs for algorithms which successfully find the optimum in all 100 runs are depicted as black bars, whereas ERTs for algorithms with success rates in  $(0, 1)$  are depicted as red bars. All bars are capped at 5. (Color figure online)

As a first observation, we note that the pattern of the results are quite regular. As can be expected, the dispersion of the running times is rather small. For reasons of space, we do not describe this dispersion in detail, but to give an impression for the concentration of the running times, we report that the standard deviation of the  $(50 + 1)$  GA on the 100-dimensional LEADINGONES function is approximately 14% of the average running time across all values of  $p_c$ . As can be expected for a genetic algorithm on LEADINGONES, the average running increases with increasing population size  $\mu$ , see [38] for a proof of this statement when  $p_c = 0$ .

Next, we compare the sub-plots in each row, i.e., fixing the dimension. We see that the  $(\mu + \lambda)$  GA suffers drastically from large  $p_c$  values when  $\mu$  is smaller, suggesting that the crossover operator hinders performance. But as  $\mu$  gets larger, the average running time at moderate crossover probabilities ( $p_c$  around 0.5) is significantly smaller than that in two extreme cases,  $p_c = 0$  (mutation-only GAs), and  $p_c = 0.95$ . This observation holds for all dimensions and for both algorithm families, the  $(\mu + 1)$  and the  $(\mu + \mu)$  GA.

Looking at the sub-plots in each column (i.e., fixing the population size), we identify another trend: for those values of  $\mu$  for which an advantage of  $p_c > 0$  is visible for the smallest tested dimension,  $n = 64$ , the relative advantage of this rate decreases and eventually disappears as the dimension increases.

Finally, we compare the results of the  $(\mu + 1)$  GA with those of the  $(\mu + \mu)$  GA. Following [24], it is not surprising that for  $p_c = 0$ , the results of the  $(\mu + 1)$  GA are better than those of the  $(\mu + \mu)$  GA (very few exceptions to this rule exist in our data, but in all these cases the differences in average runtime are negligibly small), and following our own theoretical analysis [20, Theorem 1], it is not surprising that the differences between these two algorithmic families are rather small: the typical disadvantage of the  $(\mu + \lceil \mu/2 \rceil)$  GA over the  $(\mu + 1)$  GA is around 5% and it is around 10% for the  $(\mu + \mu)$  GA, but these relative values differ between the different configurations and dimensions.

**Optimal Crossover Probabilities.** To make our observations on the crossover probability clearer, we present in Table 2 a heatmap of the values  $p_c^*$  for which we observed the best average running time (with respect to all tested  $p_c$  values). We see the same trends here as mentioned above: as  $\mu$  increases, the value of  $p_c^*$  increases, while, for fixed  $\mu$  its value decreases with increasing problem dimension  $n$ . Here again we omit details for the  $(\mu + \lceil \mu/2 \rceil)$  GA and for the fast mutation scheme, but the patterns are identical, with very similar absolute values.

**Table 2.** On LEADINGONES, the optimal value of  $p_c$  for the  $(\mu + 1)$  and the  $(\mu + \mu)$  GA with uniform crossover and standard bit mutation, for various combinations of dimension  $n$  (rows) and  $\mu$  (columns). Values are approximated from 100 independent runs each, probing  $p_c \in \{0.1k \mid k \in [0..9]\} \cup \{0.95\}$ .

		$n \setminus \mu$	2	3	5	8	10	20	30	40	50	60	70	80	90	100
$(\mu + 1)$	64	0.0	0.1	0.1	0.1	0.2	0.3	0.5	0.4	0.5	0.5	0.6	0.7	0.6	0.7	
	100	0.0	0.1	0.1	0.1	0.1	0.3	0.4	0.4	0.4	0.5	0.5	0.5	0.4	0.6	
	150	0.0	0.1	0.1	0.1	0.1	0.2	0.3	0.3	0.4	0.4	0.5	0.4	0.4	0.5	
	200	0.0	0.0	0.1	0.1	0.1	0.2	0.2	0.3	0.3	0.3	0.4	0.4	0.4	0.4	
	250	0.0	0.0	0.1	0.1	0.1	0.2	0.2	0.3	0.3	0.3	0.4	0.4	0.3	0.4	
	500	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3
$(\mu + \mu)$	64	0.0	0.2	0.1	0.1	0.2	0.2	0.4	0.4	0.6	0.5	0.5	0.7	0.5	0.7	
	100	0.0	0.0	0.1	0.1	0.2	0.3	0.3	0.3	0.5	0.4	0.5	0.5	0.6	0.5	
	150	0.0	0.0	0.1	0.1	0.2	0.2	0.3	0.3	0.5	0.4	0.5	0.5	0.5	0.5	
	200	0.0	0.0	0.1	0.1	0.1	0.1	0.3	0.3	0.3	0.3	0.4	0.5	0.4	0.5	
	250	0.0	0.0	0.1	0.1	0.1	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.4	
	500	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3

**Fixed-Target Running Times.** We now study where the advantage of the crossover-based algorithms stems from. We demonstrate this using the example of the  $(50 + 50)$  GA in 200 dimensions. We recall from Table 2 that the optimal crossover probability for this setting is  $p_c^* = 0.3$ . The left plot in Fig. 4 is a fixed-target plot, in which we display for each tested crossover probability  $p_c$  (different lines) and each fitness value  $i \in [0..200]$  ( $x$ -axis) the average time needed until the respective algorithm evaluates for the first time a search point of fitness at



**Fig. 4. Left:** Average fixed-target running times of the  $(50 + 50)$  GA with uniform crossover and standard bit mutation on LEADINGONES in 200 dimensions, for different crossover probabilities  $p_c$ . Results are averages of 100 independent runs. **Right:** Gradient of selected fixed-target curves.

least  $i$ . The mutation-only configuration ( $p_c = 0$ ) performs on par with the best configurations for the first part of the optimization process, but then loses in performance as the optimization progresses. The plot on the right shows the gradients of the fixed-target curves. The gradient can be used to analyze which configuration performs best at a given target value. We observe an interesting behavior here, namely that the gradient of the configuration  $p_c = 0.8$ , which has a very bad fixed-target performance on all targets (left plot), is among the best in the final parts of the optimization. The plot on the right therefore suggests that an adaptive choice of  $p_c$  should be investigated further.

## 5 Conclusions

In this paper, we have analyzed the performance of a family of  $(\mu + \lambda)$  GAs, in which offspring are either generated by crossover (with probability  $p_c$ ) or by mutation (probability  $1 - p_c$ ). On the IOHprofiler problem set, it has been shown that this random choice mechanism reduces the expecting running time on ONEMAX, LEADINGONES, and many W-model extensions of those two problems. By varying the value of the crossover probability  $p_c$ , we discovered on LEADINGONES that its optimal value  $p_c^*$  (with respect to the average running time) increases with the population size  $\mu$ , whereas for fixed  $\mu$  it decreases with increasing dimension  $n$ .

Our results raise the interesting question of whether a non-asymptotic runtime analysis (i.e., bounds that hold for a fixed dimension rather than in big-Oh notation) could shed new light on our understanding of evolutionary algorithms. We note that a few examples of such analyses can already be found in the literature, e.g., in [4, 6]. The regular patterns observed in Fig. 3 suggest the presence of trends that could be turned into formal knowledge.

It would certainly also be interesting to extend our study to a  $(\mu + \lambda)$  GA variant using dynamic values for the relevant parameters  $\mu$ ,  $\lambda$ , crossover probability  $p_c$ , and mutation rate  $p$ . We are also planning to extend the study to more conventional  $(\mu + \lambda)$  GA, which apply mutation right after crossover.

**Acknowledgments.** Our work was supported by the Chinese scholarship council (CSC No. 201706310143), by the Paris Ile-de-France Region, by ANR-11-LABX-0056-LMH (LabEx LMH), and by COST Action CA15140.

## References

1. Afshani, P., Agrawal, M., Doerr, B., Doerr, C., Larsen, K.G., Mehlhorn, K.: The query complexity of finding a hidden permutation. In: Brodnik, A., López-Ortiz, A., Raman, V., Viola, A. (eds.) *Space-Efficient Data Structures, Streams, and Algorithms*. LNCS, vol. 8066, pp. 1–11. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40273-9\\_1](https://doi.org/10.1007/978-3-642-40273-9_1)
2. Bäck, T.: *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press Inc, Oxford (1996)
3. Böttcher, S., Doerr, B., Neumann, F.: Optimal fixed and adaptive mutation rates for the leadingones problem. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) *PPSN 2010*. LNCS, vol. 6238, pp. 1–10. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15844-5\\_1](https://doi.org/10.1007/978-3-642-15844-5_1)
4. Buskalic, N., Doerr, C.: Maximizing drift is not optimal for solving OneMax. In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2019)*, pp. 425–426. ACM (2019). <http://arxiv.org/abs/1904.07818>
5. Pinto, E.C., Doerr, C.: A simple proof for the usefulness of crossover in black-box optimization. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) *PPSN 2018*. LNCS, vol. 11102, pp. 29–41. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_3](https://doi.org/10.1007/978-3-319-99259-4_3)
6. Chicano, F., Sutton, A.M., Whitley, L.D., Alba, E.: Fitness probability distribution of bit-flip mutation. *Evol. Comput.* **23**(2), 217–248 (2015)
7. Corus, D., Oliveto, P.S.: Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. *IEEE Trans. Evol. Comput.* **22**(5), 720–732 (2018)
8. Corus, D., Oliveto, P.S.: On the benefits of populations for the exploitation speed of standard steady-state genetic algorithms. *Algorithmica* 1–31 (2020). <https://doi.org/10.1007/s00453-020-00743-1>
9. Haziza, D., Rapin, J.: HiPlot - high dimensional interactive plotting (2020). <https://github.com/facebookresearch/hiplot>
10. Dang, D.C., et al.: Escaping local optima using crossover with emergent diversity. *IEEE Trans. Evol. Comput.* **22**(3), 484–497 (2017)
11. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, University of Michigan, Ann Arbor, MI, USA (1975)
12. Doerr, B.: Analyzing randomized search heuristics via stochastic domination. *Theoret. Comput. Sci.* **773**, 115–137 (2019)
13. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. *Theoret. Comput. Sci.* **567**, 87–104 (2015)

14. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. *Theoret. Comput. Sci.* **425**, 17–33 (2012)
15. Doerr, B., Johannsen, D., Kötzing, T., Neumann, F., Theile, M.: More effective crossover operators for the all-pairs shortest path problem. *Theoret. Comput. Sci.* **471**, 12–26 (2013)
16. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2017)*, pp. 777–784. ACM (2017)
17. Doerr, B., Winzen, C.: Black-box complexity: breaking the  $O(n \log n)$  barrier of leadingOnes. In: Hao, J.-K., Legrand, P., Collet, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) *EA 2011. LNCS*, vol. 7401, pp. 205–216. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-35533-2\\_18](https://doi.org/10.1007/978-3-642-35533-2_18)
18. Doerr, C., Wang, H., Ye, F., van Rijn, S., Bäck, T.: IOHprofiler: a benchmarking and profiling tool for iterative optimization heuristics. *arXiv e-prints:1810.05281*, October 2018. <https://arxiv.org/abs/1810.05281>
19. Doerr, C., Ye, F., Horesh, N., Wang, H., Shir, O.M., Bäck, T.: Benchmarking discrete optimization heuristics with IOHprofiler. *Appl. Soft Comput.* **88**, 106027 (2019)
20. Doerr, C., Ye, F., van Rijn, S., Wang, H., Bäck, T.: Towards a theory-guided benchmarking suite for discrete black-box optimization heuristics: profiling  $(1 + \lambda)$  EA variants on onemax and leadingones. In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2018)*, pp. 951–958. ACM (2018)
21. Elsayed, S.M., Sarker, R.A., Essam, D.L.: Multi-operator based evolutionary algorithms for solving constrained optimization problems. *Comput. Oper. Res.* **38**(12), 1877–1896 (2011)
22. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st edn. Addison-Wesley Longman Publishing Co. Inc, Boston (1989)
23. Yoon, H.S., Moon, B.R.: An empirical study on the synergy of multiple crossover operators. *IEEE Trans. Evol. Comput.* **6**(2), 212–223 (2002)
24. Jansen, T., De Jong, K.A., Wegener, I.: On the choice of the offspring population size in evolutionary algorithms. *Evol. Comput.* **13**, 413–440 (2005)
25. Jansen, T., Wegener, I.: The analysis of evolutionary algorithms—a proof that crossover really can help. *Algorithmica* **34**, 47–66 (2002). <https://doi.org/10.1007/s00453-002-0940-2>
26. Jansen, T., Wegener, I.: Real royal road functions—where crossover provably is essential. *Discrete Appl. Math.* **149**(1–3), 111–125 (2005)
27. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* **220**, 671–680 (1983)
28. Kötzing, T., Sudholt, D., Theile, M.: How crossover helps in pseudo-Boolean optimization. In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2011)*, pp. 989–996. ACM (2011)
29. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. *Algorithmica* **64**, 623–642 (2012). <https://doi.org/10.1007/s00453-012-9616-8>
30. Lehre, P.K., Yao, X.: Crossover can be constructive when computing unique input-output sequences. *Soft. Comput.* **15**(9), 1675–1687 (2011). <https://doi.org/10.1007/s00500-010-0610-2>
31. Mironovich, V., Buzdalov, M.: Evaluation of heavy-tailed mutation operator on maximum flow test generation problem. In: *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2017)*, Companion Material, pp. 1423–1426. ACM (2017)



32. Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform hill climbing? In: Proceedings of Neural Information Processing Systems Conference (NIPS 1993). Advances in Neural Information Processing Systems, vol. 6, pp. 51–58. Morgan Kaufmann (1993)
33. Murata, T., Ishibuchi, H.: Positive and negative combination effects of crossover and mutation operators in sequencing problems. In: Proceedings of Conference on Evolutionary Computation, pp. 170–175, May 1996
34. Neumann, F., Oliveto, P.S., Rudolph, G., Sudholt, D.: On the effectiveness of crossover for migration in parallel evolutionary algorithms. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2011), pp. 1587–1594. ACM (2011)
35. Selman, B., Levesque, H.J., Mitchell, D.G.: A new method for solving hard satisfiability problems. In: Proceedings of National Conference on Artificial Intelligence, pp. 440–446. AAAI (1992)
36. Spears, W.M.: Crossover or mutation? In: Banzhaf, W. (ed.) Foundations of Genetic Algorithms, vol. 2, pp. 221–237. Elsevier, Amsterdam (1993)
37. Sudholt, D.: Crossover is provably essential for the Ising model on trees. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2005), pp. 1161–1167. ACM Press (2005)
38. Sudholt, D.: A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Trans. Evol. Comput.* **17**, 418–435 (2013)
39. Sudholt, D.: How crossover speeds up building block assembly in genetic algorithms. *Evol. Comput.* **25**(2), 237–274 (2017)
40. Varadarajan, S., Whitley, D.: The massively parallel mixing genetic algorithm for the traveling salesman problem. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2019), pp. 872–879. ACM (2019)
41. Watson, R.A., Jansen, T.: A building-block royal road where crossover is provably essential. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 1452–1459. ACM (2007)
42. Weise, T., Wu, Z.: Difficult features of combinatorial optimization problems and the tunable w-model benchmark problem for simulating them. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2018, Companion Material), pp. 1769–1776. ACM (2018)
43. Whitley, D., Varadarajan, S., Hirsch, R., Mukhopadhyay, A.: Exploration and exploitation without mutation: solving the *Jump* function in  $\theta(n)$  time. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 55–66. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99259-4\\_5](https://doi.org/10.1007/978-3-319-99259-4_5)
44. Ye, F., Wang, H., Doerr, C., Bäck, T.: Experimental data sets for the study benchmarking a  $(\mu + \lambda)$  genetic algorithm with configurable crossover probability, April 2020. <https://doi.org/10.5281/zenodo.3753086>

## Author Index

- Abdelkafi, Omar I-303  
Aboutaib, Brahim II-97  
Adam, Lukáš II-257  
Adriaensen, Steven I-691  
Aguirre, Hernan I-33  
Ahiod, Belaïd II-97  
Akimoto, Youhei I-81  
Alderliesten, Tanja II-186, II-215, II-441  
Alghamdi, Mahfouth II-329  
Anastacio, Marie I-95  
Antipov, Denis II-545, II-560  
Arnold, Dirk V. I-184  
Artigues, Christian I-332  
Ashtari, Parastoo I-317  
Asteroth, Alexander I-140  
Auger, Anne I-707  
Awad, Noor I-691
- Bäck, Thomas I-140, I-229, I-512, II-313,  
II-699  
Bagherbeik, Mohammad I-317  
Bartz-Beielstein, Thomas I-243  
Bel, Arjan II-441  
Benoist, Thierry I-332  
Bernabé Rodríguez, Amín V. II-3  
Bi, Ying I-3  
Białas, Marcin I-433  
Biedenkapp, André I-691  
Binder, Martin I-448  
Bischl, Bernd I-448  
Bishop, Jordan T. II-471  
Blaise, Léa I-332  
Boria, Simonetta I-169  
Bosman, Peter A. N. II-186, II-215, II-441  
Bossek, Jakob I-48, I-111, I-346  
Bujny, Mariusz I-169  
Buzdalov, Maxim II-560, II-574, II-675  
Buzdalova, Arina II-485
- Cai, Shaowei I-373  
Caraffini, Fabio I-229  
Carlet, Claude II-343  
Chang, Furong I-484  
Chen, Weiyu I-201, II-257
- Chen, Zefeng II-634  
Chevaleyre, Yann II-661  
Chi, Zongzheng II-229  
Chicano, Francisco II-125  
Christie, Lee A. I-360  
Clare, Amanda I-390  
Coello Coello, Carlos A. I-650, II-3, II-48,  
II-201, II-285  
Cofala, Tim II-357  
Collet, Pierre I-524  
Craven, Matthew J. II-299  
Czajkowski, Marcin II-372
- da Silva Soares, Anderson II-171  
Dandl, Susanne I-448  
Daykin, Jacqueline W. I-390  
de Almeida Ribeiro, Lucas II-171  
de Lima, Telma Woerle II-171  
De Lorenzo, Andrea II-79  
Deist, Timo M. II-186  
Derbel, Bilel I-33, I-303, II-97  
Deutz, André II-313  
Djurasevic, Marko II-343  
Do, Anh Viet II-588  
Doerr, Benjamin II-139, II-545, II-560,  
II-604, II-619  
Doerr, Carola I-111, I-154, I-169, II-139,  
II-485, II-574, II-699  
Dong, Shaozheng II-229  
Dreo, Johann II-139  
du Preez-Wilkinson, Nathaniel II-500  
Duggan, Jim I-637  
Durasevic, Marko II-111
- Eiben, A. E. I-243  
Elend, Lars II-357  
Emmerich, Michael II-171, II-313  
Escalante, Hugo Jair II-48
- Falcón-Cardona, Jesús Guillermo II-201  
Fonlupt, Cyril II-97  
Fonseca, Alcides II-18  
Friess, Stephen I-583  
Fu, Guoxia I-125

- Gallagher, Marcus II-471, II-500  
 Gąsior, Jakub I-678  
 Glasmachers, Tobias I-597  
 Grimme, Christian II-154
- Hagg, Alexander I-140  
 Hakanen, Jussi II-243  
 Hall, George T. I-19  
 Hansen, Nikolaus I-707  
 Hemberg, Erik I-552  
 Hong, Wenjing I-470  
 Hoos, Holger I-65, I-95, I-373  
 Horn, Matthias II-385  
 Howley, Enda I-637  
 Hu, Shengxiang I-484  
 Huang, Zhengxin II-634  
 Hutter, Frank I-691
- Ishibuchi, Hisao I-201, II-257
- Jacques, Julie I-65  
 Jakobovic, Domagoj II-111, II-343  
 Jaskiewicz, A. I-215  
 Jeannin-Girardon, Anne I-524  
 Jiang, He II-229  
 Jin, Yaochu I-125  
 Jourdan, Laetitia I-65  
 Jurczuk, Krzysztof II-372
- Kanda, Kouichi I-317  
 Kaufmann, Thomas II-385  
 Kaźmierczak, Stanisław I-498  
 Keedwell, Ed II-413  
 Kerschke, Pascal I-48, I-111, II-154  
 Kessaci, Marie-Éléonore I-65  
 Komarnicki, Marcin M. I-418  
 Kong, Jiawen I-512  
 Kononova, Anna V. I-229  
 Kötzing, Timo II-648  
 Kowalczyk, Wojtek I-512  
 Kramer, Oliver II-357  
 Krause, Oswin I-597  
 Krawiec, Krzysztof I-623  
 Kretowski, Marek II-372
- Lacroix, Benjamin I-287  
 Lai, Zhihui I-567  
 Lengler, Johannes I-610
- Li, Ke II-271  
 Li, Xiaochen II-229  
 Liefoghe, Arnaud I-33, I-303, II-97, II-201  
 Lindauer, Marius I-691  
 Liskowski, Paweł I-623, II-515  
 Liu, Jialin II-454  
 Lu, Yuwu I-567  
 Lung, Rodica Ioana I-539  
 Luo, Chuan I-373  
 Lv, Ying I-484  
 Lynch, David II-33
- Major, Lily I-390  
 Mańdziuk, Jacek I-433, I-498  
 Maree, Stefanus C. II-186, II-215  
 Mc Donnell, Nicola I-637  
 McCall, John I-287  
 McDermott, James II-33  
 Medvet, Eric II-79  
 Meier, Jonas I-610  
 Menzel, Stefan I-512, I-583, II-428  
 Meunier, Laurent I-154, II-661  
 Michalak, Krzysztof II-399  
 Miettinen, Kaisa II-243  
 Minku, Leandro L. II-428  
 Mirbach, Philip II-357  
 Mirończuk, Marcin Michał I-433  
 Mishra, Sumit II-675  
 Molnar, Christoph I-448  
 Moore, Jason H. II-63  
 Mora, Benjamin I-390  
 Morales-Reyes, Alicia II-48  
 Mousavi, Seyed Farzad I-317
- Neumann, Aneta I-111, I-346, I-404  
 Neumann, Frank I-111, I-346, I-404, II-588  
 Niatsetski, Yury II-441
- O'Neill, Michael II-33  
 O'Reilly, Una-May I-552  
 Ochoa, Gabriela II-125  
 Ohtani, Makoto I-81  
 Oliveto, Pietro S. I-19  
 Orhand, Romain I-524
- Pacheco-Del-Moral, Oscar I-650  
 Parrend, Pierre I-524  
 Peña Gamboa, Leonel Jose I-390  
 Picek, Stjepan II-111, II-343

- Pieters, Bradley R. II-441  
 Plaat, Aske II-528  
 Pohl, Janina I-48  
 Prellberg, Jonas II-357  
 Preuss, Mike II-528  
 Przewozniczek, Michal W. I-418  
  
 Raidl, Günther R. II-385  
 Rajabi, Amirhossein I-664  
 Randone, Francesca II-79  
 Rapin, Jeremy I-154, II-661  
 Raponi, Elena I-169  
 Rebolledo, Margarita I-243  
 Rehbach, Frederik I-243, I-273  
 Ren, Zhilei II-229  
 Renau, Quentin II-139  
 Rodionova, Anna II-485  
 Rodriguez-Coayahuitl, Lino II-48  
 Ross, Nicholas II-413  
 Royer, Clément W. II-661  
 Ruberto, Stefano II-63  
  
 Saini, Bhupinder Singh II-243  
 Sakamoto, Naoki I-81  
 Santos, Paulo II-18  
 Savic, Dragan II-413  
 Schäpermeier, Lennart II-154  
 Scoczynski Ribeiro Martins, Marcella II-111  
 Seiler, Moritz I-48  
 Sendhoff, Bernhard I-583, II-428  
 Serebyński, Franciszek I-678  
 Shala, Gresa I-691  
 Shang, Ke I-201, II-257  
 Sheikholeslami, Ali I-317  
 Shirakawa, Shinichi I-719  
 Silva, Sara II-18  
 Srivastava, Rupesh Kumar II-515  
 Suciu, Mihai-Alexandru I-539  
 Sudholt, Dirk I-19  
 Sun, Chaoli I-125  
 Sun, Lei II-271  
 Susmaga, R. I-215  
 Sutton, Andrew M. II-686  
  
 Tamura, Hirotaka I-317  
 Tan, Ying I-125  
 Tanabe, Ryoji I-257  
 Tanaka, Kiyoshi I-33  
 Tang, Ke I-470  
 Tari, Sara I-65  
  
 Terragni, Valerio II-63  
 Teusch, Thomas II-357  
 Teytaud, Olivier I-154, II-661  
 Tiño, Peter I-583  
 Toal, Lauchlan I-184  
 Toklu, Nihat Engin I-623, II-515  
 Tomassini, Marco II-125  
 Tong, Hao II-428  
 Toutouh, Jamal I-552  
 Trautmann, Heike I-48  
 Treude, Christoph II-329  
  
 Uchida, Kento I-719  
  
 Valencia-Rodríguez, Diana Cristina II-285  
 van der Meer, Marjolein C. II-441  
 Varelas, Konstantinos I-707  
 Verel, Sébastien I-33, II-97  
 Virgolin, Marco II-79  
  
 Wagner, Markus II-111, II-329  
 Walker, David J. II-299  
 Walter, Mathew J. II-299  
 Wang, Hao I-169, I-229, II-699  
 Wang, Hui II-528  
 Wang, Wenjing I-567  
 Wang, Yali II-313  
 Wang, Yiwen I-470  
 Whitley, Darrell I-303, II-686  
 Wilde, Dominik I-140  
 Witt, Carsten I-664, II-648  
 Wozniak, Szymon I-418  
  
 Xue, Bing I-3  
  
 Yamaguchi, Teppei I-719  
 Yang, Peng I-470  
 Yao, Xin I-583, II-428, II-454  
 Ye, Furong II-699  
  
 Zaefferer, Martin I-273  
 Zarges, Christine I-390  
 Zăvoianu, Alexandru-Ciprian I-287  
 Zhang, Bofeng I-484  
 Zhang, Guochen I-125  
 Zhang, Han II-454  
 Zhang, Mengjie I-3  
 Zhou, Yuren II-634  
 Zhou, Zhuocheng I-484  
 Zielniewicz, P. I-215