Michael Affenzeller · Stephan M. Winkler ·
Anna V. Kononova · Heike Trautmann ·
Tea Tušar · Penousal Machado ·
Thomas Bäck (Eds.)

LNCS 15150

Parallel Problem Solving from Nature – PPSN XVIII

18th International Conference, PPSN 2024
Hagenberg, Austria, September 14–18, 2024
Proceedings, Part III

Springer

MOREMEDIA

Lecture Notes in Computer Science **15150**

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.

Michael Affenzeller · Stephan M. Winkler ·
Anna V. Kononova · Heike Trautmann ·
Tea Tušar · Penousal Machado · Thomas Bäck
Editors

Parallel Problem Solving from Nature – PPSN XVIII

18th International Conference, PPSN 2024
Hagenberg, Austria, September 14–18, 2024
Proceedings, Part III

Springer

Editors
Michael Affenzeller
University of Applied Sciences Upper Austria
Wels, Austria

Anna V. Kononova
Leiden University
Leiden, The Netherlands

Tea Tušar
Jožef Stefan Institute
Ljubljana, Slovenia

Thomas Bäck
Leiden University
Leiden, The Netherlands

Stephan M. Winkler
University of Applied Sciences Upper Austria
Hagenberg, Austria

Heike Trautmann
University of Paderborn
Paderborn, Germany

Penousal Machado
University of Coimbra
Coimbra, Portugal

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-031-70070-5 ISBN 978-3-031-70071-2 (eBook)
https://doi.org/10.1007/978-3-031-70071-2

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

Preface

Two years ago, in 2022, the international conference on Parallel Problem Solving from Nature (PPSN) returned to where it all started in 1990, namely to Dortmund, Germany. It was great to see that the community had overcome the pandemic and gathered with more than 100 participants attending in person.

On the last day of the conference, during the closing ceremony, we got the chance to propose the University of Applied Sciences Upper Austria (FH OÖ) as organizers and the Softwarepark Hagenberg as the location for PPSN 2024. We were convinced that FH OÖ as the (with respect to research and development) strongest university of applied sciences in Austria could be the ideal choice as host for PPSN 2024, especially as we presented the research group Heuristic and Evolutionary Algorithms Laboratory (HEAL), one of the most active groups in evolutionary algorithms in Austria, as the core group of the organization team. After some weeks, we were delighted to hear from the steering committee that we were chosen as organizers and Hagenberg as the location for this year's edition of PPSN.

We are pleased that a record number of researchers followed our call by submitting their papers for review. We received 294 submissions from which the program chairs selected the top 101 after an extensive peer-review process, which corresponds to an acceptance rate of 34.35%. Not all decisions were easy to make, but we benefited greatly from the careful reviews provided by the international program committee. With an average of 2.86 reviews per paper, most of the submissions received three reviews, while some received two. This led to a total of 840 reviews. Thanks to these reviews, we were able to decide about acceptance on a solid basis.

The papers included in these proceedings were assigned to 12 clusters, entitled *Combinatorial Optimization, Genetic Programming, Fitness Landscape Modeling and Analysis, Benchmarking and Performance Measures, Automated Algorithm Selection and Configuration, Numerical Optimization, Bayesian- and Surrogate-Assisted Optimization, Theoretical Aspects of Nature-Inspired Optimization, (Evolutionary) Machine Learning and Neuroevolution, Evolvable Hardware and Evolutionary Robotics, Multi-objective Optimization and Real-World Applications* which can hardly reflect, the true variety of research topics presented in the proceedings at hand. Following the tradition and spirit of PPSN, all papers were presented as posters. The eight poster sessions consisting of 12 or 13 papers each were compiled orthogonally to the clusters mentioned above to cover the range of topics as widely as possible. As a consequence, participants with different interests would find some relevant papers in every session and poster presenters were able to discuss related work in sessions different from their own.

As usual, the conference started with two days of workshops and tutorials (Saturday and Sunday), followed by three days of poster sessions and invited plenary talks (Monday to Wednesday). We are delighted that three highly renowned researchers from up-and-coming, related research fields accepted our invitation to give a keynote speech, which was be the first item on the program over the three days of the conference.

Two of our keynote speakers are young professors at excellent academic institutions, namely Oliver Schütze (Cinvestav-IPN, Mexico City) and Richard Küng (JKU Linz, Austria); the third keynoter is a researcher at Google Deepmind, namely Bernardino Romera-Paredes, with an equally impressive scientific record.

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

Last but not least, we would like to thank *Softwarepark Hagenberg* and the *University of Applied Sciences Upper Austria* for the donations. We are grateful for the long-standing support of *Springer* to this conference series. Finally, we thank the *RISC Software and Software Competence Center Hagenberg* for providing financial backing.

July 2024

Michael Affenzeller
Stephan M. Winkler
Anna V. Kononova
Heike Trautmann
Tea Tušar
Penousal Machado
Thomas Bäck

Organization

General Chairs

Michael Affenzeller	University of Applied Sciences Upper Austria, Austria
Stephan Winkler	University of Applied Sciences Upper Austria, Austria

Honorary Chair

Hans-Paul Schwefel	TU Dortmund, Germany

Program Committee Chairs

Heike Trautmann	University of Paderborn, Germany
Tea Tušar	Jožef Stefan Institute, Slovenia
Penousal Machado	University of Coimbra, Portugal
Thomas Bäck	Leiden University, Netherlands

Proceedings Chair

Anna V. Kononova	Leiden University, Netherlands

Tutorials Chair

Fabricio Olivetti de França	Federal University of ABC, Brazil

Workshops Chair

Roman Kalkreuth	RWTH Aachen University, Germany

Publicity Chairs

Jan Zenisek	University of Applied Sciences Upper Austria, Austria
Christian Haider	University of Applied Sciences Upper Austria, Austria
Louise Buur	University of Applied Sciences Upper Austria, Austria

Technical Support Chairs

Oliver Krauss	University of Applied Sciences Upper Austria, Austria
Du Nguyen Duy	Software Competence Center Hagenberg, Austria

Steering Committee

Thomas Bäck	Leiden University, Netherlands
David W. Corne	Heriot-Watt University, UK
Carlos Cotta	University of Malaga, Spain
Kenneth De Jong	George Mason University, USA
Gusz E. Eiben	Vrije Universiteit Amsterdam, Netherlands
Bogdan Filipič	Jožef Stefan Institute, Slovenia
Emma Hart Edinburgh	Napier University, UK
Juan Julián Merelo Guervós	University of 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 Birmingham, UK and SUSTech, China

Keynote Speakers

Oliver Schütze	CINVESTAV-IPN, Mexico
Bernardino Romera-Paredes	Google DeepMind London, UK
Richard Küng	Johannes Kepler University Linz, Austria

Program Committee

Michael Affenzeller	University of Applied Sciences Upper Austria, Austria
Hernán Aguirre	Shinshu University, Japan
Imène Ait Abderrahim	University of Djilali Bounaama Khemis Miliana, Algeria
Youhei Akimoto	University of Tsukuba, Japan
Richard Allmendinger	University of Manchester, UK
Marie Anastacio	Leiden University, Netherlands
Claus Aranha	University of Tsukuba, Japan
Dirk Arnold	Dalhousie University, Canada
Anne Auger	Inria, France
Dogan Aydin	Dumlupinar University, Turkey
Jaume Bacardit	Newcastle University, UK
Heder Bernardino	Federal University of Juiz de Fora, Brazil
Hans-Georg Beyer	Vorarlberg University of Applied Sciences, Austria
Martin Binder	Ludwig Maximilian University of Munich, Germany
Mauro Birattari	Université libre de Bruxelles, Belgium
Bernd Bischl	Ludwig Maximilian University of Munich, Germany
Julian Blank	Michigan State University, USA
Aymeric Blot	University College London, UK
Peter Bosman	Centrum Wiskunde & Informatica, Netherlands
Jakob Bossek	University of Paderborn, Germany
Anton Bouter	Centrum Wiskunde & Informatica, Netherlands
Jürgen Branke	University of Warwick, UK
Dimo Brockhoff	Inria, France
Alexander Brownlee	University of Stirling, UK
Larry Bull	University of the West of England, UK
Maxim Buzdalov	Aberystwyth University, UK
Stefano Cagnoni	University of Parma, Italy
Salvatore Calderaro	Palermo University, Italy
Pedro Carvalho	University of Aveiro, Portugal
Josu Ceberio	University of the Basque Country, Spain
Ying-Ping Chen	National Chiao Tung University, Taiwan
Francisco Chicano	University of Malaga, Spain
Miroslav Chlebik	University of Sussex, UK
Sung-Bae Cho	Yonsei University, South Korea
Carlos Coello Coello	CINVESTAV-IPN, Mexico

Jordan Cork	Jožef Stefan Institute, Slovenia
João Correia	University of Coimbra, Portugal
Gabriel Cortês	University of Coimbra, Portugal
Doğan Çörüş	Kadir Has University, Turkey
Ernesto Costa	University of Coimbra, Portugal
Carlos Cotta	University of Malaga, Spain
António Cunha	University of Minho, Portugal
Nguyen Dang	St Andrews University, UK
Kenneth De Jong	George Mason University, USA
Roy de Winter	Leiden University, Netherlands
Kalyanmoy Deb	Michigan State University, USA
Antonio Della Cioppa	University of Salerno, Italy
Antipov Denis	University of Adelaide, Australia
Bilel Derbel	Université de Lille, France
André Deutz	Leiden University, Netherlands
Konstantin Dietrich	TU Dresden, Germany
Benjamin Doerr	Ecole Polytechnique, France
Carola Doerr	Sorbonne University, France
John Drake	University of Leicester, UK
Rafał Dreżewski	AGH University of Science and Technology, Poland
Johann Dreo	Pasteur Institute, France
Paul Dufossé	ID Solutions Oncology, France
Tome Eftimov	Jožef Stefan Institute, Slovenia
Theresa Eimer	Leibniz University Hannover, Germany
Michael Emmerich	Leiden University, Netherlands
Andries Engelbrecht	University of Stellenbosch, South Africa
Anton Eremeev	Dostoevsky Omsk State University, Russia
Richard Everson	University of Exeter, UK
Pedro Ferreira	University of Lisbon, Portugal
Antonino Fiannaca	Italian National Research Council, Italy
Jonathan Fieldsend	University of Exeter, UK
Bogdan Filipič	Jožef Stefan Institute, Slovenia
Steffen Finck	Vorarlberg University of Applied Sciences, Austria
Marcus Gallagher	University of Queensland, Australia
José García-Nieto	University of Málaga, Spain
Mario Giacobini	University of Torino, Italy
Kyriakos Giannakoglou	National Technical University of Athens, Greece
Tobias Glasmachers	Ruhr-Universität Bochum, Germany
Christian Grimme	University of Münster, Germany

Alexander Hagg	Bonn-Rhein-Sieg University of Applied Sciences, Germany
Julia Handl	University of Manchester, UK
Nikolaus Hansen	Inria, France
Jin-Kao Hao	University of Angers, France
Hans Harder	Paderborn University, Germany
Emma Hart	Edinburgh Napier University, UK
Verena Heidrich-Meisner	CAU Kiel, Germany
Jonathan Heins	TU Dresden, Germany
Carlos Henggeler Antunes	University of Coimbra, Portugal
Carlos Ignacio Hernández Castellanos	National Autonomous University of Mexico, Mexico
Ishara Hewa Pathiranage	University of Adelaide, Australia
Martin Holeňa	Czech Academy of Sciences, Czechia
Andoni Irazusta Garrnendia	University of the Basque Country, Spain
Hisao Ishibuchi	Southern University of Science and Technology, China
Christian Jacob	University of Calgary, Canada
Domagoj Jakobović	University of Zagreb, Croatia
Anja Jankovic	RWTH Aachen University, Germany
Thomas Jansen	Aberystwyth University, UK
Laetitia Jourdan	Université de Lille, CRIStAL, CNRS, France
Bryant Julstrom	St. Cloud State University, USA
Timo Kötzing	Hasso Plattner Institute, Germany
Roman Kalkreuth	RWTH Aachen University, Germany
George Karakostas	McMaster University, Canada
Florian Karl	Ludwig Maximilian University of Munich, Germany
Ed Keedwell	University of Exeter, UK
Pascal Kerschke	TU Dresden, Germany
Marie-Eléonore Kessaci	University of Lille, France
Ahmed Kheiri	Lancaster University, UK
Wolfgang Konen	TH Cologne, Germany
Lars Kotthoff	University of Wyoming, USA
Oswin Krause	University of Copenhagen, Denmark
Krzysztof Krawiec	Poznan University of Technology, Poland
Martin S. Krejca	Ecole Polytechnique, France
William B. Langdon	University College London, UK
Manuel López-Ibáñez	University of Manchester, UK
William La Cava	Boston Children's Hospital, USA
Algirdas Lancinskas	Vilnius University, Lithuania
Yuri Lavinas	University of Toulouse, France

Per Kristian Lehre	University of Birmingham, UK
Johannes Lengler	ETH Zurich, Switzerland
Markus Leyser	TU Dresden, Germany
Ke Li	University of Exeter, UK
Arnaud Liefooghe	University of Lille, France
Giosuè Lo Bosco	University of Palermo, Italy
Fernando Lobo	University of Algarve, Portugal
Nuno Lourenço	University of Coimbra, Portugal
Jose A. Lozano	University of the Basque Country, Spain
Rodica Lung	Babes-Bolyai University, Romania
Chuan Luo	Peking University, China
Evelyne Lutton	INRAE, France
Jessica Mégane	University of Coimbra, Portugal
João Macedo	University of Coimbra, Portugal
Mikel Malagón	University of the Basque Country, Spain
Katherine Malan	University of South Africa, South Africa
Vittorio Maniezzo	University of Bologna, Italy
Valentin Margraf	Ludwig Maximilian University of Munich, Germany
Luis Martí	Center Inria Chile, Chile
Jörn Mehnen	University of Strathclyde, UK
Marjan Mernik	University of Maribor, Slovenia
Olaf Mersmann	Federal University of Applied Administrative Sciences, Germany
Silja Meyer-Nieberg	Bundeswehr University Munich, Germany
Efrén Mezura-Montes	University of Veracruz, Mexico
Krzysztof Michalak	Wroclaw University of Economics, Poland
Kaisa Miettinen	University of Jyväskylä, Finland
Edmondo Minisci	University of Strathclyde, UK
Gara Miranda Valladares	University of La Laguna, Spain
Mustafa Misir	Duke Kunshan University, China
Marco Montes de Oca	EnFi Inc. and Northeastern University, USA
Hugo Monzón	RIKEN, Japan
Mario Andrés Muñoz	University of Melbourne, Australia
Boris Naujoks	TH Cologne, Germany
Antonio J. Nebro	University of Málaga, Spain
Ferrante Neri	University of Surrey, UK
Aneta Neumann	University of Adelaide, Australia
Frank Neumann	University of Adelaide, Australia
Michael O'Neill	University College Dublin, Ireland
Gabriela Ochoa	University of Stirling, UK
Pietro S. Oliveto	University of Sheffield, UK

Una-May O'Reilly	Massachusetts Institute of Technology, USA
José Carlos Ortiz-Bayliss	Monterrey Institute of Technology and Higher Education, Mexico
Patryk Orzechowski	University of Pennsylvania, USA
Ender Özcan	University of Nottingham, UK
Ben Paechter	Edinburgh Napier University, UK
Gregor Papa	Jožef Stefan Institute, Slovenia
Luís Paquete	University of Coimbra, Portugal
Andrew J. Parkes	University of Nottingham, UK
Sebastian Peitz	Paderborn University, Germany
Kokila Kasuni	Perera University of Adelaide, Australia
Stjepan Picek	Radboud University, Netherlands
Martin Pilát	Charles University, Czechia
Nelishia Pillay	University of Pretoria, South Africa
Petr Pošík	Czech Technical University in Prague, Czechia
Raphael Patrick Prager	University of Münster, Germany
Oliver Preuß	Paderborn University, Germany
Mike Preuss	Leiden University, Netherlands
Michal Przewozniczek	Wroclaw University of Science and Technology, Poland
Chao Qian	Nanjing University, China
Günther Raidl	Vienna University of Technology, Austria
Elena Raponi	Leiden University, Netherlands
Khaled Rasheed	University of Georgia, USA
Alma Rahat	Swansea University, UK
Piotr Ratuszniak Koszalin	University of Technology, Poland
Tapabrata Ray	University of New South Wales, Australia
Quentin Renau	Edinburgh Napier University, UK
Riccardo Rizzo	Harvard University, USA
Angel Rodriguez-Fernandez	CINVESTAV-IPN, Mexico
Eduardo Rodriguez-Tello	CINVESTAV-IPN, Mexico
Andrea Roli	University of Bologna, Italy
Jeroen Rook	University of Twente, Netherlands
Jonathan Rowe	University of Birmingham, UK
Günter Rudolph	TU Dortmund, Germany
Conor Ryan	University of Limerick, Ireland
Saba Sadeghi Ahouei	University of Adelaide, Australia
Daniela Santos	Lutheran University of Brazil, Brazil
Frédéric Saubion	University of Angers, France
Lennart Schäpermeier	TU Dresden, Germany
Robert Schaefer	AGH University of Science and Technology, Poland

Andrea Schaerf	University of Udine, Italy
Larissa Schmid	Karlsruhe Institute of Technology, Germany
Lennart Schneider	Ludwig Maximilian University of Munich, Germany
Marc Schoenauer	Inria, France
Renzo Scholman	Centrum Wiskunde & Informatica, Netherlands
Oliver Schuetze	CINVESTAV-IPN, Mexico
Moritz Seiler	Paderborn University, Germany
Bernhard Sendhoff	Honda Research Institute Europe, Germany
Roman Senkerik	Tomas Bata University, Czechia
Marc Sevaux	University of South Brittany, France
Hadar Shavit	RWTH Aachen University, Germany
Ofer 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
Konstantin Sonntag	Paderborn University, Germany
Giovanni Squillero	Politecnico di Torino, Italy
Sebastian Stich	CISPA Helmholtz Center for Information Security, Germany
Catalin Stoean	University of Craiova, Romania
Thomas Stützle	Université libre de Bruxelles, Belgium
Mihai Suciu	Babes-Bolyai University, Romania
Dirk Sudholt	University of Sheffield, UK
Andrew Sutton	University of Minnesota, USA
Urban Škvorc	Paderborn University, Germany
Ricardo Takahashi	Federal University of Minas Gerais, Brazil
Sara Tari	University of the Littoral Opal Coast, France
Daniel Tauritz	Auburn University, USA
Dirk Thierens	Utrecht University, Netherlands
Kevin Tierney	Bielefeld University, Germany
Renato Tinós	University of São Paulo, Brazil
Marco Tomassini	University of Lausanne, Switzerland
Alberto Tonda	INRAE, France
Jamal Toutouh	Massachusetts Institute of Technology, USA
Kento Uchida	Yokohama National University, Japan
Ryan J. Urbanowicz	University of Pennsylvania, USA
Niki van Stein	Leiden University, Netherlands
Nadarajen Veerapen	University of Lille, France
Filippo Vella	National Research Council, Italy
Sébastien Verel	University of the Littoral Opal Coast, France
Diederick Vermetten	Leiden University, Netherlands

Anh Viet Do	University of Adelaide, Australia
Adriano Vinhas	University of Coimbra, Portugal
Markus Wagner	University of Adelaide, Australia
Hanyang Wang	Huawei Technologies, UK
Hao Wang	Leiden University, Netherlands
Elizabeth Wanner	CEFET, Brazil
Tobias Weber	Otto von Guericke University Magdeburg, Germany
Thomas Weise	Hefei University, China
Marcel Wever	Ludwig Maximilian University of Munich, Germany
Darrell Whitley	Colorado State University, USA
Dennis Wilson	University of Toulouse, France
Carsten Witt	Technical University of Denmark, Denmark
Man Leung Wong	Lingnan University, Hong Kong, China
Kaifeng Yang	University of Applied Sciences Upper Austria, Austria
Shengxiang Yang	De Montfort University, UK
Furong Ye	Leiden University, Netherlands
Martin Zaefferer	DHBW Ravensburg, Germany
Aleš Zamuda	University of Maribor, Slovenia
Saúl Zapotecas-Martínez	INAOE, Mexico
Christine Zarges	Aberystwyth University, UK
Mengjie Zhang	Victoria University of Wellington, New Zealand

Contents – Part III

Theoretical Aspects of Nature-Inspired Optimization

Theoretical Aspects of Nature-Inspired Optimization

Self-adjusting Evolutionary Algorithms are Slow on a Class of Multimodal Landscapes

Johannes Lengler and Konstantin Sturm(✉)

Department of Computer Science, ETH Zürich, Zürich, Switzerland
{johannes.lengler,konstantin.sturm}@inf.ethz.ch

Abstract. The one-fifth rule and its generalizations are a classical parameter control mechanism in discrete domains. They have also been transferred to control the offspring population size of the $(1, \lambda)$-EA. This has been shown to work very well for hill-climbing, and combined with a restart mechanism it was recently shown by Hevia Fajardo and Sudholt to improve performance on the multi-modal problem Cliff drastically. In this work we show that the positive results do not extend to other types of local optima. On the distorted OneMax benchmark, the self-adjusting $(1, \lambda)$-EA is slowed down just as elitist algorithms because self-adaptation prevents the algorithm from escaping from local optima. This makes the self-adaptive algorithm considerably worse than good static parameter choices, which do allow to escape from local optima efficiently. We show this theoretically and complement the result with empirical runtime results.

Keywords: evolutionary algorithm · comma selection · parameter control · population size · one-fifth rule · fixed-target · runtime analysis

1 Introduction

Evolutionary algorithms (EAs) are a class of randomized optimization heuristics that are popular because they are flexible and can be widely applied. It is desirable for such general-purpose optimization algorithms to be as easy to use as possible. Thus, an important goal in designing EAs is to reduce the number of hyper-parameters that need to be set by the user. A convenient way is to make the algorithms *self-adjusting*, i.e., to add mechanisms that dynamically adapt the hyper-parameters in an automatic way. This approach has some other advantages. Sometimes there is no static parameter setting which is optimal throughout the whole optimization process, in which case self-adjusting mechanisms can be superior [3,4,7].

A self-adaptation mechanism that has received increasing attention in recent years is the *one-fifth rule* and its generalization, the $(1 : s + 1)$-rule. This is a classical rule in the domain of continuous optimization [15], but in the last years

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 3–18, 2024.
https://doi.org/10.1007/978-3-031-70071-2_1

it has also been successfully transferred to discrete domains [3,6,7,10–12], see also the reviews in [2] and [6]. The $(1 : s+1)$ rule may be used to control a hyper-parameter that regulates the trade-off between efficiency and the *success rate*, which is the probability of making an improvement in one generation. It defines a *target success rate*, which is $1/s$ in the case of the $(1 : s+1)$ rule. Then, whenever a generation is successful it adapts the hyper-parameter to improve efficiency at the cost of a smaller success rate. For unsuccessful generations, it adapts the hyper-parameter in the other direction. Both adjustments are balanced in such a way that the success rate is pushed toward the target success rate. Some hyper-parameters for which this rule has been shown to work particularly well are the *step size* in continuous optimization [13], the *mutation rate* in discrete domains [4], and the *offspring population size* for hill-climbing tasks [6,11].

This work will focus on the offspring population size, specifically the SA-$(1, \lambda)$-EA. The algorithm generates λ offspring from the same parent in each generation, and proceeds the best offspring as parent for the next generation. It adapts the offspring population size λ with the $(1 : s + 1)$ rule, see Sect. 2 for details. Recently, some very positive results could be shown for this algorithm. Hevia Fajardo and Sudholt studied the SA-$(1, \lambda)$-EA on OneMax[1], a benchmark in which progress gets harder during the optimization process. They could show that for $s \leq 1$, the $(1 : s+1)$ rule automatically chooses and maintains the optimal λ throughout optimization, ranging from constant λ at the beginning to almost linear λ as the algorithm approaches the optimum. Kaufmann, Larcher, Lengler, and Zou extended this result (for smaller s) to all monotonic functions, showing that the SA-$(1, \lambda)$-EA shows optimal parameter control on every monotonic function. These results show that the SA-$(1, \lambda)$-EA can be very successful on hill-climbing tasks without local optima.

In principle, the $(1, \lambda)$-EA is also well-suited to deal with local optima. In fact, the comma strategy allows the $(1, \lambda)$-EA to escape local optima by "forgetting" the parent, other than its elitist counterpart $(1 + \lambda)$-EA, in which the parent always competes for entering the next generation. Indeed, this makes the $(1, \lambda)$-EA more efficient than the $(1+\lambda)$-EA in landscapes with planted local optima [9]. However, a priori the $(1 : s + 1)$ rule is misaligned with this escaping option. When the algorithm is stuck in a local optimum, the $(1 : s + 1)$ rule starts increasing the offspring population size, which is the correct behavior for hill-climbing. However, this also increases the probability of producing a clone of the parent among the offspring, in which case the algorithm mimics the behavior of the plus strategy and loses its ability to escape local optima. When λ is of logarithmic size or larger, the $(1, \lambda)$-EA degenerates into the $(1 + \lambda)$-EA. Thus, we can not hope that the SA-$(1, \lambda)$-EA may be suited for local optima in its standard form.

To avoid this problem, Hevia Fajardo and Sudholt proposed as a simple fix to *restart* the offspring population size at $\lambda = 1$ whenever it exceeds some threshold $\lambda_{\max}$. In a spectacular result for the notoriously hard benchmark Cliff, which

[1] OneMax is defined on the hypercube $\{0, 1\}^n$ and assigns to each bit string x the number of one-bits in x.

features a large plateau of local optima, they could show that the SA-$(1,\lambda)$-EA with resets optimizes cliff with $O(n \ln n)$ function evaluations, not substantially slower than OneMax [7]. This is not only much better than any known performance of elitist algorithms on Cliff, but it is also drastically faster than the $(1,\lambda)$-EA with any *static* parameter λ, which needs time $\Omega(n^{3.98})$ even for optimally chosen static λ [7]. These results gave hope that the SA-$(1,\lambda)$-EA with resets may be able to provide optimal strategies for a wide range of fitness landscapes. Unfortunately, in this paper we show that this algorithm has some severe limitations when the local optima are not clustered in form of a large cliff, but rather scattered throughout the fitness landscape. While we do believe that the SA-$(1,\lambda)$-EA with resets deserves its place in modern optimization portfolios, our result shows that it is no panacea.

1.1 Our Result

We study the SA-$(1,\lambda)$-EA with resets on the *distorted OneMax* benchmark disOM in a fixed-target setting. The function disOM is obtained from the OneMax benchmark by increasing the fitness of each search point with some probability p by some value $d > 1$, thus planting local optima at random places of the landscape. For the formal definition, see Sect. 2. We mostly take the parameters of disOM from [9]; in fact, we even allow slightly more general parameters. In particular, we choose $p = \omega(1/n \ln n)$ to make sure that the algorithms encounter distorted points during optimization, and we choose the fixed target in such a way that the target can be reached efficiently with some static values of λ, see [9] for a more thorough discussion.

In [9] it was shown that the $(1+\lambda)$-EA is slowed down by a factor of $1/p$, yielding runtime $\Omega(n \ln n/p)$. This can be substantial since $1/p$ may be an almost linear factor. The algorithm is slowed down because the plus strategy is not able to escape local optima and thus needs to hop from one local optimum to the next. This makes it by a factor p harder to find an improvement since the algorithm does not only need to create an offspring of larger OneMax value, but in addition this offspring must be distorted. On the other hand, the $(1,\lambda)$-EA with static λ is unaffected because it can efficiently escape from local optima and has the same runtime $O(n \ln n)$ as for the corresponding OneMax problem.

We show that the SA-$(1,\lambda)$-EA with resets suffers the same performance loss as the $(1+\lambda)$-EA on disOM: it needs time $\Omega(n \ln n/p)$ to reach the fitness target. Thus, the self-adjusting mechanism costs performance and slows down the algorithm by a factor of $1/p$ compared to the known runtime $O(n \log n)$ of the $(1,\lambda)$-EA with *static* λ [9].

Theorem 1. *Consider the SA-$(1,\lambda)$-EA with a resetting mechanism for the offspring population size on* disOM *with $p = \omega(1/n \ln n)$, $d = \Omega(\ln n)$, and $\lambda_{max} \geq n^{\Omega(1)}/p$. With high probability the algorithm takes $\Omega(n \ln n/p)$ function evaluations to reach a target fitness of $n - k^*$ for $k^* = n^{1-\Omega(1)}$.*

We put this into context with the known results on the $(1+\lambda)$-EA and the $(1,\lambda)$-EA. The $(1,\lambda)$-EA with an optimal *static* λ is faster than both the self-adjusting one and the $(1+\lambda)$-EA in the specific setting presented in [9]. The following Corollary summarizes these findings.

Corollary 2. *Let* $k^* = n^{\Omega(1)} \cap n^{1-\Omega(1)}$, $p = \omega(1/(n \ln n))$, *and assume that there is a constant* $\varepsilon > 0$ *such that*

$$p \leq (k^*/n)^{1+\varepsilon}. \tag{1}$$

Finally, assume that $d = \Omega(\ln n)$ *with* $d \leq k^*$ *and* $\lambda_{\max} = n^{\Omega(1)}$. *Then w.h.p. on* DISOM *the number of evaluations to reach fitness at least* $n - k^*$ *is*

1. $\Omega(n \ln n/p)$ *for the* $(1+\lambda)$*-EA with any static* $\lambda \geq 1$,
2. $\Omega(n \ln n/p)$ *for the SA-*$(1,\lambda)$*-EA,*
3. $O(n \ln n)$ *for the* $(1,\lambda)$*-EA with a suitable static* $\lambda = \Theta(\ln n)$.

The difference is a factor of $1/p$, which can be substantial, for example settings with $1/p = \Omega(n)$ are included. The conditions come directly from [9, Assumption 1.4], and the third statement comes from [9, Theorem 1.1]. We note that a sufficient condition for λ to make the $(1,\lambda)$-EA efficient was given in [9] as λ which satisfy $(1+\delta)\log_{e/(e-1)}(1/p) \leq \lambda \leq (1-\delta)\log_{e/(e-1)}(n/k^*)$ for an arbitrary constant $\delta > 0$.[2]

We further corroborate the theoretical findings with some empirical results, which are presented in Sect. 5. Those show quite clearly an asymptotic of $\Theta(n \ln n/p)$ for the SA-$(1,\lambda)$-EA, indicating that our lower bound in Theorem 1 is tight. Moreover, they confirm the asymptotic statement from Corollary 2 that the $(1,\lambda)$-EA with static λ is much faster than both the $(1+\lambda)$-EA with static λ and the SA-$(1,\lambda)$-EA.

We want to emphasize that we do not advertise abolishing self-adaptation and returning to static parameter choices. While we show that there are regimes in which the self-adjusting algorithm is slow, there are also other regimes where static choices have disadvantages. In particular, if the target fitness is large (e.g., $k^* = 0$) then there is no static λ which can reach the target fitness and at the same time avoid being stuck in local optima for long. Thus, future research should aim for alternatives which can avoid the downsides from both approaches. Moreover, for practical matters, general-purpose optimizers like EAs should always be used as part of a portfolio of optimization techniques which does not hinge on a single algorithm.

2 Notation and Preliminaries

General Notation. We write $[n] := \{1, ..., n\}$. Search points are denoted by $x = (x_1, ..., x_n) \in \{0,1\}^n$, and the ONEMAX value is $\mathrm{OM}(x) := \sum_{i \in [n]} x_i$. The

[2] The authors of [9] comment that the second condition may not be necessary. For more details on the parameters we refer to the discussion in [9].

ZeroMax function is defined as $\mathrm{ZM}(x) := n - \mathrm{OM}(x)$. We denote the all-one-string by $\vec{1} = (1, ..., 1)$. For $x, y \in \{0,1\}^n$, the Hamming distance $H(x, y)$ of x and y is the number of positions $i \in [n]$ such that $x_i \neq y_i$. We denote the natural logarithm of n by $\ln n$. With high probability (w.h.p.) means with probability $1 - o(1)$ for $n \to \infty$. For a real number a, we denote by $\lfloor a \rceil := \lfloor a + 1/2 \rfloor$ the closest integer to a.

Distorted OneMax. The function is formally defined as $\textsc{disOM} : \{0,1\}^n \to \mathbb{R}_{\geq 0}$. We partition the search space $\{0,1\}^n$ into two sets $\mathcal{C}$ and $\mathcal{D}$ of "clean" and "distorted" points, respectively. For each $x \in \{0,1\}^n$ we have $x \in \mathcal{D}$ with probability p and $x \in \mathcal{C}$ otherwise, independently of the other points. We define disOM as

$$\textsc{disOM}(x) := \mathrm{OM}(x) + \begin{cases} d & \text{if } x \in \mathcal{D}, \\ 0 & \text{otherwise.} \end{cases}$$

The function was introduced in [9], where it was shown that plus strategies are slowed down by a factor of $1/p$, while comma strategies are not affected. Very recently, it was shown that this effect is even more drastic when the height of the distortion is drawn randomly for every distorted point, making the plus strategies super-polynomially slow [14]. However, we will follow the setup in [9] and use the same offset d for all distorted points.

Algorithms. We consider the SA-$(1, \lambda)$-EA with resets, which is identical to the one presented in [7]. The algorithm is called self-adjusting as the offspring population size λ is adapted in each generation. A fitness increase results in a decrease of λ to λ/F for some $F > 1$. If the fitness stays the same or decreases, the generation is called unsuccessful, and λ is increased to $\lambda \cdot F^{1/s}$ for some $s > 0$. Throughout the paper we will assume that F, s are constant and $0 < s < 1$. When a sequence of s successful generations and a single unsuccessful one occur, the final offspring population size is unchanged due to $\lambda \cdot (F^{1/s})^s \cdot (1/F) = \lambda$, which follows the previously mentioned $(1 : s+1)$-success rule [13]. The "reset part" in the algorithm's name refers to the maximum offspring population size $\lambda_{\max}$ we impose. If a generation with $\lambda = \lambda_{\max}$ is unsuccessful, the offspring population size is reset to 1 instead of being increased further.

New offspring of a search point x are created by applying a *standard bit mutation*: Each bit in x is being flipped independently with probability $1/n$. We consider a fixed target setting of $n - k^*$ following [9].

3 Properties of the SA-$(1, \lambda)$-EA

In this section, we provide a series of useful probability estimates, most of which are not specific to our benchmark algorithm combination and may prove useful in other settings as well.

We call an offspring a *clone* of the parent if it is an exact copy. The first lemma provides bounds on (not) having a clone among the offspring. Similar versions of the results can be found in the proof of [7, Lemma 4.5] and [6, Lemma 2.2].

Algorithm 1: Self-adjusting $(1, \lambda)$ EA resetting λ for maximizing f to target $n - k^*$.

Initialization: $t = 0$; choose $x_0 \in \{0,1\}^n$ uniformly at random and $\lambda_0 = 1$;
Optimization: **while** $f(x_t) < n - k^*$ **do**
  Mutation: **for** $i \in \{1, ..., \lfloor \lambda_t \rceil\}$ **do**
    $y_t^i \in \{0,1\}^n \leftarrow$ standard bit mutation(x_t);
  Selection: Let $y_t = \arg\max\{f(y_t^1), ..., f(y_t^{\lfloor \lambda_t \rceil})\}$, breaking ties uniformly at random;
  Update: $x_{t+1} \leftarrow y_t$;
  if $f(x_{t+1}) > f(x_t)$ **then** $\lambda_{t+1} \leftarrow \max\{\lambda_t/F, 1\}$;
  if $f(x_{t+1}) \le f(x_t) \wedge \lambda_t = \lambda_{\max}$ **then** $\lambda_{t+1} \leftarrow 1$;
  if $f(x_{t+1}) \le f(x_t) \wedge \lambda_t \ne \lambda_{\max}$ **then** $\lambda_{t+1} \leftarrow \min\{\lambda F^{1/s}, \lambda_{\max}\}$; $t \leftarrow t + 1$;

Lemma 3. *The probability of not having a clone of the parent among λ_t offspring is at least $(1 - 1/e)^{\lambda_t}$. The probability of having at least one clone among the offspring is at least* $\exp\left(-en/(\lambda_t(n-1))\right)$.

Proof. The probability that a single offspring is a clone is $(1-1/n)^n$. The probability that none of the offspring is a clone is therefore at least

$$\left(1 - \left(1 - \frac{1}{n}\right)^n\right)^{\lambda_t} \ge \left(1 - \frac{1}{e}\right)^{\lambda_t}.$$

The probability of at least one clone is at least

$$1 - \left(1 - \left(1 - \frac{1}{n}\right)^n\right)^{\lambda_t} \ge 1 - \left(1 - \frac{n-1}{en}\right)^{\lambda_t} \ge 1 - e^{-\lambda_t(n-1)/en}$$
$$\ge 1 - \frac{1}{1 + \lambda_t(n-1)/en} = \frac{1}{1 + en/(\lambda_t(n-1))} \ge \exp\left(-\frac{en}{\lambda_t(n-1)}\right).$$

□

The next result is more specific to DISOM. It will allow us to argue that if each generation has a distorted point among its offspring, the algorithm will not leave the set of distorted points $\mathcal{D}$.

Lemma 4. *Let the size of the distortion be $d = \Omega(\ln n)$. For any constant $c > 0$, if any of the $\lambda = n^c$ offspring is distorted, with probability $1 - n^{-\omega(1)}$ the accepted offspring will also be distorted.*

Proof. For a clean offspring to be accepted, its fitness must be larger than that of the distorted point. Since $d \in \Omega(\ln n)$, either the distorted or the clean point must have Hamming distance $\Omega(\ln n)$ to the parent. By [9, Lemma 3.3] the probability of a single offspring satisfying this is $n^{-\Omega(\ln \ln n)}$. With a union bound over the n^c offspring, the lemma follows. □

A major complication in runtime analyses of algorithms on DISOM is the fact that the noise of the benchmark is frozen [9,14]. This means that when we sample an offspring, we can not simply assume that it is distorted with probability p since it may have been sampled earlier. The probability of having been sampled may depend on the history of the run. This may seem like a technicality, but in fact it may have a major impact on the resulting runtimes, see [9] for a discussion. Following a technique invented in [14], we prove the following lemma to show that enough neighbours of the current search point have not been queried yet and thus provide "fresh randomness".

Lemma 5. *Consider any algorithm that creates offspring from previously visited search points with standard bit mutation of mutation rate $1/n$ on any fitness function. Assume the algorithm has created $o(n^3)$ offspring so far. Let $x \in \{0,1\}^n$ be any search point. Then the probability that a random offspring of x has not yet been queried is $\Omega(1)$. In particular, for* DISOM *the probability that this offspring is distorted is $\Omega(p)$.*

Proof. The probability of an offspring having a Hamming distance of exactly 3 to the parent is

$$\binom{n}{3}\left(\frac{1}{n}\right)^3\left(1-\frac{1}{n}\right)^{n-3} \geq \left(\frac{n}{3}\right)^3\left(\frac{1}{n}\right)^3\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{1}{27e} = \Omega(1).$$

For a parent, there are $\binom{n}{3}$ points in the three-neighborhood. Since we assumed $o(n^3)$ points have been queried so far, each new offspring has not been sampled before with probability $\Omega(1)$, conditional on having Hamming distance three to the parent. Together, this implies that with probability $\Omega(1)$ each new offspring has not been sampled before. The second claim follows immediately because a search point that has not been sampled before is distorted with probability p. □

The final lemma enables us to contend that a significant drop in λ is unlikely in certain settings. It is closely related to the Gambler's Ruin Problem [5].

Lemma 6. *Consider the SA-$(1,\lambda)$-EA on* DISOM *in a state (x_0,λ_0) with $\lambda_0 \in [\alpha F^{\beta-1}, \alpha F^{\beta})$, for some $\alpha \geq 1$ and $\beta \in \mathbb{N}$. Suppose there is $q > 0$ and a set $X \subseteq \{0,1\}^n \times [1,\lambda_{\max}]$ of states such that for any state $(x_t,\lambda_t) \in X$, the probability that the next generation is successful is at most q. Then the probability that λ falls below α before either the algorithm leaves X or it increases λ to at least αF^{β} is at most $(1/q-2)/((1/q-1)^{\beta+1}-1)$.*

Proof. We approach the problem from a random walk perspective and aim to build a connection to the Gambler's Ruin problem [5]. Let us define the states $S_0, S_1, ..., S_{\beta+1}$. The algorithm is in state S_i if the current offspring population size λ_t is in the interval $[\alpha F^{i-1}, \alpha F^i)$, and x_t is arbitrary in X. Notice that when in S_i, i consecutive successful generations reduce the offspring population size from λ_t to $\lambda_t F^{-i} < \alpha F^i F^{-i} = \alpha$. Let P_i denote the probability of the algorithm reaching the state S_0 before leaving X or reaching $S_{\beta+1}$. Clearly $P_0 = 1$ and

$P_{\beta+1} = 0$. We proceed to compute P_i for $1 \leq i \leq \beta$. We will pessimistically assume that the algorithm does not leave the set X.

If the current state is S_i and we encounter a successful generation, the algorithm moves to S_{i-1}. Conversely, if the generation is not successful λ is multiplied by $F^{1/s}$ resulting in a move to a state S_k with $k > i$. By the bound q on the probability of successful generation, we can bound P_i recursively[3] as

$$P_i \leq qP_{i-1} + (1-q)P_k.$$

Pessimistically assuming $k = i+1$ and equality, we obtain the classical recursion for some upper bound $\tilde{P}_i \geq P_i$,

$$\tilde{P}_i = q\tilde{P}_{i-1} + (1-q)\tilde{P}_{i+1}.$$

The above equation is the recursion for the Gambler's Ruin Problem, which has the following solution [5] for $0 < i \leq \beta$:

$$\tilde{P}_i = \frac{1 - ((1-q)/q)^{\beta+1-i}}{1 - ((1-q)/q)^{\beta+1}}.$$

The lemma assumes the offspring population size is in the interval $[\alpha F^{\beta-1}, \alpha F^{\beta})$. Thus, recalling P_β represents the measure we are looking for:

$$P_\beta \leq \tilde{P}_\beta = \frac{1 - ((1-q)/q)}{1 - ((1-q)/q)^{\beta+1}} = \frac{1/q - 2}{(1/q-1)^{\beta+1} - 1}$$

□

4 Lower Runtime Bounds

In this section, we prove Theorem 1. The core idea is to show that the algorithm must traverse a OneMax-interval I of size n^ε and that it requires $\Omega(n \ln n/p)$ evaluations to do so. The $\Omega(1/p)$ factor stems from the observation that the algorithm will stay among the distorted points while crossing the interval.

To ensure that the algorithm stays among search points in $\mathcal{D}$ throughout the interval I, we need to show that the algorithm enters it in a distorted point. In light of this, we look at a preceding interval I' of Hamming distances to $\vec{1}$ and show that the algorithm reaches a state in I' such that the search point is distorted and λ is at least logarithmic in size.

[3] It could happen that the P_i are not increasing due to the set X, in which case the bound may not hold. However, the recursion is correct if we replace P_i with the minimal probability over all possible search spaces that satisfy the condition of the lemma because those probabilities are increasing. We suppress this complication from the proof.

Lemma 7. *Consider the SA-$(1,\lambda)$-EA with resets on* DISOM *as in Theorem 1. Let $\varepsilon > 0$ be a small constant such that $\lambda_{\max} \geq n^{\varepsilon}$ and $k^* + d \leq n^{1-\varepsilon}$, and let $I' := [n^{1-\varepsilon/2}, n^{1-\varepsilon/4}]$ be an interval of Hamming distances to $\vec{1}$. Then with high probability the algorithm will either make $\Omega(n \ln n/p)$ evaluations before reaching a search point of distance smaller than $n^{1-\varepsilon/2}$ from $\vec{1}$, or it reaches a state (x_t, λ_t) where x_t is in I' and distorted, and $\lambda_t \geq 6eF^{16/\varepsilon} \ln n$.*

Proof. The lemma holds trivially if the algorithm takes $\Omega(n \ln n/p)$ evaluations to cross I'. Going forward, we thus assume the algorithm makes $o(n \ln n/p)$ evaluations. We show that w.h.p.,

(i) λ is increased to at least $\gamma := 6eF^{32/\varepsilon} \ln n$ in $O(\ln^2 n)$ evaluations if the algorithm stays in I',
(ii) if $\lambda \geq \gamma$ it will not drop below $6eF^{16/\varepsilon} \ln n$ before entering a distorted point, and
(iii) the algorithm makes $\Omega(n \ln n)$ evaluations in consecutive generations in I'.

From these three items, the lemma follows. By (i), λ is increased to γ in $O(\ln^2 n)$ evaluations. Once the algorithm has reached this offspring population size, w.h.p. it will not drop below $6eF^{16/\varepsilon} \ln n$ according to (ii). Together with (iii), this implies $\Omega(n \ln n) - O(\ln^2 n) = \Omega(n \ln n)$ evaluations are from states in which the offspring population size is at least $6eF^{16/\varepsilon} \ln n$. By Lemma 5, each of these offspring have probability $\Omega(p) = \omega(1/(n \ln n))$ to be distorted. Hence, the expected number of distorted points among these offspring is $\Omega(p \cdot n \ln n) = \omega(1)$. By Lemma 4, such an offspring is accepted w.h.p..

It remains to prove the three items, starting with (i). Let λ_t be the current offspring population size. Using a similar argumentation to the proof of [7, Lemma 4.6], $\lceil s \log_F(\gamma/\lambda_t) \rceil$ consecutive unsuccessful generations are sufficient to ensure an increase in the offspring populations size to at least γ. Using $\lfloor x \rceil \leq 2x$ for $x \geq 1$, the number of evaluations this requires is at most

$$\sum_{i=0}^{\lceil s \log_F(\gamma/\lambda_t) \rceil} \left\lfloor \lambda_t F^{i/s} \right\rceil \leq 2\lambda_t \sum_{i=0}^{s \log_F(\gamma/\lambda_t)+1} F^{i/s}$$
$$\leq 2\lambda_t \frac{\left(F^{1/s}\right)^{s \log_F(\gamma/\lambda_t)+2} - 1}{F^{1/s} - 1} \leq \frac{2F^{2/s}}{F^{1/s} - 1} \gamma = O(\ln n).$$

To show that we only encounter unsuccessful generations until λ is increased to the desired value, we show that w.h.p. none of the $O(\ln n)$ offspring increase the fitness.

We first consider the case that no distorted offspring is created from a clean parent. Let p_{imp} be the probability of a single offspring increasing its fitness. $p_{\text{imp}} \leq n^{-\varepsilon/4}$, since at least one of the at most $n^{1-\varepsilon/4}$ zero-bits must be flipped. With a union bound over the $O(\ln n)$ offspring, the probability of this is $O(\ln n) \cdot p_{\text{imp}} = o(1)$.

We turn to the case in which the algorithm creates a distorted offspring from a clean parent before λ is increased to at least γ. Assume the algorithm has just

jumped from a clean to a distorted point. Now, it either has to make another such jump within the next $O(\ln n)$ evaluations, or w.h.p. the algorithm increases λ to at least γ, using the same argumentation as before. For another jump, the algorithm must first leave the set of distorted points $\mathcal{D}$ again. By showing that with probability $\Omega(1)$ the algorithm will not leave $\mathcal{D}$ once entered, w.h.p. after $O(\ln n)$ jumps from clean to distorted points, the algorithm increases the offspring population size to at least γ. This process takes $O(\ln^2 n)$ evaluations.

It remains to show that the algorithm will stay in the distorted points with probability $\Omega(1)$. A sufficient condition is that each generation has a clone among its offspring. With Lemma 3, the probability of this is at least

$$\prod_{i=0}^{\lceil s \log_F(\gamma/\lambda_t)\rceil} \exp\left(-\frac{en}{\lambda_t F^{i/s}(n-1)}\right) \geq \exp\left(-\frac{en}{(n-1)}\sum_{i=0}^{\infty} F^{-i/s}\right)$$
$$\geq \exp\left(-\frac{en}{(n-1)}\frac{1}{1-F^{-1/s}}\right) = \Omega(1).$$

In order to show (ii), note that the offspring population size can drop below $\ln n$ in two different ways. Either the algorithm encounters a reset by increasing λ beyond λ_{max}, or on the "natural way" by a series of successful generations decreasing the offspring population size. We begin by showing that w.h.p. the algorithm does not encounter a reset. For a reset, a generation with offspring population size λ_{max} must be unsuccessful, and therefore, no offspring can increase the OneMax-value. The probability of a single offspring y increasing the number of one-bits is at least $\mathrm{ZM}(y)/(en) \leq n^{1-\varepsilon/4}/(en)$ [6, Lemma 2.2]. Using that $\lambda_{max} \geq n^{\varepsilon}$, the probability of this is at most

$$\left(1-\frac{n^{1-\varepsilon/4}}{en}\right)^{\lambda_{max}} \leq \exp\left(-\frac{\lambda_{max}}{en^{\varepsilon/4}}\right) \leq \exp\left(-\frac{n^{\varepsilon}}{en^{\varepsilon/4}}\right) = o\left(1/n^3\right).$$

Recall that we assumed the algorithm makes $o(n \ln n/p)$ evaluations. Hence, the algorithm does not encounter a reset w.h.p..

It remains to show that λ does not "naturally" drop below $6eF^{16/\varepsilon}\ln n$. We introduce the notion of a *phase*. A phase starts as soon as λ is reduced below γ and ends if it is either increased back to at least γ or falls below $6eF^{16/\varepsilon}\ln n$. We assume the algorithm does not sample a distorted point during a phase as otherwise, the lemma follows immediately. We want to apply Lemma 6 with $\alpha := 6eF^{16/\varepsilon}\ln n$ and $\beta := 16/\varepsilon$. To bound the probability of a successful generation, notice that each generation creates at most γ offspring and the number of zero-bits is at most $n^{1-\varepsilon/4}$. With a union bound, the probability that a single generation is successful is therefore at most $\gamma n^{1-\varepsilon/4}/n = \gamma n^{-\varepsilon/4} =: q$. By Lemma 6, the probability of a phase ending due to drop in λ below α is at most

$$\frac{1/q-2}{(1/q-1)^{\beta+1}-1} = \frac{\gamma^{-1}n^{\varepsilon/4}-2}{\left(\gamma^{-1}n^{\varepsilon/4}-1\right)^{16/\varepsilon+1}-1} \leq \frac{n^{\varepsilon/4}}{\left(\gamma^{-1}n^{\varepsilon/4}-1\right)^{16/\varepsilon+1}-1}.$$

By choosing an appropriate small positive constant ξ such that $\xi < (1-\varepsilon/4)\varepsilon/16$, this is at most

$$\frac{n^{\varepsilon/4}}{\left(\gamma^{-1}n^{\varepsilon/4}-1\right)^{16/\varepsilon+1}-1} = \frac{n^{\varepsilon/4}}{\left(n^{\varepsilon/4-\xi}(\gamma^{-1}n^{\xi})-1\right)^{16/\varepsilon+1}-1}$$
$$\leq \frac{n^{\varepsilon/4}}{\left(n^{\varepsilon/4-\xi}\right)^{16/\varepsilon+1}-1} \leq \frac{n^{\varepsilon/4}}{\left(n^{\varepsilon/4-\xi}\right)^{16/\varepsilon}} = \frac{n^{\varepsilon/4}}{n^{4-16\xi/\varepsilon}} < n^{-3}.$$

Since there are $o(n \ln n/p)$ evaluations and therefore also $o(n \ln n/p)$ phases, w.h.p. no phase will result in a drop of λ below $6eF^{16/\varepsilon} \ln n$.

It remains to show (iii). All points in I' have fitness at most $n - n^{1-\varepsilon/2} + d \leq n - k^*$. Therefore, the algorithm can not terminate before crossing the interval. If the algorithm enters and leaves I' several times, we consider the last such time. By [9, Lemma 3.3] the probability of an offspring having a Hamming distance $\Omega(\ln n)$ to its parent is $n^{-\Omega(\ln \ln n)}$. Thus when the algorithm enters I', w.h.p. it enters in a point x_t such that $\mathrm{ZM}(x_t) \geq n^{1-\varepsilon/4} - O(\ln n)$. From such a starting point, even the $(1+1)$-EA on OneMax requires $\Omega(n \ln n)$ evaluations to cross I' [9, Theorem 3.6]. By the domination result [9, Theorem 3.5], the SA-$(1,\lambda)$-EA with resets requires at least as many evaluations as the $(1+1)$-EA, hence $\Omega(n \ln n)$ evaluations to cross the interval. □

Having established that the algorithm enters the set of distorted points $\mathcal{D}$ with at least logarithmic λ in I', we proceed to show that the algorithm will not leave the set $\mathcal{D}$ before crossing the following interval I as well. We additionally show that the algorithm becomes elitist, i.e., the fitness is not reduced throughout.

Lemma 8. *Consider the SA-$(1,\lambda)$-EA with resets on* DisOM *as in Theorem 1, in particular, let δ be such that $p \geq n^{\delta}/\lambda_{max}$. Let $\varepsilon > 0$ be a small constant such that $k^* + d \leq n^{1-\varepsilon}$ and $\varepsilon \leq \delta/4$. Let the current state (x_0, λ_0) satisfy $ZM(x_0) \leq n^{1-\varepsilon/4}$, x_0 is distorted and $\lambda_0 \geq 6eF^{16/\varepsilon} \ln n$. With high probability the algorithm neither leaves the set of distorted points $\mathcal{D}$ nor decreases the fitness in a single step until either the total number of evaluations is in $\Omega(n \ln n/p)$, or the distance to $\vec{1}$ is reduced to at most $n^{1-\varepsilon}$.*

Proof. If each generation has a clone among the offspring w.h.p. the algorithm neither leaves $\mathcal{D}$ nor reduces the fitness by Lemma 4. To show that each iteration has a clone, assume λ does not drop below $6e \ln n$. By Lemma 3, a single generation does not have a clone among its offspring with probability at most

$$\left(1 - \frac{n-1}{en}\right)^{6e \ln n} \leq n^{-6(n-1)/n} \leq n^{-3}.$$

Therefore, the probability of not having a clone among the first $o(n \ln n/p)$ generations is $1 - o(1)$ by a union bound.

It remains to show that λ does not drop below $6e \ln n$. We proceed almost identically as in the proof of (ii) in Lemma 7. We start with the probability of a reset. If p_{imp} is the probability of a single offspring increasing the fitness

w.r.t. the parent, the probability of a single generation causing a reset is at most $(1-p_{\text{imp}})^{\lambda_{max}}$. For an offspring to increase the fitness it must both increase the OneMax-value and be distorted. It might be tempting to bound the probability of an offspring being distorted by p, but we need to mind that the noise is frozen. Consequently, we do not get fresh randomness in each step. We circumvent the problem in a similar fashion to Lemma 5.

There are $\binom{n^{1-\varepsilon}}{3} = \Omega(n^{3-3\varepsilon})$ points in the three-neighborhood of the parent, which additionally increase the OneMax-value by three. For sufficiently small ε any such offspring is distorted with probability $\Omega(p)$ by an analogous reasoning to Lemma 5. The probability of an offspring falling into this category is

$$\binom{n^{1-\varepsilon}}{3}\frac{1}{n^3}\left(1-\frac{1}{n}\right)^{n-3} \geq \left(\frac{n^{1-\varepsilon}}{3n}\right)^3\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{n^{-3\varepsilon}}{27e} = \Omega(n^{-3\varepsilon}).$$

Together this implies that $p_{\text{imp}} = \Omega(pn^{-3\varepsilon})$. Leveraging the relationship $p \geq n^{\delta}/\lambda_{max} \geq n^{4\varepsilon}/\lambda_{max}$, the probability of a reset in a single generation is at most

$$(1-p_{\text{imp}})^{\lambda_{max}} \leq \exp\left(-\Omega\left(\frac{n^{4\varepsilon}}{\lambda_{max}}n^{-3\varepsilon}\right)\lambda_{max}\right) \leq \exp\left(-\Omega\left(n^{\varepsilon}\right)\right).$$

W.h.p. the algorithm will not encounter a reset during the $o(n\ln n/p)$ generations for sufficiently small ε.

It remains to show that λ does not drop below $6e\ln n$ as a consequence of a "natural" reduction. Using an identical reasoning to the proof of (i) in Lemma 7, with $\alpha := 6e\ln n$, $\beta := 16/\varepsilon$ and $q := (6eF^{16/\varepsilon}\ln n)n^{-\varepsilon/4} \leq (6eF^{32/\varepsilon}\ln n)n^{-\varepsilon/4}$, it is clear that w.h.p. λ will not drop below $6e\ln n$. □

As a final step, it remains to show that the algorithm takes $\Omega(n\ln n/p)$ evaluations to cross the second interval I.

Lemma 9. *Let $\varepsilon > 0$ be a small constant such that $k^* + d \leq n^{1-\varepsilon}$. Consider the SA-$(1,\lambda)$-EA with resets on* DisOM *and the interval $I := [n^{1-\varepsilon}, n^{1-\varepsilon/2}]$ of Hamming distances to $\vec{1}$. If the algorithm neither leaves the set $\mathcal{D}$ of distorted points nor decreases the fitness, then with high probability it takes $\Omega(n\ln n/p)$ evaluations to cross the interval I.*

Proof. Due to the similar nature of the problem on hand, the following part is closely related to the analysis of the lower bound of T^{plus} in [9, Theorem 1.1]. We adopt the notion of a $(1-p)$-rejection run. Such a run differs from a regular one in that each sampled search point is discarded with a probability $1-p$. We show that for any state (x_t, λ_t) of the algorithm on DisOM, the probability of increasing the fitness by $r > 0$ without leaving $\mathcal{D}$ is at most the probability of increasing it by r in a $(1-p)$-rejection run on OneMax.

Assume an offspring y satisfies $\text{OM}(y) = \text{OM}(x_t) + r$. If y has been sampled before, it is not distorted as we assumed the algorithm does not decrease the fitness, and therefore, it would have moved to the point the first time it was sampled. Thus, y is not considered for selection with probability 1. On the other hand, if y has not yet been sampled, it is distorted with probability p. Thus, it

is not considered for selection with probability $1-p$. Summarizing, each point increasing the fitness by r is not considered for selection with probability at least $1-p$. In other words, it is rejected with at least this probability.

It remains to show that a $(1-p)$-rejection run of the algorithm on OneMax takes $\Omega(n \ln n/p)$ evaluations. By [9, Theorem 3.6] w.h.p. the $(1+1)$-EA takes $\Omega(n \ln (n^{1-\varepsilon/2}/n^{1-\varepsilon})) = \Omega(n \ln n)$ evaluations to cross I. Hence a $(1-p)$-rejection run of the same algorithm takes $\Omega(n \ln n/p)$ evaluations. By the domination result [9, Theorem 3.5] this implies that the same is true for a $(1-p)$-rejection run of the SA-$(1,\lambda)$-EA with resets on OneMax. □

We now bring everything together and prove Theorem 1.

Proof (of Theorem 1). Let $\varepsilon > 0$ be a small constant such that $k^* + d \leq n^{1-\varepsilon}$ and $\varepsilon \leq \delta/4$. Consider the intervals $I' := [n^{1-\varepsilon/2}, n^{1-\varepsilon/4}]$ and $I := [n^{1-\varepsilon}, n^{1-\varepsilon/2}]$ of Hamming distances to $\vec{1}$. With high probability the initial search point of the algorithm has a distance of at least $n/3$ to $\vec{1}$. It thus has to cross both intervals to reach the target fitness of $n - k^*$. By [9, Lemma 3.3], the probability of an offspring having Hamming distance at least $c \ln n$ to its parent is $n^{-\Omega(\ln \ln n)}$. Therefore, the algorithm will not increase the OneMax-value by $\Omega(\ln n)$ in the first $o(n \ln n/p)$ evaluations. As a result, the algorithm will jump over neither of the two intervals (both have size $\omega(\ln n)$).

Let a *trial* be defined as a sequence of generations, which starts as soon as the algorithm samples a search point in I' and ends if either

(i) the algorithm accepts an offspring y with $\mathrm{ZM}(y) > n^{1-\varepsilon/4}$, or
(ii) the algorithm accepts an offspring y with $\mathrm{ZM}(y) < n^{1-\varepsilon/2}$, or
(iii) the algorithm reaches a state (x_t, λ_t) such that $\mathrm{ZM}(x_t) \in I'$, x_t is distorted and $\lambda_t \geq 6eF^{16/\varepsilon}$.

If a trial ends due to condition (i), with an analogous argumentation to before, a new trial will start again w.h.p.. Under the assumption of a trial not ending due to condition (i), by Lemma 7, w.h.p. the trial will end due to condition (iii), or the number of evaluations is in $\Omega(n \ln n/p)$, in which case the theorem would follow immediately.

From such a state w.h.p. the algorithm will neither decrease the fitness nor leave the set of distorted points $\mathcal{D}$ before either the total number of evaluations is in $\Omega(n \ln n/p)$ or the algorithm accepts an offspring y with $\mathrm{ZM}(y) < n^{1-\varepsilon}$, by Lemma 8. This allows us to apply Lemma 9. To cross the interval I the algorithm will require $\Omega(n \ln n/p)$ evaluations, which concludes the proof. □

Corollary 2 puts our findings in context with [9]. Items (2) and (3) are direct results of Theorem 1 and [9, Theorem 1.1] respectively. Even though (1) is similar to the corresponding result in [9, Theorem 1.1], it is not a direct consequence. We provide a brief proof sketch.

Proof (of Corollary 2 (1)). We define the same intervals I and I' as in the proof of Theorem 1. Since even the $(1+1)$-EA takes $\Omega(n \ln n)$ evaluations to cross I', so does the $(1,\lambda)$-EA. With Lemma 5, among these offspring at least one is distorted, which is also accepted w.h.p. by Lemma 4. Since no offspring among the

first $o(n \ln n/p)$ increases the Hamming distance by $\Omega(\ln n)$, the algorithm will not leave the set of distorted points $\mathcal{D}$. To cross the second interval I, the algorithm requires $\Omega(n \ln n/p)$ evaluations with analogous reasoning to Lemma 9.

5 Experiments

We corroborate our theoretical results empirically with two sets of experiments[4]. We theoretically showed a lower bound on the runtime T of the SA-$(1, \lambda)$-EA with resets of $\Omega(n \ln n/p)$, so we plot the normalized runtime $T/((n \ln n)/p)$ in Fig. 1. We show the mean and the standard deviation for three problem sizes with varying distortion probabilities p over 50 runs each. The remaining parameters stay unchanged. Indeed, for larger p, the curve is almost horizontal, which suggests that the lower bound is tight and $T = \Theta(n \ln n/p)$. For smaller p, the runtime becomes irregular, showing that a lower bound on p as in Theorem 1 is indeed necessary.

In Fig. 2, we compare the runtime behavior of the SA-$(1, \lambda)$-EA with resets, the $(1, \lambda)$-EA, and the $(1 + \lambda)$-EA. This confirms that the $(1, \lambda)$-EA with static λ is much faster than the two other algorithms.

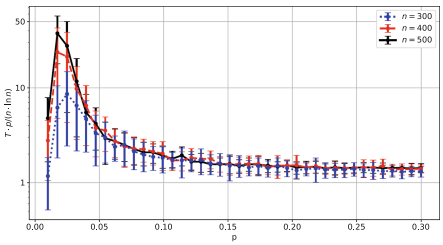


Fig. 1. Normalized number of evaluations required by the SA-$(1, \lambda)$-EA with resets to optimize DISOM for different distortion probabilities p. We set $d = \ln n$, $k^* = n^{0.4}$, $F = 1.5$, $s = 1$, $\lambda_{max} = n \ln n$ and average over 50 runs each. The cutoff of 10^7 evaluations was never reached. Note that the y-axis shows the number of evaluations T multiplied by $p/(n \ln n)$ and is scaled logarithmically.

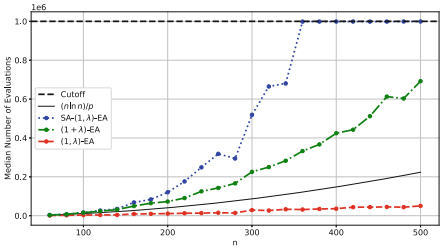


Fig. 2. We take the median over 50 runs for the $(1, \lambda)$-EA, the $(1 + \lambda)$-EA and the SA-$(1, \lambda)$-EA with resets. We set $d = \ln n$, $k^* = n^{0.4}$, $\lambda_{com,plus} = \lfloor 1.5 \ln n \rceil$ for the $(1, \lambda)$-EA and the $(1 + \lambda)$-EA, $p = (e/(e-1))^{-\lambda_{com,plus}}$, $F = 1.5$, $s = 1$, $\lambda_{max} = n \ln n$. We make a cutoff at 10^6 evaluations. We additionally plot the curve $n \ln n/p$ for reference.

[4] The code for the experiments can be found at https://github.com/kosturm/EAs-on-Distorted-OneMax.

6 Conclusion

We have investigated the SA-$(1, \lambda)$-EA with resets on DISOM. While this algorithm has been very successful on hill-cimbing tasks and on the multimodal function CLIFF, we have shown that this does not extend to the type of local optima that DISOM represents. We believe that it is worthwhile to explore the algorithm on other theoretical benchmarks to understand better in which situations it is slowed down. Candidates include HURDLE, the recently introduced benchmark BBFUNNEL [1], and the multimodal landscapes introduced by Jansen and Zarges [8]. Moreover, it is important to explore other self-adaptation mechanisms that may provide alternatives to the resetting mechanism studied in this paper and which may be able to keep the advantages of comma selection for local optima of the type as in DISOM.

Acknowledgements. This research was strongly influenced by the discussions at the Dagstuhl seminars 22081 "Theory of Randomized Optimization Heuristics" and 23332 "Synergizing Theory and Practice of Automated Algorithm Design for Optimization".

Disclosure of Interests. The authors have no competing interests.

References

1. Dang, D.C., Eremeev, A., Lehre, P.K.: Non-elitist evolutionary algorithms excel in fitness landscapes with sparse deceptive regions and dense valleys. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1133–1141 (2021)
2. 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 (2020)
3. Doerr, B., Doerr, C., Lengler, J.: Self-adjusting mutation rates with provably optimal success rules. Algorithmica **83**(10), 3108–3147 (2021)
4. Doerr, B., Witt, C., Yang, J.: Runtime analysis for self-adaptive mutation rates. Algorithmica **83**(4), 1012–1053 (2021)
5. Feller, W.: An Introduction to Probability Theory and Its Applications, vol. 2, vol. 81. Wiley, New York (1991)
6. Hevia Fajardo, M.A., Sudholt, D.: Self-adjusting population sizes for non-elitist evolutionary algorithms: why success rates matter. Algorithmica (2023)
7. Hevia Fajardo, M.A., Sudholt, D.: Self-adjusting offspring population sizes outperform fixed parameters on the cliff function. Artif. Intell. **328**, 104061 (2024)
8. Jansen, T., Zarges, C.: Example landscapes to support analysis of multimodal optimisation. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 792–802. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45823-6_74
9. Jorritsma, J., Lengler, J., Sudholt, D.: Comma selection outperforms plus selection on onemax with randomly planted optima. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1602–1610. ACM, New York (2023)

10. Kaufmann, M., Larcher, M., Lengler, J., Sieberling, O.: Hardest monotone functions for evolutionary algorithms. In: Stützle, T., Wagner, M. (eds.) EvoCOP 2024. LNCS, vol. 14632, pp. 146–161. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-57712-3_10
11. Kaufmann, M., Larcher, M., Lengler, J., Zou, X.: Self-adjusting population sizes for the (1, λ)-EA on monotone functions. Theoret. Comput. Sci. **979**, 114181 (2023)
12. Kaufmann, M., Larcher, M., Lengler, J., Zou, X.: Onemax is not the easiest function for fitness improvements. Evol. Comput. 1–30 (2024)
13. 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)
14. Lengler, J., Schiller, L., Sieberling, O.: Plus strategies are exponentially slower for planted optima of random height. In: Proceedings of the Genetic and Evolutionary Computation Conference (2024)
15. Rechenberg, I.: Evolutionsstrategien. In: Simulationsmethoden in der Medizin und Biologie: Workshop, Hannover, 29 September–1 October 1977, pp. 83–114. Springer, Heidelberg (1978)

Runtime Analysis of Evolutionary Diversity Optimization on a Tri-Objective Version of the (LeadingOnes, TrailingZeros) Problem

Denis Antipov[1(✉)], Aneta Neumann[1], Frank Neumann[1], and Andrew M. Sutton[2]

[1] Optimisation and Logistics, School of Computer and Mathematical Sciences, University of Adelaide, Adelaide, Australia
{denis.antipov,aneta.neumann,frank.neumann}@adelaide.edu.au
[2] University of Minnesota Duluth, Duluth, USA
amsutton@d.umn.edu

Abstract. Diversity optimization is a class of optimization problems in which we aim to find a diverse set of good solutions. One of the frequently used approaches to solve such problems is to use evolutionary algorithms which evolve a desired diverse population. This approach is called evolutionary diversity optimization (EDO).

In this paper, we analyse EDO on a 3-objective function LOTZ_k, which is a modification of the 2-objective benchmark function (LeadingOnes, TrailingZeros). We prove that the GSEMO computes a set of all Pareto-optimal solutions in $O(kn^3)$ expected iterations. We also analyze the runtime of the GSEMO_D (a modification of the GSEMO for diversity optimization) until it finds a population with the best possible diversity for two different diversity measures, the total imbalance and the sorted imbalances vector. For the first measure we show that the GSEMO_D optimizes it asymptotically faster than it finds a Pareto-optimal population, in $O(kn^2 \log(n))$ expected iterations, and for the second measure we show an upper bound of $O(k^2 n^3 \log(n))$ expected iterations. We complement our theoretical analysis with an empirical study, which shows a very similar behavior for both diversity measures that is close to the theoretical predictions.

Keywords: Diversity optimization · Multi-objective optimization · Theory · Runtime analysis

1 Introduction

Computing a diverse set of high quality solutions has recently become an important topic in the area of artificial intelligence and in particular in the field of evolutionary computation [13–15]. Different approaches have been designed for

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 19–35, 2024.
https://doi.org/10.1007/978-3-031-70071-2_2

using classical solvers in order to compute diverse sets of high quality solutions for problems in the areas of planning [16] and satisfiability [25]. Such problems are often met in practice, especially when there are some factors which are hard to formalize, such as politics, ethics or aesthetics. In these cases the algorithm user would prefer to have several different good solutions rather than one single best solution to have an opportunity to choose among them. In practice this problem arises in, e.g., optimization of a building floor plans [9] or in the cutting problem [12].

In contrast to standard single-objective optimization, where the search is performed in a space of potential solutions, diversity optimization works in a space of sets of solutions and is usually also harder from a computation complexity perspective [13]. This makes it natural to use evolutionary algorithms (EAs) for solving such problems, since they are designed to evolve populations of solutions. This approach is called the *evolutionary diversity optimization* (EDO). Evolutionary diversity optimization aims at finding a set of solutions such that (i) all solutions meet a given quality threshold and (ii) the set of solutions has maximum diversity according to a chosen diversity measure.

In multi-objective problems, where the aim is to find a set of Pareto-optimal solutions which are diverse in their fitness, there also might be a need in a diversity of their genotype. E.g., in [17], the authors designed optimal mechanical parts for different contexts, and the aim was to find a good design for each context. To reach this goal, the authors of [17] used quality diversity (QD) approach, which is closely related to EDO. Their work was an inspiration for a further development of multi-objective QD algorithms in [27]. EDO would also be useful in the setting of [17], since it allows to find a diverse Pareto-optimal set, where for each balance between the main objectives the decision maker could choose a good design for the context of their interest.

1.1 Related Work

Feature-based EDO approaches that seek to compute a diverse set of solutions with respect to a given set of features have been carried out for evolving different sets of instances of the traveling salesperson problem [10] as well as evolving diverse sets of images [1]. For these computations a variety of different diversity measures with respect to the given features such as the star discrepancy measure [19] and the use of popular indicators from the area of evolutionary multi-objective optimization [20] have been studied.

Classical combinatorial optimization problems for which EDO algorithms have been designed to compute diverse sets of solutions include the traveling salesperson problem [6,22,23], the traveling thief problem [24], the computation of minimum spanning trees [4] and related communication problems in the area of defense [21].

Establishing the theoretical foundations of evolutionary diversity optimization in the context of runtime analysis is a challenging task as it involves the understanding of population dynamics with respect to the given problem and

used diversity measure. Initial studies have been carried out for classical benchmark problems in the area of evolutionary computation such as OneMax and LeadingOnes [11]. For permutation problems such as the traveling salesperson problem and the quadratic assignment problem runtime bounds have been provided on computing a maximal diverse set of permutation when there is no restriction on the quality of solutions [6]. In the context of the optimization of monotone submodular problems with a given constraint, results on the approximation quality of diversifying greedy approaches that result in diverse population have been provided in [7,18].

In the domain of multi-objective optimization, EDO was studied in [8], where it was shown that on the OneMinMax benchmark problem the GSEMO_D can compute a Pareto-optimal population with the best possible total Hamming distance in $O(n^3 \log(n))$ expected iterations. It was assumed in [2] that this runtime is actually $O(n^2 \log(n))$, and hence it is asymptotically the same as the runtime needed for computing the whole Pareto front with classic multi-objective EAs.

1.2 Our Contribution

We contribute to the rigorous analysis of EDO on multi-objective problems and for the first time consider a 3-objective problem called LOTZ_k (formally defined in the following section). This problem is a modification of the classic bi-objective benchmark problem LOTZ with a significantly larger set of Pareto-optimal solutions, the size of which is regulated with parameter $k \geq 2$.

We perform a runtime analysis of a simple evolutionary multi-objective optimizer GSEMO_D, which optimizes diversity only when breaking ties between individuals with the same fitness which compete for being included into the population. We prove that the GSEMO_D finds a Pareto-optimal population on LOTZ_k in $O(kn^3)$ expected iterations. We also prove for two different diversity measures the upper bounds on the expected time which it takes the GSEMO_D to find the optimal diversity starting from a Pareto-optimal population. For one measure (called the total imbalance) this bound is $O(kn^2 \log(n))$, which is smaller than the upper bound on the runtime until finding a Pareto-optimal population. For the second diversity measure (called the sorted imbalances vector) we prove a weaker upper bound $O(k^2 n^3 \log(n))$. Our proofs are based on a rigorous study of which individuals allow us to improve the diversity, and we believe that similar arguments might be fruitful in the future theoretical studies of EDO.

We also show empirically that it takes the GSEMO_D a relatively short time to find the optimal diversity after covering the whole Pareto front for both diversity measures. This demonstrates the benefits of EDO approach and also suggests that optimizing diversity via tie-breaking rules is an easy-to-implement and a very effective method.

2 Preliminaries

In this paper we consider only pseudo-Boolean optimization, that is, our search space is the set of bit strings of length n. We use the following notation. By $[a..b]$ we denote an integer interval $\{a, a+1, \ldots, b\}$. Bits of a bit string x of length n are denoted by x_i, where $i \in [1..n]$. We assume that x_1 is the leftmost bit and x_n is the rightmost bit. For any non-negative integer i by 1^i (or 0^i) we denote the all-ones (or all-zeros) bit string of length i. If $i = 0$, this is an empty string. When we have a Boolean predicate A, we use Iverson bracket $[A]$ to map this Boolean value into $\{0, 1\}$.

Dominance. Given two points x and y and a k-objective function $f = (f_1, \ldots, f_k)$ defined on these points, we say that x dominates y with respect to f, if for all $i \in [1..k]$ we have $f_i(x) \geq f_i(y)$ and there exists $j \in [1..k]$ for which $f_j(x) > f_j(y)$. We write it as $x \succ y$.

2.1 GSEMO$_D$

The *Simple Evolutionary Multi-objective Optimizer* (SEMO) is an evolutionary algorithm for solving multi-objective problems. At all iterations this algorithm keeps a population of non-dominated solutions. The SEMO is initialized with a point from the search space chosen uniformly at random (u.a.r. for brevity). In each iteration it chooses an individual from its current population u.a.r. and mutates it. If the mutated offspring is not dominated by any individual in the population, it is added to the population, and the individuals which are dominated by this offspring are removed from the population.

The SEMO does not allow two individuals with the same fitness to be in the population. A situation when a new offspring y is identical (in terms of fitness) to some individual x in the population can be handled in different ways. Usually, y replaces x to enhance the exploration of the search space, since y is a new individual, and x is an older one. This tie, however, can also be broken in a way which improves some secondary objective. In this paper we are interested in finding a *diverse* set of non-dominated solutions, so we can remove an individual with the smallest contribution to the diversity. We call the SEMO which optimizes a diversity measure D in such way the SEMO$_D$.

This mechanism is similar to the tie-breaking rule in the $(\mu + 1)$ genetic algorithm (GA) for the single-objective optimization described in [5]. There a tie-breaking rule which optimized the diversity of the population allowed to use crossover in a very effective way to escape local optima. This resulted into a $O(n \log(n) + 2^k)$ runtime on JUMP$_k$ benchmark, which is much smaller than the $\Omega((\frac{n}{k})^k)$ runtime of the most common mutation-based algorithms on that problem. It was also much better than the long-standing $O(n^{k-1})$ bound for many classic crossover-based GAs, which has been improved only recently in [26] to $O(\mu n \log(k) + 4^k/p_c)$ (by showing that the $(\mu+1)$ GA diversifies its population without additional mechanisms). We have a situation different from [5] and [26], since diversity is our primary goal, and the fitness has a role of a constraint (that

Algorithm 1: The Global SEMO$_D$ maximizing a multi-objective function g and optimizing diversity measure D.

```
Choose x ∈ {0,1}^n uniformly at random;
P ← {x};
repeat
    Choose x ∈ P uniformly at random;
    Create y by flipping each bit of x with probability 1/n;
    if ∃w ∈ P : g(w) = g(y) then
        if D((P ∪ {y}) \ {w}) is not worse than D(P) then
            P ← (P ∪ {y}) \ {w};
    else if ∄w ∈ P : w ≻ y then
        P ← (P ∪ {y}) \ {z ∈ P | y ≻ z};
until stop;
```

is, we want the solutions to be Pareto-optimal). However, since the diversity is a measure of the whole population, but not of a single individual, such a tie-breaking rule is a natural way to optimize it after finding a Pareto-optimal population.

In literature, the SEMO which uses standard bit mutation to generate new offspring is usually called the *Global SEMO* (GSEMO). Similarly, we call the SEMO$_D$ with standard bit mutation the GSEMO$_D$. The pseudocode of the GSEMO$_D$ is shown in Algorithm 1.

2.2 Diversity Measures

In this paper we consider diversity measures which are based on the balance between 1-bits and 0-bits in each position in the population. For each $i \in [1..n]$ we denote by $n_1(i)$ the number of individuals in the population, which have a 1-bit in position i. More formally, $n_1(i) = \sum_{x \in P} x_i$, where P is the population of the GSEMO$_D$. Similarly, by $n_0(i)$ we denote the number of 0-bits in position i. We define the *imbalance* $b(i)$ of position i as $|n_1(i) - n_0(i)|$. Intuitively, when we have a large imbalance in position i, the population is too monotonous in that position, hence the optimal diversity implies minimizing the imbalance of each position.

Based on this observation we define two diversity measures. The first one is called the *total imbalance* and it is equal to the sum of the imbalances of all positions. Namely, we have $D(P) = \sum_{i=1}^{n} b(i)$. The smaller this measure is, the better the diversity.

The second measure is the *sorted imbalances vector*, which is defined by vector $D(P) = (b(\sigma(i)))_{i=1}^{n}$, where σ is a permutation of positions $[1..n]$ in the descending order of their imbalances $b(\sigma(i))$. When comparing two populations of the same size, the one with a lexicographically smaller vector $D(P)$ is more diverse than another. We do not determine how to compare diversity of

populations of different sizes, since this comparison never occurs in GSEMO$_D$, and most of the other classic EAs have populations of a constant size.

We note that using the imbalances to estimate the diversity is also implicitly used in the total Hamming distance, which was first shown in [28].

2.3 LOTZ$_k$ Problem

In this paper we consider a classic benchmark bi-objective function (LeadingOnes, TrailingZeros) (LOTZ for brevity), which is defined on a space of bit strings of length n. We call n the problem size. The first objective LeadingOnes (LO for brevity) returns the length of the longest prefix consisting only of 1-bits, more formally,

$$\textsc{LeadingOnes}(x) = \mathrm{LO}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

The second objective TrailingZeros (TZ for brevity) returns the length of the longest suffix which consists only of 0-bits, namely

$$\textsc{TrailingZeros}(x) = \mathrm{TZ}(x) = \sum_{i=1}^{n} \prod_{j=n-i+1}^{n} (1 - x_j).$$

These two objectives contradict each other, and for any bit string x we have $\mathrm{LO}(x) + \mathrm{TZ}(x) \leq n$.

The Pareto front of LOTZ consists of $n+1$ bit strings of form 1^i0^{n-i}, for all $i \in [0..n]$. This means that the Pareto-optimal population has a fixed diversity. To study the aspects of diversity optimization by the GSEMO$_D$, we modify this problem.

We introduce parameter $k \in [0..n]$, and we say that all bit strings x which have $\mathrm{LO}(x)+\mathrm{TZ}(x) \geq n-k$ do not dominate each other. We call such bit strings and also their fitness values *feasible*.[1] We note that there is no bit string x such that $\mathrm{LO}(x) + \mathrm{TZ}(x) = n - 1$, for the following reason. Assume that $\mathrm{LO}(x) = i$. Then x has prefix 1^i0, hence $x_{i+1} = 0$. If $\mathrm{TZ}(x) = n-1-\mathrm{LO}(x) = n-1-i$, then x also has suffix 10^{n-1-i}. Then $x_{i+1} = 1$, which contradicts with the requirement on the prefix, hence we cannot have $\mathrm{LO}(x) + \mathrm{TZ}(x) = n - 1$.

To allow the GSEMO$_D$ handle our requirement on the non-domination between feasible bit strings, we define LOTZ$_k$ as a 3-objective problem

$$\mathrm{LOTZ}_k(x) = (\mathrm{LO}(x), \mathrm{TZ}(x), h(\mathrm{LO}(x) + \mathrm{TZ}(x))),$$

where $h : \mathbb{R} \mapsto \mathbb{R}$ is defined as

$$h(x) \coloneqq \begin{cases} 0, & \text{if } x < n - k, \\ n + 1 - x, & \text{if } x \geq n - k. \end{cases}$$

[1] The illustration of the feasible fitness values can be found in the arXiv version of this paper [3] (Fig. 1). We omit this illustration for reasons of space.

From this definition it follows that for any $x, y \in [0..n]$ we have

$$x > y \Rightarrow \begin{cases} h(x) \geq h(y), & \text{if } y < n-k, \\ h(x) < h(y), & \text{if } y \geq n-k. \end{cases}$$

Consequently, if x dominates y in terms of LOTZ (that is, the first two objectives) and both of them are feasible, then we have $\mathrm{LO}(x) + \mathrm{TZ}(x) > \mathrm{LO}(y) + \mathrm{TZ}(y)$, and hence the third objective is better for y. Hence, any pair of feasible individuals do not dominate each other. Otherwise, if at least one of the two bit strings is not feasible, then the additional objective does not affect the domination relation.

Problem Statement. In this paper we study the behavior of the GSEMO_D on LOTZ_k for a variable parameter k. We aim at estimating the runtime of the GSEMO_D, that is, the number of iterations this algorithm performs until it finds a population which has all feasible solutions in it, and also has the best possible diversity. We note that such a population cannot contain any infeasible solution, since each infeasible solution is dominated by at least some feasible one, thus it cannot be accepted into the final population of the GSEMO_D. As a part of this problem, we also aim at estimating the time until the GSEMO finds a Pareto-optimal population of the 3-objective LOTZ_k function.

3 Optimal Diversity

In this section we show the best possible diversity of a population which covers the whole Pareto front of LOTZ_k (that is, which contains all feasible fitness values). Although we will not derive a simple formula for the optimal vector of imbalances for each position, we will show how the optimal diversity can be computed. We will use the observations from this section in our runtime analysis and also in our experiments to determine the moment when the algorithm finds the optimal diversity.

Before discussing the optimal diversity, we show the following lemma which estimates the population size of GSEMO_D in different stages of the optimization.

Lemma 1. *When the GSEMO_D runs on LOTZ_k with $k \geq 2$, before it finds the first feasible solution, the population size is at most $\max_{x \in P}(\mathrm{LO}(x) + \mathrm{TZ}(x)) + 1 \leq n - k$. Once the GSEMO_D finds a feasible solution, the population size is at most $\mu_{\max} = nk - \frac{(k-2)(k+1)}{2} \leq nk$. The size of any Pareto-optimal population containing all feasible fitness values is also $\mu_{\max}$.*

Proof. Before we find a feasible solution, if we have two different individuals in the population with the same LO value and different TZ values, then one of them dominates another. By the definition of the GSEMO_D, it is impossible. It is also impossible to have two individuals with the same LO and TZ values. Hence, in this stage of optimization we have at most one individual per each LO value in the population. The number of different LO values in the population

is not greater than the maximum LO + TZ value in the population plus one. If we do not have feasible solutions in the population, this value is always at most $n - k$.

We now consider a situation when we have a feasible solution in the population. We cannot have two solutions with the same LO and TZ values in the population. For each LO value $\ell < n$ we can have at most $\min(k, n - \ell)$ solutions with different TZ values in the population: either one infeasible solution x with $\mathrm{TZ}(x) < n - k - \ell$, or a set of feasible solutions with TZ value in $[\max(0, n-k-\ell)..(n-\ell)] \setminus \{n-\ell-1\}$. For $\ell = n$ we can have only one solution, which is 1^n. Summing this up over all LO values, we obtain that the maximum population size $\mu_{\max}$ is

$$\mu_{\max} = \sum_{\ell=0}^{n-1} \min(k, n-\ell) + 1 = k(n-k+1) + \sum_{\ell=n-k+1}^{n-1} (n-\ell) + 1$$
$$= nk - k^2 + k + \frac{k(k-1)}{2} + 1 = nk - \frac{(k-2)(k+1)}{2} \leq nk.$$

This is the same as the total number of different feasible fitness values, thus the maximum size of a Pareto-optimal population. □

We now show how to compute the minimum imbalance in the following Lemma.

Lemma 2. *Consider a population, which covers all feasible fitness values of* LOTZ*. For any position $i \in [1..n]$ the minimum imbalance is*

$$b_{opt}(i) = \max\left(|m_0(i) - m_1(i)| - m(i), \delta\right), \tag{1}$$

where

- *$m_0(i)$ is the number of individuals in the population which are guaranteed to have a 0-bit in position i, and*

$$m_0(i) = \frac{i(i-1)}{2} - [i > k+1] \cdot \frac{(i-k-1)(i-k)}{2} + \min(n-i+1, k),$$

- *$m_1(i)$ is the number of individuals in the population which are guaranteed to have a 1-bit in position i, and*

$$m_1(i) = \frac{(n-i+1)(n-i)}{2} - [i < n-k] \cdot \frac{(n-k-i)(n-k-i+1)}{2} + \min(i, k),$$

- *$m(i)$ is the number of individuals in the population which can have any value in position i, and*

$$m(i) = \min(k-2, i-1) \cdot \min(k-2, n-i) - \frac{a(i)(a(i)+1)}{2}, \text{ where}$$
$$a(i) \coloneqq \max(0, \min(k-3, i-2, n-i-1, n-k)),$$

– δ *is* 1*, if* $\mu_{\max}$ *is odd and* 0*, if* $\mu_{\max}$ *is even.*

We omit the proof for reasons of space, and also since these computations are quite straightforward.[2] The most important outcome of Lemma 2 for our theoretical investigation is that for each position i to have an optimal imbalance $b(i) = b_{\mathrm{opt}}(i)$ in this position in a Pareto-optimal population, we need to have a particular number of 1-bits in position i among the $m(i)$ individuals which can have any value in position i.

With this observation we define $M(i)$ as the set of feasible fitness values which allow any value of bit in position i and we say that the i-th bit of an individual of a Pareto-optimal population with fitness in $M(i)$ is *wrong*, if it has a majority (in the whole population) value of the bits in this position. Otherwise it is *right*. Note that we use this notation only for the bits of individuals with fitness in $M(i)$. The following two lemmas show, how many wrong bits we have in a position depending on its imbalance and how the wrong bits help us to improve the imbalance. We omit the proofs for reasons of space.

Lemma 3. *For any position* $i \in [1..n]$ *the number of wrong bits in this position is at least* $\frac{b(i)-b_{opt}(i)}{2}$.

Lemma 4. *Consider some arbitrary* $i \in [1..n]$ *and a Pareto-optimal population of* LOTZ_k*. If we have* $b(i) > b_{opt}(i)$ *and have* $m'(i) > 0$ *individuals in* $M(i)$ *with a wrong bit in position* i*, then replacing any of such individuals with the same bit string, but with a different bit value in position* i *would reduce* $b(i)$ *by two. The probability that the* GSEMO_D *does it in one iteration is at least* $\frac{m'(i)}{ekn^2}$.

4 Runtime Analysis of Covering All Fitness Values

In this section we analyze the first stage of the algorithm, namely how it gets a Pareto-optimal population which contains all feasible solutions. The main result of this section is the following theorem. We note that although it is formulated for the GSEMO_D, the proof does not use the tie breaking rule in any of the arguments, hence this upper bound also holds for the GSEMO.

Theorem 1. *The expected runtime until the* GSEMO_D *finds all feasible solutions of* LOTZ_k *is* $O(kn^3)$.

Proof. We split the analysis into three phases. The first phase is from the initial population until we find a feasible solution. The second phase lasts until for each LeadingOnes value we have at least one feasible solution in the population. And the third phase lasts until we find all feasible solutions. We note that once a feasible solution gets into a population of the GSEMO_D, its fitness value will always be present in the population, since this solution is never dominated by any other solution.

[2] The full version of this paper with all omitted proofs can be found at arXiv [3].

Phase 1: from initial solution to a feasible solution. Let X_t be $\max_{x \in P_t}(\mathrm{LO}(x)+\mathrm{TZ}(x))$, where P_t is the population in the beginning of iteration t. During this phase we have $X_t < n-k$ and the phase ends as soon as we get $X_t > n-k$. We also note that X_t never decreases with t, since for any two bit strings x, y it is impossible for y to dominate x when $\mathrm{LO}(x)+\mathrm{TZ}(x) > \mathrm{LO}(y)+\mathrm{TZ}(y)$, hence any point x can be removed from the population only by accepting a point with an equal or lager LO + TZ value.

To get $X_{t+1} > X_t$ after iteration t we can choose an individual x with the maximum value of $(\mathrm{LO}(x)+\mathrm{TZ}(x))$ as a parent (or any such individual, if there are more than one) and increase either its LO value by flipping the first 0-bit in it (and not flipping any other bit) or its TZ value by flipping its last 1-bit (and not flipping any other bit). Since we use standard bit mutation, the probability to flip only one of two particular bits is at least $\frac{2}{n}(1-\frac{1}{n})^{n-1} \geq \frac{2}{en}$. By Lemma 1, during this phase the population size in iteration t is at most $X_t + 1$, hence the probability to choose such x as a parent is at least $\frac{1}{X_t+1}$. Therefore, the probability to increase X_t is at least $\frac{2}{en(X_t+1)} > \frac{2}{en^2}$. Hence, for any value of X_t the expected number of iterations until we get a larger X_t is less than $\frac{en^2}{2}$. To get a feasible solution, we need to increase X_t at most $n-k$ times, hence the total expected time until we get a feasible solution is at most $(n-k)\cdot\frac{en^2}{2} < \frac{en^3}{2}$.

Phase 2: finding a feasible solution for each LO value. In this phase we denote by $F_t \subset P_t$ the set of feasible solutions in the population and by L_t we denote the set of different LO values which are present in F_t, that is, $L_t = \{\mathrm{LO}(x)\}_{x \in F_t}$. We estimate the expected time τ until we find a feasible bit string x with $\mathrm{LO}(x)$ not in L_t. We distinguish two cases.

Case 1: $\max(L_t) < n$. In this case we can choose an individual x from F_t with the maximum LO value as a parent and flip its first 0-bit. This would create an individual x', which has $\mathrm{TZ}(x') = \mathrm{TZ}(x)$ and $\mathrm{LO}(x') > \mathrm{LO}(x)$, hence it is feasible and it adds a new LO value to L_t. The probability to chose such an individual is at least $\frac{1}{nk}$, since the population size is at most nk by Lemma 1. The probability to flip only one particular bit is $\frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en}$. Hence, the probability to extend L_t is at least $\frac{1}{ekn^2}$, and $\tau \leq ekn^2$ in this case.

Case 2: $\max(L_t) = n$. In this case we have at least one LO value $\ell \notin L_t$ for which we have $(\ell+1) \in L_t$. Consider an individual $x \in F_t$ with $\mathrm{LO}(x) = \ell+1$. If there are several such individuals let x be the one with the largest $\mathrm{TZ}(x)$ value.

If $\mathrm{TZ}(x) > n-k-\mathrm{LO}(x)$, then $(\mathrm{LO}(x)-1, \mathrm{TZ}(x)) = (\ell, \mathrm{TZ}(x))$ is a feasible fitness value. Hence, if we choose x as a parent and flip a 1-bit in position $\ell+1$, then we get a feasible individual x' with $\mathrm{LO}(x') = \ell$ and $\mathrm{TZ}(x') = \mathrm{TZ}(x)$ (such that $\mathrm{LO}(x') + \mathrm{TZ}(x') \geq n-k$), which adds ℓ to L_t. The probability to do that (similar to the previous case) is $\frac{1}{ekn^2}$, and therefore, $\tau \leq ekn^2$.

If $\mathrm{TZ}(x) = n-k-\mathrm{LO}(x)$, then if we just reduce LO value of x, we get an infeasible individual. However, we can chose x as a parent and flip its last 1-bit. This gives us an individual x' with $\mathrm{LO}(x') = \mathrm{LO}(x)$ and $\mathrm{TZ}(x') \geq \mathrm{TZ}(x)+1$. Hence, this individual is feasible, and adding it into the population gives us an individual which satisfies $\mathrm{TZ}(x') > n-k-\mathrm{LO}(x')$ and $\mathrm{LO}(x') = \ell+1$, hence

after obtaining it we will need at most ekn^2 expected iterations to add ℓ to L_t, as it has been shown in the previous paragraph. The probability to create such x' is at least $\frac{1}{ekn^2}$, and expected runtime until it happens is at most ekn^2. Hence, in the worst scenario of Case 2 we have $\tau \leq 2ekn^2$.

To get all LO values in L_t we need to extend L_t for at most n times, hence the expected time of Phase 2 is at most $2ekn^3$.

Phase 3: covering all TZ values for each LO value. Consider some arbitrary LO value $\ell \in [0..n]$ and let S be a set of individuals x in the population with $\mathrm{LO}(x) = \ell$. We have two ways to extend S (that does not yet contain all possible TZ values), which depend on the maximum TZ value s among individuals in S.

Case 1: $s < n - \ell$, then we can create a bit string with $\mathrm{LO} = i$ and $\mathrm{TZ} > s$ by selecting the individual with $\mathrm{LO} = i$ and $\mathrm{TZ} = s$ (the probability of this is at least $\frac{1}{nk}$) and flip the last 1-bit in it without flipping any other bit (the probability of this is at least $\frac{1}{en}$). Hence, we create an individual with $\mathrm{LO} = \ell$ and with an unseen TZ value with probability at least $\frac{1}{ekn^2}$.

Case 2: $s = n - \ell$ (which is the maximum TZ value for LO value ℓ), then we can create a bit string x with $\mathrm{LO}(x) = \ell$ and a new uncovered TZ value $j < n - \ell - 1$ by selecting the individual with $\mathrm{LO} = \ell$ and $\mathrm{TZ} = s$ and flipping a 0-bit in position $n - \ell$ in it. The probability of this is at least $\frac{1}{ekn^2}$.

We now consider $2ekn^3$ consecutive iterations. Let k' be the number of uncovered TZ values which are missing in S in the beginning of these iterations. Note that $k' \leq k \leq n$. The event when we create a bit string with an uncovered TZ value for the fixed LO value ℓ happens in each of the $2ekn^3$ iterations with probability at least $\frac{1}{ekn^2}$ until we cover all k' uncovered TZ values. Hence, the number of such events during these iterations dominates a random variable $X \sim \min(k', Y)$, where $Y \sim \mathrm{Bin}(2ekn^3, \frac{1}{ekn^2})$. During this series of iterations we have less than k' such events with probability at most $\Pr[X \leq k'] = \Pr[Y \leq k'] \leq \Pr[Y \leq n]$. Since $E[Y] = 2n$, by the Chernoff bound the latter probability is at most

$$\Pr\left[Y \leq \left(1 - \frac{1}{2}\right) E[Y]\right] \leq \exp\left(-\frac{\frac{1}{4}E[Y]}{3}\right) = e^{-n/6}.$$

By the union bound over all $(n-1)$ different LO values[3] the probability that after $2ekn^3$ iteration we have at least one such value with a non-covered TZ value is at most $(n-1)e^{-n/12} = o(1)$. Hence, the expected number of such phases of length $2ekn^3$ which we need to cover all feasible solutions is $(1 + o(1))$. Therefore, the expected time of Phase 4 is $2e(1 + o(1))kn^3$.

Summing up the expected times of each phase, we obtain that the expected runtime is at most $\frac{en^3}{2} + 2ekn^3 + 2e(1 + o(1))kn^3 = O(kn^3)$. □

[3] We exclude values n and $n - 1$, for which we have only one feasible pair and hence for those LO values all TZ values are covered after the second phase.

5 Runtime Analysis of Diversity Optimization

In this section we analyze how much time it takes the GSEMO$_D$ to find a population with the optimal diversity after it has already found a population of all feasible solutions. We start with the following theorem for the total imbalance measure.

Theorem 2. *Consider a run of the GSEMO$_D$ on* LOTZ$_k$*, which minimizes the diversity measure* $D(P) = \sum_{i=1}^{n} b(i)$ *and which starts with population* P_0 *that covers all feasible fitness values. Then the expected runtime until the GSEMO$_D$ finds a population with the best possible diversity is* $O(kn^2 \log(n))$.

Proof. Let P_t be a population of the GSEMO$_D$ in the beginning of iteration t and let $\phi_t(i)$ be the difference between the imbalance of position i and its optimal imbalance in P_t, that is, $\phi_t(i) \coloneqq b(i) - b_{\text{opt}}(i)$. Let also $\Phi_t \coloneqq \sum_{i=1}^{n} \phi_t(n)$, which we call the *potential* of the population in iteration t. Note that the potential decreases strongly monotone with the diversity $D(P_t)$ and hence no population increasing the potential is accepted. When $\Phi_t = 0$, it implies that all $\phi_t(i) = 0$, and therefore, population P_t has an optimal diversity. Therefore, to estimate the runtime of the GSEMO$_D$, we need to estimate the time until Φ_t becomes zero.

Note that for each i the imbalance of position i is defined by the i-th bits of individuals with fitness in $M(i)$, hence the maximum difference of $b(i)$ and $b_{\text{opt}}(i)$ is $m(i)$ which by Lemma 2 is at most k^2. Therefore, each $\phi_t(i)$ is at most k^2 and thus, Φ_t is at most nk^2.

By Lemma 4, the probability to reduce $\phi_t(i)$ by two in one iteration is at least $\frac{m'(i)}{ekn^2}$, where we recall that $m'(i)$ is the number of wrong bits in position i. By Lemma 3 we have $m'(i) \geq \frac{\phi_t(i)}{2}$, hence the probability to reduce $\phi_t(i)$ by two is at least $\frac{\phi_t(i)}{2ekn^2}$. The probability to reduce Φ_t by two is at least the probability that we reduce at least one $\phi_t(i)$ by two. Since the events considered in Lemma 4 are disjoint for different positions, we have

$$\Pr[\Phi_t - \Phi_{t+1} = 2] \geq \sum_{i=1}^{n} \frac{\phi_t(i)}{2ekn^2} = \frac{\Phi_t}{2ekn^2}.$$

For each value Φ_t we reduce this value at most once. Conditional on $\Phi_t = s$, the probability to reduce it is at least $\frac{s}{2ekn^2}$, and the expected time until we reduce Φ_t is at most $\frac{2ekn^2}{s}$. Since Φ_t can only take integer values from $[1..nk^2]$ before we find the optimum, the total expected runtime until we find the optimal population is at most the sum of the runtimes to reduce each of the possible values of Φ_t, that is,

$$E[T] \leq \sum_{s=1}^{nk^2} \frac{2ekn^2}{s} \leq 2ekn^2(\ln(nk^2) + 1) = O(kn^2 \log(n)).$$

□

Note that Theorem 2 gives an upper bound which is asymptotically smaller than the upper bound on the runtime until the GSEMO_D finds all feasible solutions, which by Theorem 1 is $O(kn^3)$, hence the expected runtime until the GSEMO_D finds a population of all feasible solutions with an optimal diversity starting from a random bit string is also $O(kn^3)$.

The second diversity measure which we consider in this paper is the vector of position imbalances sorted in descending order and which is to be minimized lexicographically. For this diversity measure we show the following theorem.

Theorem 3. *Consider a run of the GSEMO_D on LOTZ_k, which starts with population P_0, which covers all feasible fitness values, and which minimizes diversity measure $D(P) = (b(\sigma(i)))_{i=1}^{n}$, where σ is a permutation of positions $[1..n]$ in descending order of their imbalances. Then the expected runtime until the GSEMO_D finds a population with the best possible diversity is $O(k^2n^3\log(n))$.*

We only sketch the proof for reason of space.[4] We define the potential Φ_t of the population at iteration t as the largest imbalance of a position, which has a non-optimal imbalance. The maximum value of Φ_t is nk and we can show that we decrease it in at most $ekn^2(\ln(n)+1)$ expected iterations, hence the total expected runtime is $O(k^2n^3\log(n))$.

We note that when we optimize the diversity measured by the sorted imbalances vector, the total imbalance might increase. This happens, when we decrease the imbalance of some position i, but also increase imbalances of positions which at the moment have smaller imbalances than position i. This situation resembles optimization of the BinVal benchmark function with the $(1+1)$ EA, for which OneMax value can increase, but it does not slow down the optimization [29]. This analogy makes us optimistic that it takes much less time to optimize this sorted imbalances vector than our upper bound in Theorem 3, and the results of the experiments shown in the next section support this optimism. However, the main problem with proving it is that imbalances can be changed in large chunks, when we replace one individual with a one-bit mutation of another, but not of itself.

6 Experiments

In this section we show the results of our empirical study. We run the GSEMO_D on LOTZ_k on different problem sizes and with different values of k. We used $n \in \{2^3, 2^4, 2^5, 2^6, 2^7\}$ and for each n, except $n = 2^7$, we used $k \in \{2, 4, \lfloor\sqrt{n}\rfloor, \frac{n}{2}, n\}$. For the largest $n = 2^7$ we used only $k \in \{2, 4, \lfloor\sqrt{n}\rfloor\}$. We used both diversity measures (the total imbalance and the sorted balances vector) and made 128 runs (this number gives us enough confidence in that the mean runtime does not deviate too much from its expectation) of the GSEMO_D for each parameter setting and each diversity measure. In our experiments we do not initialize random seed, that is, it was initialized with the timestamp at the moment of starting the experiments.

[4] The omitted proof is included in the full version of this paper at arXiv [3].

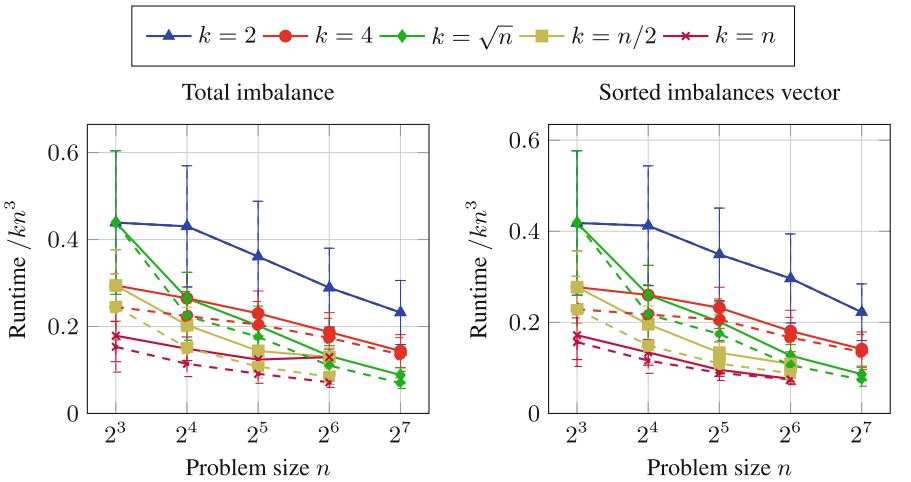


Fig. 1. The normalized runtimes of the GSEMO$_D$ when optimizing the total imbalance diversity (on the left) and the sorted imbalances vector (on the right). The dashed lines show the time until a Pareto-optimal population is found and the solid lines show the time until an optimal diversity is obtained. All runtimes are normalized by kn^3, which is asymptotically the same as the upper bound shown in Theorem 1.

All plots in this section show the mean runtimes over 128 runs and they have errorbars which indicate the standard deviation. All of them are normalized by the upper bound from Theorem 1 on the time of the first phase of the optimization, that is, by kn^3.

6.1 Sum of Imbalances

The results of the runs when the GSEMO$_D$ minimized the sum of total imbalances are shown in the left plot in Fig. 1. In this figure we see that all the normalized runtimes (both the runtimes until we obtain a Pareto-optimal population, indicated by the dashed lines, and runtimes until the optimal diversity, indicated by the solid lines) are decreasing, which suggests that the asymptotical upper bound might be even smaller than $O(kn^3)$, but not by a large factor.

We observe that the runtime required by the algorithm to get an optimal diversity after computing a Pareto-optimal population is small compared to the runtime required to find a Pareto-optimal population for the first time. This matches the ratio between the upper bounds shown in Theorems 1 and 3, however, we note that without a proof of the lower bound for the first stage (until we cover the Pareto front) we cannot state that the runtime of the second stage (that is, after finding a Pareto-optimal population) is a small fraction of the total runtime.

Also, note that for $k = 2$ both lines coincide. This is because for any fitness pair $(f_{\text{LO}}, f_{\text{TZ}})$ such that $f_{\text{LO}} + f_{\text{TZ}} = n - 2$ there exists only one bit string with this fitness, hence there exists a unique Pareto-optimal population, which therefore has an optimal diversity.

6.2 Sorted Imbalances Vector

The results of the runs when the GSEMO_D minimized the vector of imbalances, sorted in descending order, are shown in the right plot in Fig. 1. In this figure we see that all the normalized runtimes are decreasing, which suggests that the asymptotical upper bound might be even smaller than $O(kn^3)$, but not by a large factor. This also indicates that the upper bound on the diversity optimization time given in Theorem 3 is not tight and in practice the runtime required for diversity optimization is not larger than the runtime needed for finding a Pareto-optimal population. For large values of k the runtime of diversity optimization is even smaller than it is for the total imbalance.

7 Conclusion

In this paper, we have shown that optimizing diversity with EAs in a multi-objective setting might be easy compared to the time needed for the computation of the Pareto front. We showed that a simple tie-breaking rule implemented into the GSEMO can effectively find the best possible diversity of a Pareto-optimal population. This lines up with the result of [2] for the OneMinMax problem, even though the main source of diversity improvements is different in their setting (in [2] and also in [8] the proofs relied on two-bits flips which improve the diversity). Our analysis is also the first one performed on a multi-objective problem with more than two objectives, which demonstrates that evolutionary algorithms can be effective within such a multi-dimensional domain, where the main factor which slows down the optimization is usually a big size of the Pareto front.

The results, however, raise a question, which diversity measures are the fastest to optimize. As we see from our results, although the two considered measures share the set of populations which have the optimal diversity, they are optimized with the GSEMO_D in different ways. In practice the difference might be even larger, since the diversity is also optimized in the earlier stages of optimization, that is, before we find a Pareto-optimal population, and this difference might be of particular interest for using EDO in practice.

From theoretical perspective, it would be valuable to find the lower bounds on the runtime of diversity optimization. In general, finding lower bounds for multi-objective problems is already a challenging task. In EDO it is even more complicated, since the diversity might be optimized also before we cover the Pareto front. Thus, a very precise analysis of the multi-objective optimization and diversity optimization in parallel is required. We are optimistic that studying this problem might inspire new analysis methods for a broader class of multi-objective problems.

Acknowledgements. This work has been supported by the Australian Research Council through grants DP190103894 and FT200100536.

References

1. Alexander, B., Kortman, J., Neumann, A.: Evolution of artistic image variants through feature based diversity optimisation. In: Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 171–178. ACM (2017)
2. Antipov, D., Neumann, A., Neumann, F.: Rigorous runtime analysis of diversity optimization with GSEMO on OneMinMax. In: Foundations of Genetic Algorithms, FOGA 2023, pp. 3–14. ACM (2023)
3. Antipov, D., Neumann, A., Neumann, F.: Runtime analysis of evolutionary diversity optimization on the multi-objective (LeadingOnes, TrailingZeros) problem. CoRR abs/2404.11496 (2024). https://arxiv.org/abs/2404.11496
4. Bossek, J., Neumann, F.: Evolutionary diversity optimization and the minimum spanning tree problem. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 198–206. ACM (2021)
5. Dang, D., et al.: Escaping local optima using crossover with emergent diversity. IEEE Trans. Evol. Comput. **22**(3), 484–497 (2018)
6. Do, A.V., Guo, M., Neumann, A., Neumann, F.: Analysis of evolutionary diversity optimization for permutation problems. ACM Trans. Evol. Learn. Optim. **2**(3), 11:1–11:27 (2022)
7. Do, A.V., Guo, M., Neumann, A., Neumann, F.: Diverse approximations for monotone submodular maximization problems with a matroid constraint. In: International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5558–5566. ijcai.org (2023)
8. Doerr, B., Gao, W., Neumann, F.: Runtime analysis of evolutionary diversity maximization for OneMinMax. In: Genetic and Evolutionary Computation Conference, GECCO 2016, pp. 557–564. ACM (2016)
9. Galle, P.: Branch & sample: a simple strategy for constraint satisfaction. BIT **29**(3), 395–408 (1989)
10. Gao, W., Nallaperuma, S., Neumann, F.: Feature-based diversity optimization for problem instance classification. Evol. Comput. **29**(1), 107–128 (2021)
11. Gao, W., Neumann, F.: Runtime analysis for maximizing population diversity in single-objective optimization. In: Genetic and Evolutionary Computation Conference, GECCO 2014, pp. 777–784. ACM (2014)
12. Haessler, R.W., Sweeney, P.E.: Cutting stock problems and solution procedures. Eur. J. Oper. Res. **54**, 141–150 (1991)
13. Hanaka, T., Kiyomi, M., Kobayashi, Y., Kobayashi, Y., Kurita, K., Otachi, Y.: A framework to design approximation algorithms for finding diverse solutions in combinatorial problems. In: AAAI Conference on Artificial Intelligence, AAAI 2023, pp. 3968–3976. AAAI Press (2023)
14. Hanaka, T., Kobayashi, Y., Kurita, K., Lee, S.W., Otachi, Y.: Computing diverse shortest paths efficiently: A theoretical and experimental study. In: AAAI Conference on Artificial Intelligence, AAAI 2022, pp. 3758–3766. AAAI Press (2022)
15. Ingmar, L., de la Banda, M.G., Stuckey, P.J., Tack, G.: Modelling diversity of solutions. In: AAAI Conference on Artificial Intelligence, AAAI 2020, pp. 1528–1535. AAAI Press (2020)
16. Katz, M., Sohrabi, S.: Reshaping diverse planning. In: AAAI Conference on Artificial Intelligence, AAAI 2020, pp. 9892–9899. AAAI Press (2020)

17. Makatura, L., Guo, M., Schulz, A., Solomon, J., Matusik, W.: Pareto gamuts: exploring optimal designs across varying contexts. ACM Trans. Graph. **40**(4), 171:1–171:17 (2021)
18. Neumann, A., Bossek, J., Neumann, F.: Diversifying greedy sampling and evolutionary diversity optimisation for constrained monotone submodular functions. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 261–269. ACM (2021)
19. Neumann, A., Gao, W., Doerr, C., Neumann, F., Wagner, M.: Discrepancy-based evolutionary diversity optimization. In: Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 991–998. ACM (2018)
20. Neumann, A., Gao, W., Wagner, M., Neumann, F.: Evolutionary diversity optimization using multi-objective indicators. In: Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 837–845. ACM (2019)
21. Neumann, A., et al.: Diversity optimization for the detection and concealment of spatially defined communication networks. In: Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 1436–1444. ACM (2023)
22. Nikfarjam, A., Bossek, J., Neumann, A., Neumann, F.: Computing diverse sets of high quality TSP tours by EAX-based evolutionary diversity optimisation. In: Foundations of Genetic Algorithms, FOGA 2021, pp. 9:1–9:11. ACM (2021)
23. Nikfarjam, A., Bossek, J., Neumann, A., Neumann, F.: Entropy-based evolutionary diversity optimisation for the traveling salesperson problem. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 600–608. ACM (2021)
24. Nikfarjam, A., Neumann, A., Neumann, F.: Evolutionary diversity optimisation for the traveling thief problem. In: Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 749–756. ACM (2022)
25. Nikfarjam, A., Rothenberger, R., Neumann, F., Friedrich, T.: Evolutionary diversity optimisation in constructing satisfying assignments. In: Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 938–945. ACM (2023)
26. Opris, A., Lengler, J., Sudholt, D.: A tight $O(4^k/p_c)$ runtime bound for a $(\mu+1)$ GA on Jump_k for realistic crossover probabilities. In: Genetic and Evolutionary Computation Conference, GECCO 2024. ACM (2024, to appear)
27. Pierrot, T., Richard, G., Beguir, K., Cully, A.: Multi-objective quality diversity optimization. In: Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 139–147. ACM (2022)
28. Wineberg, M., Oppacher, F.: The underlying similarity of diversity measures used in evolutionary computation. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1493–1504. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45110-2_21
29. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Combin. Probab. Comput. **22**(2), 294–318 (2013)

Sliding Window 3-Objective Pareto Optimization for Problems with Chance Constraints

Frank Neumann[1(✉)] and Carsten Witt[2]

[1] Optimisation and Logistics, School of Computer and Mathematical Sciences, The University of Adelaide, Adelaide, Australia
frank.neumann@adelaide.edu.au

[2] DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark

Abstract. Constrained single-objective problems have been frequently tackled by evolutionary multi-objective algorithms where the constraint is relaxed into an additional objective. Recently, it has been shown that Pareto optimization approaches using bi-objective models can be significantly sped up using sliding windows [16]. In this paper, we extend the sliding window approach to 3-objective formulations for tackling chance constrained problems. On the theoretical side, we show that our new sliding window approach improves previous runtime bounds obtained in [15] while maintaining the same approximation guarantees. Our experimental investigations for the chance constrained dominating set problem show that our new sliding window approach allows one to solve much larger instances in a much more efficient way than the 3-objective approach presented in [15].

Keywords: chance constraints · evolutionary algorithms · multi-objective optimization

1 Introduction

Multi-objective formulations have been widely used to solve single-objective optimization problems. The initial study carried out by Knowles et al. [8] for the H-IFF and the traveling salesperson problem shows that such formulations can significantly reduce the number of local optima in the search space and uses the term *multi-objectivization* for such approaches. Using multi-objective formulations to solve constrained single-objective optimization problems by evolutionary multi-objective optimization using the constraint as an additional objective has shown to be highly beneficial for a wide range of problems [4,9,12]. Using the constraint as an additional objective for such problems allows simple evolutionary multi-objective algorithms such as GSEMO mimic a greedy behaviour and as a consequence allows us to achieve theoretically best possible performance guarantees for a wide range of constrained submodular optimization problems [17–19]. Such approaches have been widely studied recently under the term *Pareto optimization* in the artificial intelligence and machine learning literature [22].

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 36–52, 2024.
https://doi.org/10.1007/978-3-031-70071-2_3

In the context of problems with stochastic constraints, it has recently been shown that 3-objective formulations where the given constraint is relaxed into a third objective lead to better performance than 2-objective formulations that optimize the expected value and variance of the given stochastic components under the given constraint [14,15]. The experimental investigations for the chance constrained dominating set problem carried out in [15] show that the 3-objective approach is beneficial and outperforms the bi-objective one introduced in [14] for medium size instances of the problem. However, it has difficulties in computing even a feasible solution for larger graphs. In order to deal with large problem instances we design a new 3-objective Pareto optimization approach based on the sliding window technique for Pareto optimization recently introduced in [16]. Using sliding window selection has been shown to scale up the applicability of GSEMO type algorithms for the optimization of monotone functions under general cost constraints. Here, at a given time step only solutions with a fixed constraint value are chosen in the parent selection step. This allows the algorithm to proceed with achieving progress in the same way as the analysis for obtaining theoretical performance guarantees. It does so by dividing the given function evaluation budget $t_{\max}$ equally among the different constraint values starting with selecting individuals with the smallest constraint value at the beginning and increasing it over time until it reaches that given constraint bound at the end of the run. A positive effect is that the maximum population size can be eliminated as a crucial factor in the given runtime bounds. Furthermore, experimental studies carried out in [16] show that the approach provides major benefits when solving problems on graphs with up to 21,000 vertices.

Making the sliding window technique work requires one to deal with a potentially large number of trade-off solutions even for a small number of constraint values. We design highly effective 3-objective Pareto optimization approaches based on the sliding window technique. Our theoretical investigations using runtime analysis for the chance constrained problem under a uniform constraint show that our approach may lead to a significant speed-up in obtaining all required Pareto optimal solutions. In order to make the approach even more suitable in practice, we introduce additional techniques that do not hinder the theoretical performance guarantees, but provide additional benefits in applications. One important technique is to control the sliding window through an additional parameter a. Choosing $a \in]0,1[$ allows the algorithm to move the sliding window faster at the beginning of the optimization process and slow down when approaching the constraint bound. This allows us to maintain the benefit of the Pareto optimization approach including its theoretical performance guarantees while focusing more on the improvement of already high quality solutions at the end of the optimization run. The second technique that we incorporate is especially important for problems like the dominating set problem where a constraint that is not fulfilled for most of the optimization process needs to be fulfilled at the end. In order to deal with this, we introduce a parameter $t_{\text{frac}} \in [0,1]$ which determines the fraction of time our sliding window technique is used. If after $t_{\text{frac}} \cdot t_{\max}$ steps a feasible solution has not been found yet, then in each step a

solution from the population that is closest to feasible is selected in the parent selection step to achieve feasibility within the last $(1 - t_{\text{frac}}) \cdot t_{\max}$ steps.

This paper is structured as follows: in Sect. 2, we present the multi-objective algorithms considered in this paper, in particular the 3-objective approach using sliding window selection. Section 3 proves the improved runtime bounds for this approach. Section 4 presents the empirical comparison of the different algorithms on a large set of instances of the minimum dominating set problem. We finish with some conclusions.

2 Algorithms

In this section, we define the algorithmic framework incorporating sliding window selection into two-objective optimization problems under constraints. It combines the 3-objective problem formulation from [15], where the underlying problem is 2-objective and a constraint is converted to a helper objective, with the two-objective formulation from [16], where the problem is single-objective and the constraint is converted to a helper objective and additionally undergoes the so-called sliding window selection. More precisely, sliding window is based on the observation that several problems under uniform constraints can be solved by iterating over increasing constraint values and optimizing the actual objectives for each fixed constraint value.

We consider an optimization problem on bit strings $x \in \{0,1\}^n$ involving two objective functions $\mu(x), v(x)\colon \{0,1\}^n \to \mathbb{R}_0^+$ and an integer-valued constraint function $c(x)\colon \{0,1\}^n \to \mathbb{N}$ with bound B, i. e., the only solutions satisfying $c(x) \leq B$ are feasible. Our new approach called SW-GSEMO3D is shown in Algorithm 1 (which will later be extended to Algorithm 4 explained below). The sliding window selection in Algorithm 3 will be used as a module in SW-GSEMO3D and choose from its current population P, which is the first parameter of the algorithm. The idea is to select only from a subpopulation of constraint values in a specific interval determined by the maximum constrained value B, the current generation t, the maximum number of iterations of the algorithm $t_{\max}$, and further parameters. In the simplest case (where the remaining parameters are set to $a = 1$, $c_{\max} = -1$ and $t_{\text{frac}} = 1$), the time interval $[1, t_{\max}]$ is uniformly divided into B time intervals in which only the subpopulation having constraint values in the interval $[\lfloor (t/t_{\max})B \rfloor - \text{std}, \lfloor (t/t_{\max})B \rfloor + \text{std}]$, where $\text{std} \geq 0$ is a deviation that allows selection from a larger interval, which is another heuristic component investigated in Sect. 4. Moreover, as not all problems may benefit from selecting according to the specific interval order, the calls to Algorithm 3 resort to selection from the interval $[B - \text{std}, B]$ for the last $(1 - t_{\text{frac}})t_{\max}$ steps. Finally, since making progress may become increasingly difficult for increasing constraint values, the selection provides the parameter a which will allow time intervals of varying length for the different constraint values to choose from. If $a < 1$, the time allocated to choosing from a specific constraint value (interval) increases with the constraint value. Lines 8–10 of the algorithm make sure that solutions with too low constraint value (less than ℓ), but not equaling the

Algorithm 1: Sliding Window GSEMO3D (SW-GSEMO3D)

1 Choose initial solution $x \in \{0,1\}^n$;
2 Set $t_0 \leftarrow -1$;
3 $P \leftarrow \{x\}$;
4 Compute $f(x) = (\mu(x), v(x), c(x))$;
5 $t \leftarrow 1$;
6 $\mu_{\min} \leftarrow \mu(x)$;
7 **if** $\mu_{\min} = 0$ **then**
8 $\quad t_0 \leftarrow t$;
9 **repeat**
10 $\quad$ **if** $(t_0 = -1) \wedge (t \leq t_{\max})$ **then**
11 $\quad\quad x \leftarrow \arg\min\{\mu(z) \mid z \in P\}$ (breaking ties arbitrarily)
12 $\quad$ **else**
13 $\quad\quad x = \text{sliding-selection}(P, t - t_0, t_{\max} - t_0, 0, B, 1, 1, -1)$;
14 $\quad$ Create y from x by mutation;
15 $\quad$ Compute $f(y) = (\mu(y), v(y), c(y))$;
16 $\quad$ **if** $\mu(y) < \mu_{\min}$ **then**
17 $\quad\quad \mu_{\min} \leftarrow \mu(y)$;
18 $\quad$ **if** $(t_0 = -1) \wedge (\mu_{\min} = 0)$ **then**
19 $\quad\quad t_0 \leftarrow t$;
20 $\quad$ **if** $\nexists\, w \in P : w \succ y$ **then**
21 $\quad\quad P \leftarrow (P \setminus \{z \in P \mid y \succeq z\}) \cup \{y\}$;
22 $\quad t \leftarrow t + 1$;
23 **until** $t \geq t_{\max}$;

parameter $c_{\max}$ are permanently removed from the population. Line 11 confines the population to select from to the desired window of constraint values $[\ell, h]$. In case that no solution of those values exists, a uniform choice from the population remaining after removal of individuals of too low constraint values is made. Hence, even if there are no individuals with constraint values in the interval $[\ell, h]$, then lines 8–10 favor increasing constraint values.

In our theoretical studies, we focus on SW-GSEMO3D which uses sliding window selection with the default choices std $= 0$, $t_{\text{frac}} = 1$, $a = 1$ and $c_{\max} = -1$. It starts out with a solution chosen uniformly at random and is run on the bi-objective optimization problem $(\mu(x), v(x))$, both of which are minimized and will correspond to expected value and variance of a chanced constrained optimization problem further explained in Sect. 3. In particular, we assume $\mu(x) \geq 0$ for all $x \in \{0,1\}^n$ and $\mu(0^n) = 0$, and accordingly for $v(x)$. Following the usual definitions in multi-objective optimization, we say that a solution x dominates a solution y ($x \succeq y$) iff $c(x) \geq c(y) \wedge \mu(x) \leq \mu(y) \wedge v(x) \leq v(y)$. Furthermore, we say a solution x strongly dominates y ($x \succ y$) iff $x \succeq y$ and $(\mu(x), v(x), c(x)) \neq (\mu(x), v(x), c(x))$.

Algorithm 2: Standard-bit-mutation-plus(x)

```
y ← x;
repeat
    Create y from x by flipping each bit x_i of x with probability 1/n.
until x ≠ y;
Return y;
```

Algorithm 3: sliding-selection$(P, t, t_{\max}, std, B, t_{\text{frac}}, a, c_{\max})$

```
1  ĉ ← (t^a/(t_frac · t_max)^a) · B;
2  if t ≤ (t_frac · t_max) then
3      ℓ = ⌊ĉ⌋ − std;
4      h = ⌈ĉ⌉ + std;
5  else
6      ℓ = B − std;
7      h = B;
8  for x ∈ P do
9      if (c(x) < ℓ) ∧ (c(x) ≠ c_max) ∧ (c_max ≠ −1) ∧ (|P| > 1) then
10         P ← P \ {x};
11 P̂ = {x ∈ P | ℓ ≤ c(x) ≤ h};
12 if P̂ = ∅ then
13     P̂ ← P;
14 Choose x ∈ P̂ uniformly at random;
15 Return x;
```

The SW-GSEMO3D starts out with a solution $x \in \{0,1\}^n$ chosen by the user, e. g., as the all-zeros string or uniformly at random. It works in two phases. As long as the minimum μ-value of the population called $\mu_{\min}$ is positive, it chooses a solution of this smallest μ-value, applies mutation, usually standard bit mutation avoiding duplicates (Algorithm 2), and accepts the offspring into the population if it is not strictly dominated by another member of the population. All individuals that are weakly dominated by the offspring are then removed from the population. In any case, the current population always consists of mutually non-dominating solutions only. From the point of time t_0 on where a solution x satisfying $\mu(x) = 0$ is found for the first time, the algorithm chooses from the population using sliding window selection (see Algorithm 3) for the remaining $t_{\max} - t_0$ steps. In Algorithm 3, the choice $c_{\max} = -1$ implies that lines 8–10 do nothing.

Algorithm 4 called Fast SW-GSEMO3D extends Algorithm 1 with heuristic elements as follows. First of all, sliding window selection is called with user-specified choices of std, t_{frac} and a as defined above. Moreover, it keeps track of the maximum constraint value $c_{\max}$ found in the population (lines 24–25), uses that in the sliding window selection and introduces a margin parameter ϵ

Algorithm 4: Fast Sliding-Window GSEMO3D (Fast SW-GSEMO3D) (Parameters: $t_{\max}, t_{\text{frac}}, std, a, \epsilon$)

Choose initial solution $x \in \{0,1\}^n$;
$t_0 \leftarrow -1$, $t \leftarrow 1$, $\mu_{\min} \leftarrow \mu(x)$, $c_{\max} \leftarrow -1$;
$P \leftarrow \{x\}$;
Compute $f(x) = (\mu(x), v(x), c(x))$;
if $(c(x) > c_{\max}) \wedge (c(x) \leq B)$ **then**
    $c_{\max} \leftarrow c(x)$;
if $\mu_{\min} = 0$ **then**
    $t_0 \leftarrow t$;
repeat
    $t \leftarrow t + 1$;
    if $(t_0 = -1) \wedge (t \leq t_{\text{frac}} \cdot t_{\max})$ **then**
        $x \leftarrow \arg\min\{\mu(z) \mid z \in P\}$
    else
        if $(t > t_{\text{frac}} \cdot t_{\max}) \wedge (c_{\max} < B - \epsilon)$ **then**
            $x \leftarrow \arg\max\{c(z) \mid z \in P\}$
        else
            $x = \text{sliding-selection}(P, t - t_0, t_{\max} - t_0, std, B, t_{\text{frac}}, a, c_{\max})$;
    Create y from x by mutation;
    Compute $f(y) = (\mu(y), v(y), c(y))$;
    if $\mu(y) < \mu_{\min}$ **then**
        $\mu_{\min} \leftarrow \mu(y)$;
    if $(t_0 = -1) \wedge (\mu_{\min} = 0)$ **then**
        $t_0 \leftarrow t$;
    if $(c(y) > c_{\max}) \wedge (c(y) \leq B)$ **then**
        $c_{\max} \leftarrow c(y)$;
    if $\nexists\, w \in P : w \succ y$ **then**
        $P \leftarrow (P \setminus \{z \in P \mid y \succeq z\}) \cup \{y\}$;
until $t \geq t_{\max}$;

such that sliding window selection is only run until $c_{\max} = B - \epsilon$. Afterwards, i. e., when the algorithm is close to the constraint boundary, making further progress in the constraint value may be too difficult for sliding window selection. Therefore, for the last $(1 - t_{\text{frac}})t_{\max}$ steps, the algorithm chooses an individual of maximum constraint value if $c_{\max} < B - \epsilon$ holds. These heuristic elements underlying the parameters std, t_{frac}, a and ϵ and the use of $c_{\max}$ in the sliding window selection will show some empirical benefit in Sect. 4.

For the sake of completeness, we also define GSEMO, a classical multi-objective optimization algorithm [5,10] that has inspired the developments of Algorithms 1 and 4 and serves as a baseline in our experiments. It maintains a population of non-dominating solutions of unbounded size, starting from a

solution chosen uniformly at random, and creates one offspring per generation by choosing an individual uniformly at random, applying standard bit mutation avoiding duplicates, and accepting the offspring if it is not dominated by any member of the population. Depending on the number of objectives used in the experiments in Sect. 4, we will consider specific instances of the algorithm called GSEMO2D and GSEMO3D as in [15].

3 Runtime Analysis of 3D Sliding Window Algorithm

In our theoretical study, we consider the chance constrained problem investigated in [14] using rigorous runtime analysis, which is a major direction in the area of theory of evolutionary computation [3,7,13]. Given a set of n items $I = \{e_1, \ldots, e_n\}$ with stochastic weights w_i, $1 \leq i \leq n$, we want to solve

$$\min W \quad \text{subject to} \quad (Pr(w(x) \leq W) \geq \alpha) \wedge (|x|_1 \geq k), \tag{1}$$

where $w(x) = \sum_{i=1}^{n} w_i x_i$, $x \in \{0,1\}^n$, and $\alpha \in [1/2, 1[$. The weights w_i are independent random variables following a normal distribution $N(\mu_i, \sigma_i^2)$, $1 \leq i \leq n$, where $\mu_i \geq 1$ and $\sigma_i \geq 1$, $1 \leq i \leq n$. We denote by $\mu(x) = \sum_{i=1}^{n} \mu_i x_i$ the expected weight and by $v(x) = \sum_{i=1}^{n} \sigma_i^2 x_i$ the variance of the weight of solution x.

Algorithm 5: Global simple evolutionary multi-objective optimizer (GSEMO)

```
1 Choose initial solution x ∈ {0,1}^n;
2 P ← {x};
3 repeat
4 |   Choose x ∈ P uniformly at random;
5 |   Create y from x by mutation;
6 |   if ∄ w ∈ P : w ≻ y then
7 |   |   P ← (P \ {z ∈ P | y ⪰ z}) ∪ {y};
8 until stop;
```

As stated in [14], the problem given in Equation (1) is equivalent to minimizing

$$\hat{w}(x) = \mu(x) + K_\alpha \sqrt{v(x)}, \tag{2}$$

under the constraint that $|x|_1 \geq k$ holds. Here, K_α denotes the α-fractional point of the standard Normal distribution. Our algorithm can also be used to maximize a given deterministic objective $c(x)$ under a given chance constraint, i. e.,

$$\max c(x) \quad \text{subject to} \quad Pr(w(x) \leq B) \geq \alpha. \tag{3}$$

with $w(x) = \sum_{i=1}^{n} w_i x_i$ where each w_i is chosen independently of the other according to a Normal distribution $N(\mu_i, \sigma_i^2)$, and B and $\alpha \in [1/2, 1[$ are a given

weight bound and reliability probability. Such a problem formulation includes for example the maximum coverage problem in graphs with so-called chance constraints [1,11], where $c(x)$ denotes the nodes covered by a given solution x and the costs are stochastic. Furthermore, the chance constrained knapsack problem as investigated in [20,21] fits into this problem formulation.

In [15], the 3-objective formulation of chance-constrained optimization problems under a uniform constraint given in (1) was proposed. Let $f_{3D}(x) = (\mu(x), v(x), c(x))$, where $\mu(x)$ and $v(x)$ are the expected weight and variance, respectively, as above, and $c(x)$ is the constraint value of a given solution that should be maximized. In our theoretical study, we focus on the case $c(x) = |x|_1$, which turns the constraint $|x|_1 \geq k$ into the additional objective of maximizing the number of bits in the given bitstring. This 3-objective formulation was introduced as an alternative model to the bi-objective model from [14], which considers penalty terms for violating the constraint $|x|_1 \geq k$.

Based on the ideas for the 3-objective GSEMO from [15], we formulate the following result for SW-GSEMO3D (Algorithm 1). The analysis is additionally inspired by [16], where a bi-objective sliding windows approach for submodular optimization was analyzed. Our theorem assumes an initialization with the all-zeros string. If uniform initialization is used, SW-GSEMO3D nevertheless reaches the all-zeros string efficiently, as shown in a subsequent lemma (Lemma 1).

The following theorem is based on the maximum population size $P^{(i)}_{\max}$ observed in any of the sliding window intervals. Note that when running the algorithm, the runtime for a given sliding window can be adapted to $t^{(i)}_{\max} = P^{(i)}_{\max} n \ln n$ during the run based on the observed value of $P^{(i)}_{\max}$ in order to guarantee the stated approximation result. Note that the previous result from [15] showed an upper bound of $O(n^2 P_{\max})$, where $P_{\max}$ is the overall maximum population size observed in the run of the algorithm. If the largest $P^{(i)}_{\max}$ is significantly smaller than $P_{\max}$, the following theorem gives a significantly stronger upper bound.

Theorem 1. *Let $P^{(i)}_{\max}$ denote the largest number of individuals with constraint value i present in the population at all points in time where SW-GSEMO3D can select such individuals, let $t^{(i)}_{\max} = P^{(i)}_{\max} n \ln n$ and let $t_{\max} = 4en \max_{i=0}^{n-1} t^{(i)}_{\max}$. Then SW-GSEMO3D, initialized with 0^n, computes a population which includes an optimal solution for the problem given in Equation (1) (for any choice of $k \in \{0,\ldots,n\}$ and $\alpha \in [1/2,1[$) and Equation (3) (with $c(x) = |x|_1$ for any choice of $B \in \{0,\ldots,n\}$ and $\alpha \in [1/2,1[$) until time $t_{\max} = O(\max_{i=0}^{n-1}\{P^{(i)}_{\max}\} \cdot n^2 \log n)$ with probability $1 - o(1)$.*

Proof. Let $X^k = \{x \in \{0,1\}^n \mid |x|_1 = k\}$ be the set of all solutions having exactly k elements. We show the following more technical statement S: *the population P at time $t_{\max}$ will, with the probability bound claimed in the theorem, contain for each $\alpha \in [1/2,1[$ and $k \in \{0,\ldots,n\}$ a solution*

$$x^k_\alpha = \arg\min_{x \in X^k} \left\{ \mu(x) + K_\alpha \sqrt{v(x)} \right\}, \tag{4}$$

i. e., $P \supseteq \{x_\alpha^k \mid 0 \leq k \leq n, \alpha \in [1/2, 1[\}$. By Theorem 4.3 in [15], such a population contains the optimal solutions for any choice of $\alpha \in [1/2, 1[$. Note that not the whole set of Pareto optimal solutions is necessarily required.

To show statement S, we re-use the following definitions from [14]. Let $\lambda_{i,j} = \frac{\sigma_j^2 - \sigma_i^2}{(\mu_i - \mu_j) + (\sigma_j^2 - \sigma_i^2)}$ for the pair of elements e_i and e_j of the given input where $\sigma_i^2 < \sigma_j^2$ and $\mu_i > \mu_j$ holds, $1 \leq i < j \leq n$. The set $\Lambda = \{\lambda_0, \lambda_1, \ldots, \lambda_\ell, \lambda_{\ell+1}\}$ where $\lambda_1, \ldots, \lambda_\ell$ are the values $\lambda_{i,j}$ in increasing order and $\lambda_0 = 0$ and $\lambda_{\ell+1} = 1$. Moreover, we define the function $f_\lambda(x) = \lambda\mu(x) + (1-\lambda)v(x)$ and also use it applied to elements e_i, i. e. $f_\lambda(e_i) = \lambda\mu_i + (1-\lambda)\sigma_i^2$.

As noted in [14], for a given λ and a given number k of elements to include, the function f_λ can be optimized by a greedy approach which iteratively selects a set of k smallest elements according to $f_\lambda(e_i)$. For any $\lambda \in [0, 1]$, an optimal solution for f_λ with k elements is Pareto optimal as there is no other solution with at least k elements that improves the expected cost or variance without impairing the other. Hence, once obtained such a solution x, the resulting objective vector $f_{3D}(x)$ will remain in the population for the rest of the run of SW-GSEMO3D. Furthermore, the set of optimal solutions for different λ values only change at the λ values of the set Λ as these λ values constitute the weighting where the order of items according to f_λ can switch [6,14].

We consider a $\lambda_i \in \Lambda$ with $0 \leq i \leq \ell$. Similarly to [6], we define $\lambda_i^* = (\lambda_i + \lambda_{i+1})/2$. The order of items according to the weighting of expected value and variance can only change at values $\lambda_i \in \Lambda$ and the resulting objective vectors are not necessarily unique for values $\lambda_i \in \Lambda$. Choosing the λ_i^*-values in the defined way gives optimal solutions for all $\lambda \in [\lambda_i, \lambda_{i+1}]$ which means that we consider all orders of the items that can lead to optimal solutions when inserting the items greedily according to any fixed weighting of expected weights and variances. In the following, we analyze the time until an optimal solution with exactly k elements has been produced for $f_{\lambda_i^*}(x) = \lambda_i^*\mu(x) + (1-\lambda_i^*)v(x)$ for any $k \in \{0, \ldots, n\}$ and any $i \in \{0, \ldots, \ell\}$. Note that these λ_i^* values allow to obtain all optimal solutions for the set of functions f_λ, $\lambda \in [0, 1]$.

For a given i, let the items be ordered such that $f_{\lambda_i^*}(e_1) \leq \cdots \leq f_{\lambda_i^*}(e_k) \leq \cdots \leq f_{\lambda_i^*}(e_n)$ holds. An optimal solution for k elements and λ_i^* consists of k elements with the smallest $f_{\lambda_i^*}(e_i)$-value. If there are more than one element with the value $f_{\lambda_i^*}(e_k)$ then reordering these elements does not change the objective vector or $f_{\lambda_i^*}$-value.

Note that for $k = 0$ the search point 0^n is optimal for any $\lambda \in [0, 1]$. Picking an optimal solution with k elements for $f_{\lambda_i^*}$ and inserting an element with value $f_{\lambda_i^*}(e_{k+1})$ leads to an optimal solution for $f_{\lambda_i^*}$ with $k+1$ elements. We call such a step, picking the solution that is optimal for $f_{\lambda_i^*}$ with k elements and inserting an element with value $f_{\lambda_i^*}(e_{k+1})$, a success. Assuming such a solution is picked, the probability of inserting the element is at least $(1/n)(1-1/n)^{n-1} \geq 1/(en)$ since it suffices that SW-GSEMO3D flips a specific bit and does not flip the rest.

We now consider a sequence of events leading to the successes for all values of $k \in \{0, \ldots, n-1\}$ and all $i \in \{0, \ldots, \ell - 1\}$. We abbreviate $P_{\max}^* := \max_{j=0}^{n-1} P_{\max}^{(j)}$. By the assumption from the theorem, 0^n, an optimal solution

for $k = 0$, is in the population at time 0. Assume that optimal solutions with k elements all for $f_{\lambda_i^*}$, where $i \in \{0, \ldots, \ell\}$, are in the population P at time $\tau_k := kt_{\max}/n = 4eP^*_{\max}kn \ln n$.

Then, by definition of the set $\hat{P}$ of GSEMO3D, for any fixed i, such a solution is available for selection up to time

$$\tau_{k+1} - 1 = (k+1)t_{\max}/n - 1 = 4eP^*_{\max}(k+1)n \ln n - 1$$

since $\lfloor((k+1)t_{\max}/n - 1)/t_{\max}) \cdot n \rfloor = k$. The size of the subset population that the algorithm selects from during this period has been denoted by $P^{(k)}_{\max}$. Therefore, the probability of a success at any fixed value k and i is at least $1/(P^{(k)}_{\max}en)$ from time τ_k until time $\tau_{k+1}-1$, i. e., for a period of $4eP^*_{\max}n \ln n \geq 4eP^{(k)}_{\max}n \ln n$ steps, and the probability of this not happening is at most

$$\left(1 - \frac{1}{P^{(k)}_{\max}en}\right)^{4eP^{(k)}_{\max}n \ln n} \leq \frac{1}{n^4}.$$

The number ℓ of different values of λ_i^* is at most the number of pairs of elements and therefore at most n^2. By a union bound over this number of values and all k, the probability to have not obtained all optimal solutions for all $f_{\lambda_i^*}$, where $i \in \{0, \ldots, \ell\}$, and all values of $k \in \{0, \ldots, B\}$ by time $t_{\max}$ is $O(1/n)$. This shows the result for Equation (1). The result for (3) follows from the proof of [15, Theorem 4.3]. □

Finally, as mentioned above, we consider a uniform choice of the initial individual of SW-GSEMO3D and show that the time to reach the all-zeros string is bounded by $O(n \log n)$ if the largest possible expected value $\mu_{\max} := \sum_{i=1}^{n} \mu_i$ of an individual is polynomially bounded. Hence, this constitutes a lower-order term in terms of the optimization time bound proved in Theorem 1 above. Even if $\mu_{\max}$ is exponential like 2^{n^c} for a constant c, the bound of the lemma is still polynomial.

Lemma 1. *Consider SW-GSEMO3D initialized with a random bit string. Then the expected time until its population includes the all-zeros string for the first time is bounded from above by* $O(n(\log \mu_{\max} + 1))$.

Proof. We apply multiplicative drift analysis [2] with respect to the stochastic process $X_t := \min\{\mu(x) \mid x \in P_t\}$, i. e., the minimum expected value of the individuals of the population at time t. By definition, before the all-zeros string is included in the population, SW-GSEMO3D chooses only individuals of minimum μ-value for mutation. The current μ-value of an individual is the sum of the expected values belonging to the bit positions that are set to 1. Standard-bit mutation flips each of these positions to 0 without flipping any other bit with probability at least $(1/n)(1 - 1/n)^{n-1} \geq 1/(en)$. Such steps decrease the μ-value of the solution, which is therefore not dominated by any other solution in the population and will be included afterwards. Hence, we obtain the drift $E(X_t - X_{t+1} \mid X_t) \geq X_t/(en)$. Using the parameter $\delta = 1/(en)$, $X_0 \leq \mu_{\max}$

and the fact that the smallest non-zero expected value of a bit is at least 1, we apply the multiplicative drift theorem [2] and obtain an expected time of at most $\frac{\ln(\mu_{\max})+1}{\delta} = O(n(\log \mu_{\max} + 1))$ to reach an individual with all zeros. □

4 Experiments

We carry out experimental investigations for the new sliding window approach on the chance constrained dominating set problem and show where the new approach performs significantly better than the ones introduced in [14,15].

We recall the chance-constrained dominating set problem. The input is given as a graph $G = (V, E)$ with $n = |V|$ nodes and weights on the nodes. The goal is to compute a set of nodes $D \subseteq V$ of minimal weight such that each node of the graph is dominated by D, i.e. either contained in D or adjacent to a node in D. In our setting the weight w_i of each node v_i is chosen independently of the others according to a normal distribution $N(\mu_i, \sigma_i^2)$. The constraint function $c(x)$ counts the number of nodes dominated by the given search point x. As each node needs to be dominated in a feasible solution, x is feasible iff $c(x) = n$ holds and therefore work with the bound $B = n$ in the algorithms. We start with an initial solution $x \in \{0,1\}^n$ chosen uniformly at random. We also investigate starting with $x = 0^n$ for Fast SW-GSEMO3D (denoted as Fast SW-GSEMO3D$_0$) in the case of large graphs as this could be beneficial for such settings. We try to give some explanation by considering how the maximal population size differs when starting with a solution chosen uniformly at random or with 0^n.

As done in [14,15], we consider the graphs cfat200-1, cfat200-2, ca-netscience, ca-GrQc, and Erdos992 consisting of 200, 200, 379, 4158, and 6100 nodes respectively, together with the following categories for choosing the weights. In the *uniform* setting each weight $\mu(u)$ is an integer chosen independently and uniformly at random in $\{n, \ldots, 2n\}$. The variance $v(u)$ is an integer chosen independently and uniformly at random in $\{n^2, \ldots, 2n^2\}$. In the *degree-based* setting, we have $\mu(u) = (n+\deg(u))^5/n^4$ where $\deg(u)$ is the degree of node u in the given graph. The variance $v(u)$ is an integer chosen independently and uniformly at random in $\{n^2, \ldots, 2n^2\}$. For these graphs, we use 10M (million) fitness evaluations for each run. We also use the graphs ca-CSphd, ca-HepPh, ca-AstroPh, ca-CondMat, which consist of 1882, 11204, 17903, 21363 nodes. They have already been investigated in [16] in the context of the maximum coverage problem. We examine the same uniform random and degree-based setting as described before. We consider 1M fitness evaluations for these graphs in order to investigate the performance on large graphs with a smaller fitness evaluation budget.

For our new sliding window algorithms we use $t_{\text{frac}} = 0.9$, $std = 10$, $a = 0.5$, $\epsilon = 0$ based on some preliminary experimental investigations. Furthermore, we consider $10M$ fitness evaluations for all algorithms and results presented in Table 1 and 1M fitness evaluations for the instances in Table 2. For each setting, each considered algorithm is run on the same set of 30 randomly generated instances. We use the Mann-Whitney test to compute the p-value for algorithm

Table 1. Results for stochastic minimum weight dominating set with different confidence levels of α where $\alpha = 1 - \beta$. Results after 10M fitness evaluations. p_1: Test GSEMO2D vs GSEMO3D, p_2: Test GSEMO2D vs Fast SW-GSEMO3D, p_3: Test GSEMO3D vs Fast SW-GSEMO3D, p_4: Fast GSEMO3D vs Fast SW-GSEMO3D$_0$. Penalty function value for run not obtaining a feasible solution is 10^{10} (applied to GSEMO3D for graphs ca-GrQc and Erdos992)

Graph/ weight type	β	GSEMO2D [15]		GSEMO3D [15]			Fast SW-GSEMO3D				Fast SW-GSEMO3D$_0$		
		Mean	Std	Mean	Std	p_1-value	Mean	Std	p_2-value	p_3-value	Mean	Std	p_4-value
cfat200-1/ uniform	0.2	3615	91	3599	79	0.544	**3594**	75	0.420	0.807	3598	74	0.767
	0.1	3989	96	3972	80	0.544	**3967**	77	0.391	0.734	3971	79	0.745
	0.01	4866	109	4845	86	0.535	**4842**	87	0.383	0.836	4846	90	0.784
	1.0E-4	6015	126	5991	98	0.455	**5989**	101	0.412	0.894	**5989**	100	0.888
	1.0E-6	6855	138	6832	108	0.605	6827	108	0.420	0.712	**6825**	107	0.848
	1.0E-8	7546	147	7525	118	0.641	7517	115	0.455	0.723	**7514**	114	0.935
	1.0E-10	8145	154	8125	125	0.751	8115	120	0.525	0.717	**8112**	120	0.853
	1.0E-12	8680	159	8660	130	0.859	8651	126	0.615	0.790	**8646**	124	0.802
	1.0E-14	9169	164	9148	133	0.842	9139	130	0.600	0.728	**9133**	128	0.830
cfat200-2/ uniform	0.2	1791	49	1767	32	0.049	1766	33	0.031	0.712	**1765**	33	0.971
	0.1	2040	54	2016	37	0.074	2014	36	0.044	0.819	**2013**	37	0.824
	0.01	2621	72	2593	51	0.162	2588	49	0.066	0.610	**2587**	50	0.912
	1.0E-4	3381	97	3336	65	0.070	**3334**	66	0.061	0.836	**3334**	67	0.947
	1.0E-6	3937	113	3880	71	0.044	**3879**	75	0.036	0.853	**3879**	76	0.994
	1.0E-8	4394	124	4329	77	0.032	**4328**	79	0.027	0.853	**4328**	79	1.000
	1.0E-10	4793	132	4720	82	0.028	4719	82	0.021	0.877	**4718**	82	1.000
	1.0E-12	5149	139	5071	85	0.024	5069	85	0.020	0.888	**5068**	85	0.988
	1.0E-14	5475	145	5391	88	0.020	**5389**	87	0.016	0.912	**5389**	87	0.994
ca-netscience/ uniform	0.2	33042	1289	33007	1023	0.712	**32398**	814	0.038	0.018	32399	861	0.976
	0.1	34568	1302	34514	1028	0.745	**33907**	815	0.031	0.019	33908	865	0.988
	0.01	38189	1334	38089	1040	0.848	**37486**	821	0.019	0.019	37489	874	0.941
	1.0E-4	43012	1380	42846	1054	1.000	**42248**	841	0.011	0.020	42252	881	0.988
	1.0E-6	46591	1415	46377	1065	0.824	**45783**	858	0.009	0.023	45786	888	0.882
	1.0E-8	49557	1442	49303	1076	0.712	**48712**	870	0.008	0.021	48717	896	0.894
	1.0E-10	52145	1465	51857	1087	0.615	**51266**	883	0.009	0.028	51275	906	0.935
	1.0E-12	54467	1487	54150	1096	0.564	**53557**	894	0.007	0.028	53570	914	0.923
	1.0E-14	56592	1507	56249	1105	0.487	**55653**	905	0.006	0.029	55670	923	0.912
ca-GrQc /uniform	0.2	5646101	79194	9666938258	1824254292	0.000	**4920986**	45094	0.000	0.000	4924856	40968	0.756
	0.1	5712770	79494	9666940921	1824239705	0.000	**4983403**	45308	0.000	0.000	4987255	41159	0.756
	0.01	5871104	80213	9666947246	1824205061	0.000	**5131640**	45823	0.000	0.000	5135447	41621	0.790
	1.0E-4	6082155	81182	9666955677	1824158882	0.000	**5329219**	46511	0.000	0.000	5332980	42256	0.767
	1.0E-6	6238913	81909	9666961940	1824124582	0.000	**5475970**	47028	0.000	0.000	5479688	42740	0.779
	1.0E-8	6369023	82517	9666967137	1824096113	0.000	**5597768**	47451	0.000	0.000	5601451	43151	0.802
	1.0E-10	6482579	83051	9666971674	1824071266	0.000	**5704069**	47822	0.000	0.000	5707719	43516	0.779
	1.0E-12	6584589	83534	9666975749	1824048945	0.000	**5799561**	48160	0.000	0.000	5803181	43848	0.767
	1.0E-14	6677976	83978	9666979480	1824028511	0.000	**5886980**	48471	0.000	0.000	5890573	44156	0.767
Erdos992/ uniform	0.2	13716872	82588	10000000000	0	0.000	13482678	62860	0.000	0.000	**13477560**	55830	0.848
	0.1	13842990	82789	10000000000	0	0.000	13607667	62812	0.000	0.000	**13602550**	55731	0.848
	0.01	14142509	83278	10000000000	0	0.000	13904505	62706	0.000	0.000	**13899386**	55512	0.813
	1.0E-4	14541754	83954	10000000000	0	0.000	14300178	62586	0.000	0.000	**14295055**	55242	0.790
	1.0E-6	14838295	84474	10000000000	0	0.000	14594065	62512	0.000	0.000	**14588938**	55059	0.836
	1.0E-8	15084429	84917	10000000000	0	0.000	14837996	62461	0.000	0.000	**14832866**	54917	0.836
	1.0E-10	15299247	85313	10000000000	0	0.000	15050890	62423	0.000	0.000	**15045759**	54801	0.824
	1.0E-12	15492221	85674	10000000000	0	0.000	15242138	62395	0.000	0.000	**15237005**	54703	0.836
	1.0E-14	15668883	86011	10000000000	0	0.000	15417219	62375	0.000	0.000	**15412085**	54619	0.871
cfat200-1/ degree	0.2	4444	115	**4387**	6	0.001	4407	75	0.011	0.535	4398	55	0.779
	0.1	4781	119	**4721**	9	0.003	4742	77	0.023	0.446	4733	56	0.790
	0.01	5582	129	**5512**	16	0.004	5535	83	0.036	0.348	5526	61	0.819
	1.0E-4	6650	143	**6566**	26	0.003	6592	91	0.035	0.287	6584	68	0.830
	1.0E-6	7443	154	**7349**	34	0.003	7378	98	0.037	0.268	7369	74	0.830
	1.0E-8	8101	163	**7999**	40	0.003	8029	103	0.041	0.268	8021	79	0.830
	1.0E-10	8675	171	**8567**	45	0.003	8598	108	0.044	0.261	8590	84	0.865
	1.0E-12	9191	178	**9076**	50	0.003	9109	113	0.043	0.261	9101	88	0.865
	1.0E-14	9663	185	**9542**	55	0.003	9577	118	0.041	0.268	9569	92	0.853
cfat200-2/ degree	0.2	3041	172	**2963**	4	0.027	**2963**	4	0.027	0.929	**2963**	4	0.830
	0.1	3267	178	**3185**	6	0.027	**3185**	6	0.027	0.929	**3185**	6	0.830
	0.01	3803	194	3713	11	0.027	3713	10	0.027	0.929	**3712**	10	0.830
	1.0E-4	4518	216	4416	17	0.027	**4415**	16	0.027	0.929	**4415**	17	0.830
	1.0E-6	5049	232	4938	22	0.027	**4937**	21	0.027	0.929	**4937**	21	0.830
	1.0E-8	5490	245	5371	26	0.027	5371	24	0.027	0.929	**5370**	25	0.830
	1.0E-10	5875	257	5749	30	0.027	5749	28	0.027	0.929	**5748**	28	0.830
	1.0E-12	6220	267	6089	33	0.027	6088	30	0.027	0.929	**6087**	31	0.830
	1.0E-14	6537	277	6400	36	0.027	6399	33	0.027	0.929	**6398**	34	0.830
ca-netscience/ degree	0.2	28164	1002	26169	196	0.000	**26097**	197	0.000	0.017	26098	193	0.900
	0.1	29689	1029	27657	200	0.000	**27580**	207	0.000	0.038	27583	201	0.853
	0.01	33300	1098	31183	216	0.000	**31092**	238	0.000	0.114	31098	224	0.848
	1.0E-4	38103	1192	35874	251	0.000	**35758**	284	0.000	0.092	35767	266	0.813
	1.0E-6	41665	1265	39355	285	0.000	**39220**	324	0.000	0.076	39230	303	0.813
	1.0E-8	44620	1327	42243	317	0.000	**42091**	359	0.000	0.067	42103	336	0.813
	1.0E-10	47198	1381	44763	347	0.000	**44596**	390	0.000	0.067	44610	366	0.784
	1.0E-12	49514	1429	47026	374	0.000	**46845**	418	0.000	0.074	46861	394	0.842
	1.0E-14	51633	1474	49098	400	0.000	**48905**	444	0.000	0.081	48921	419	0.830
ca-GrQc/ degree	0.2	4032668	60538	9666845352	1824763160	0.000	**3455870**	17041	0.000	0.000	3457971	16176	0.460
	0.1	4100297	61062	9666847956	1824748898	0.000	**3517608**	17204	0.000	0.000	3519680	16336	0.442
	0.01	4260901	62312	9666854140	1824715027	0.000	**3664186**	17591	0.000	0.000	3666208	16722	0.442
	1.0E-4	4474975	63984	9666862383	1824669878	0.000	**3859529**	18140	0.000	0.000	3861506	17249	0.408
	1.0E-6	4633978	65230	9666868505	1824636344	0.000	**4004604**	18568	0.000	0.000	4006550	17648	0.460
	1.0E-8	4765953	66266	9666873587	1824608510	0.000	**4125007**	18930	0.000	0.000	4126935	17988	0.469
	1.0E-10	4881136	67173	9666878022	1824584217	0.000	**4230080**	19246	0.000	0.000	4232003	18291	0.460
	1.0E-12	4984607	67988	9666882006	1824562394	0.000	**4324470**	19538	0.000	0.000	4326386	18569	0.451
	1.0E-14	5079332	68736	9666885654	1824542416	0.000	**4410880**	19810	0.000	0.000	4412792	18828	0.478
Erdos992/ degree	0.2	9307396	60880	10000000000	0	0.000	9104433	4932	0.000	0.000	**9104421**	4931	0.965
	0.1	9433699	61228	10000000000	0	0.000	9229249	5100	0.000	0.000	**9229244**	4958	0.906
	0.01	9733657	62061	10000000000	0	0.000	**9525667**	5566	0.000	0.000	9525686	5110	0.988
	1.0E-4	10133488	63184	10000000000	0	0.000	**9920775**	6299	0.000	0.000	9920827	5490	0.953
	1.0E-6	10430463	64027	10000000000	0	0.000	**10214242**	6902	0.000	0.000	10214318	5882	0.941
	1.0E-8	10676958	64732	10000000000	0	0.000	**10457822**	7430	0.000	0.000	10457921	6265	0.988
	1.0E-10	10892090	65351	10000000000	0	0.000	**10670412**	7907	0.000	0.000	10670530	6635	0.976
	1.0E-12	11085348	65911	10000000000	0	0.000	**10861385**	8347	0.000	0.000	10861521	6990	0.976
	1.0E-14	11262269	66425	10000000000	0	0.000	**11036215**	8757	0.000	0.000	11036367	7332	1.000

Table 2. Results for stochastic minimum weight dominating set with different confidence levels of α where $\alpha = 1 - \beta$. Results after 1M fitness evaluations. p_1: Test (1+1) EA vs GSEMO2D, p_2: Test (1+1) EA vs Fast SW-GSEMO3D, p_3: Test GSEMO2D vs Fast SW-GSEMO3D, p_4: Test (1+1) EA vs Fast SW-GSEMO3D$_0$, p_5: Test GSEMO2D vs Fast SW-GSEMO3D$_0$, p_6: Test Fast GSEMO3D vs Fast SW-GSEMO3D$_0$.

Graph/ weight type	β	(1+1) EA [14]		GSEMO2D [14, 15]			Fast SW-GSEMO3D				Fast SW-GSEMO3D$_0$				
		Mean	Std	Mean	Std	p_1-val	Mean	Std	p_2-val	p_3-val	Mean	Std	p_4-val	p_5-val	p_6-val
ca-CSphd/ uniform	0.2	1176951	29560	1149185	21187	0.000	1053428	5919	0.000	0.000	**1052480**	4910	0.000	0.000	0.367
	0.1	1200964	25599	1173498	21419	0.000	1076406	5973	0.000	0.000	**1075454**	4965	0.000	0.000	0.367
	0.01	1235668	29329	1231241	21969	0.836	1130976	6108	0.000	0.000	**1130017**	5105	0.000	0.000	0.383
	1.0E-4	1314570	28190	1308208	22705	0.451	1203715	6301	0.000	0.000	**1202747**	5308	0.000	0.000	0.375
	1.0E-6	1378890	25618	1365376	23254	0.062	1257743	6455	0.000	0.000	**1256767**	5471	0.000	0.000	0.391
	1.0E-8	1410240	22358	1412826	23711	0.712	1302586	6587	0.000	0.000	**1301605**	5612	0.000	0.000	0.383
	1.0E-10	1455663	21030	1454239	24110	0.894	1341724	6707	0.000	0.000	**1340738**	5740	0.000	0.000	0.375
	1.0E-12	1495936	29008	1491441	24470	0.574	1376883	6818	0.000	0.000	**1375892**	5859	0.000	0.000	0.433
	1.0E-14	1526403	25752	1525499	24799	1.000	1409069	6921	0.000	0.000	**1408074**	5970	0.000	0.000	0.469
ca-HepPh/ uniform	0.2	24866045	323815	24664260	251849	0.010	21903190	229592	0.000	0.000	**21655867**	211163	0.000	0.000	0.000
	0.1	25126756	223438	24941951	253168	0.006	22162387	230935	0.000	0.000	**21913353**	212217	0.000	0.000	0.000
	0.01	25709929	219138	25601440	256304	0.101	22777957	234129	0.000	0.000	**22524858**	214726	0.000	0.000	0.000
	1.0E-4	26602650	271535	26480507	260496	0.132	23598486	238398	0.000	0.000	**23339968**	218088	0.000	0.000	0.000
	1.0E-6	27104133	304517	27133437	263618	0.595	24207935	241578	0.000	0.000	**23945393**	220598	0.000	0.000	0.000
	1.0E-8	27675018	335011	27675380	266213	0.953	24713788	244221	0.000	0.000	**24447901**	222696	0.000	0.000	0.000
	1.0E-10	28123068	314336	28148371	268482	0.941	25155281	246532	0.000	0.000	**24886470**	224540	0.000	0.000	0.000
	1.0E-12	28616742	357514	28573268	270522	0.636	25551883	248611	0.000	0.000	**25280441**	226199	0.000	0.000	0.000
	1.0E-14	28831138	286317	28962248	272392	0.143	25914960	250516	0.000	0.000	**25641110**	227723	0.000	0.000	0.000
ca-AstroPh/ uniform	0.2	51557918	568600	51043030	528254	0.001	64103184	4470490	0.000	0.000	**45226809**	500442	0.000	0.000	0.000
	0.1	51942457	555700	51548678	531285	0.021	64668884	4491484	0.000	0.000	**45698407**	502905	0.000	0.000	0.000
	0.01	53161346	658583	52749539	538490	0.017	66012371	4541343	0.000	0.000	**46818411**	508759	0.000	0.000	0.000
	1.0E-4	54581672	577272	54350226	548110	0.160	67803180	4607807	0.000	0.000	**48311327**	516571	0.000	0.000	0.000
	1.0E-6	55574306	568036	55539139	555269	0.965	69133305	4657175	0.000	0.000	**49420191**	522386	0.000	0.000	0.000
	1.0E-8	56482376	659036	56525957	561218	0.525	70237331	4698155	0.000	0.000	**50340566**	527216	0.000	0.000	0.000
	1.0E-10	56997947	442067	57387223	566415	0.003	71200892	4733922	0.000	0.000	**51143842**	531435	0.000	0.000	0.000
	1.0E-12	58002729	535712	58160914	571088	0.255	72066476	4766054	0.000	0.000	**51865440**	535228	0.000	0.000	0.000
	1.0E-14	58598177	480173	58869203	575369	0.033	72858892	4795471	0.000	0.000	**52526040**	538702	0.000	0.000	0.000
ca-CondMat/ uniform	0.2	87564936	940507	86293144	783450	0.000	431800766	1807172824	0.000	0.000	**75931086**	610598	0.000	0.000	0.000
	0.1	87993459	758163	87014750	786716	0.000	432555511	1807030509	0.000	0.000	**76602241**	613185	0.000	0.000	0.000
	0.01	89127748	754815	88728501	794478	0.069	434347964	1806692530	0.000	0.000	**78196177**	619334	0.000	0.000	0.000
	1.0E-4	91086972	739979	91012856	804836	0.859	436737226	1806242026	0.000	0.000	**80320825**	627546	0.000	0.000	0.000
	1.0E-6	92467544	650611	92709566	812539	0.204	438511855	185907420	0.000	0.000	**81898913**	633655	0.000	0.000	0.000
	1.0E-8	93588939	815736	94117866	818937	0.015	439984829	1805629695	0.000	0.000	**83208753**	638731	0.000	0.000	0.000
	1.0E-10	94695345	520061	95346987	824526	0.002	441270396	1805387308	0.000	0.000	**84351942**	643167	0.000	0.000	0.000
	1.0E-12	96086744	975803	96451130	829550	0.183	442425245	1805169569	0.000	0.000	**85378890**	647155	0.000	0.000	0.000
	1.0E-14	96686021	889063	97461938	834151	0.001	443482473	1804970239	0.000	0.000	**86319029**	650809	0.000	0.000	0.000
ca-CSphd/ degree	0.2	1176359	23453	1166190	32090	0.071	1053397	6005	0.000	0.000	**1052796**	5364	0.000	0.000	0.668
	0.1	1197763	27695	1190714	32418	0.322	1076425	6076	0.000	0.000	**1075804**	5419	0.000	0.000	0.657
	0.01	1243411	22570	1248957	33200	0.416	1131114	6256	0.000	0.000	**1130446**	5555	0.000	0.000	0.647
	1.0E-4	1318313	23041	1326592	34244	0.294	1204011	6514	0.000	0.000	**1203281**	5748	0.000	0.000	0.615
	1.0E-6	1370672	30267	1384255	35022	0.110	1258155	6718	0.000	0.000	**1257380**	5898	0.000	0.000	0.615
	1.0E-8	1411274	28448	1432117	35668	0.004	1303096	6895	0.000	0.000	**1302282**	6027	0.000	0.000	0.605
	1.0E-10	1465714	31864	1473889	36233	0.322	1342319	7054	0.000	0.000	**1341472**	6143	0.000	0.000	0.584
	1.0E-12	1494845	26265	1511414	36742	0.076	1377554	7202	0.000	0.000	**1376676**	6249	0.000	0.000	0.595
	1.0E-14	1539841	28989	1545767	37207	0.756	1409811	7339	0.000	0.000	**1408905**	6349	0.000	0.000	0.584
ca-HepPh/ degree	0.2	24940255	229915	24770247	328453	0.019	21925184	256481	0.000	0.000	**21672753**	170643	0.000	0.000	0.000
	0.1	25129650	329488	25048454	330378	0.322	22184365	258158	0.000	0.000	**21930197**	171670	0.000	0.000	0.000
	0.01	25755684	291735	25709164	334958	0.584	22799898	262146	0.000	0.000	**22541605**	174114	0.000	0.000	0.000
	1.0E-4	26478853	313950	26589855	341060	0.156	23620377	267470	0.000	0.000	**23356586**	177388	0.000	0.000	0.000
	1.0E-6	27073736	305766	27243988	345599	0.055	24229788	271431	0.000	0.000	**23961915**	179829	0.000	0.000	0.000
	1.0E-8	27647166	283416	27786927	349368	0.086	24735610	274722	0.000	0.000	**24464348**	181862	0.000	0.000	0.000
	1.0E-10	28101126	327539	28260785	352656	0.079	25177076	277597	0.000	0.000	**24902857**	183641	0.000	0.000	0.000
	1.0E-12	28523939	323332	28686461	355612	0.071	25573654	280182	0.000	0.000	**25296778**	185243	0.000	0.000	0.000
	1.0E-14	28937484	306489	29076155	358320	0.147	25936707	282550	0.000	0.000	**25657400**	186712	0.000	0.000	0.000
ca-AstroPh/ degree	0.2	51524407	570578	50681144	611971	0.000	64564376	6120887	0.000	0.000	**45109042**	578792	0.000	0.000	0.000
	0.1	52090421	613178	51184838	615193	0.000	65131764	6149639	0.000	0.000	**45579883**	581545	0.000	0.000	0.000
	0.01	53271848	477150	52381067	622852	0.000	66479261	6217924	0.000	0.000	**46698084**	588091	0.000	0.000	0.000
	1.0E-4	54408644	498038	53975585	633070	0.012	68275417	6308950	0.000	0.000	**48188591**	596839	0.000	0.000	0.000
	1.0E-6	55533826	541501	55159915	640666	0.015	69609515	6376561	0.000	0.000	**49295668**	603348	0.000	0.000	0.000
	1.0E-8	56254153	556558	56142930	646976	0.469	70716840	6432682	0.000	0.000	**50214561**	608758	0.000	0.000	0.000
	1.0E-10	57221946	419431	57000876	652487	0.165	71683280	6481663	0.000	0.000	**51016543**	613485	0.000	0.000	0.000
	1.0E-12	58115722	620857	57771583	657440	0.048	72551451	6525664	0.000	0.000	**51736978**	617736	0.000	0.000	0.000
	1.0E-14	58619146	587556	58477140	661975	0.433	73346234	6565947	0.000	0.000	**52396514**	621630	0.000	0.000	0.000
ca-CondMat/ degree	0.2	87579791	869378	86547921	742313	0.000	103694132	9559675	0.000	0.000	**75939104**	868161	0.000	0.000	0.000
	0.1	87713127	806685	87270373	745665	0.028	104482533	9598957	0.000	0.000	**76609900**	872129	0.000	0.000	0.000
	0.01	89207002	639620	88986134	753635	0.132	106354916	9692249	0.000	0.000	**78202983**	881557	0.000	0.000	0.000
	1.0E-4	90856334	857580	91273168	764272	0.110	108850723	9816605	0.000	0.000	**80326490**	894132	0.000	0.000	0.000
	1.0E-6	92392083	865712	92971867	772182	0.017	110704488	9908973	0.000	0.000	**81903730**	903477	0.000	0.000	0.000
	1.0E-8	93674413	808337	94381818	778752	0.003	112243147	9985641	0.000	0.000	**83212867**	911237	0.000	0.000	0.000
	1.0E-10	94664923	600565	95612380	784491	0.000	113586040	10052555	0.000	0.000	**84355442**	918012	0.000	0.000	0.000
	1.0E-12	96145670	762399	96717817	789649	0.017	114792388	10112666	0.000	0.000	**85381839**	924101	0.000	0.000	0.000
	1.0E-14	96814075	636459	97729809	794374	0.000	115896760	10167696	0.000	0.000	**86321473**	929676	0.000	0.000	0.000

Table 3. Average maximum population size and standard deviation during the 30 runs of 1M iterations for Fast SW-GSEMO3D and Fast SW-GSEMO3D$_0$ in the uniform random, degree-based setting for large graphs.

Graph	Fast SW-GSEMO3D				$FastSW-GSEMO3D_0$			
	uniform		degree		uniform		degree	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
ca-CSphd	665	40.555	670	37.727	225	16.829	230	15.616
ca-HepPh	2770	124.786	2713	166.804	125	15.561	128	20.372
ca-AstroPh	3608	167.880	3602	132.344	140	26.422	144	25.647
ca-CondMat	5196	130.968	5245	109.568	107	20.817	104	19.662

pairs to check whether results are statistically significant, which we assume to be the case if the p-value is at most 0.05.

We first consider results for the instances already investigated in [15]. We consider the random and degree based instances and results for the examined algorithms are shown in Table 1. Results for the GSEMO2D approach developed in [14] and the GSEMO3D developed in [15] have already been obtained in [15]. Each run that does not obtain a dominating set gets allocated a fitness value of 10^{10}. We note that this only applies to GSEMO3D for ca-GrQc and Erdos992 and GSEMO3D. It has already been stated in [15] that GSEMO3D has difficulties in obtaining feasible solutions for these graphs. In fact, it never returns a feasible solution for Erdos992 in both chance constrained settings and only in 1 out of 30 runs for ca-GrQc in both chance constrained settings. Comparing the results of GSEMO2D and GSEMO3D to our new approaches Fast SW-GSEMO3D and Fast SW-GSEMO3D$_0$, we can see that all approaches behave quite similar for cfat200-1 and cfat200-2. For ca-netscience, there is a slight advantage for our fast sliding window approaches that is statistically significant when compared to GSEMO2D and GSEMO3D, but no real difference on whether the sliding window approach starts with an initial solution chosen uniformly at random or with the search point 0^n. Both Fast SW-GSEMO3D and Fast SW-GSEMO3D$_0$ show their real advantage for the larger graphs ca-GrQc and Erdos992 where the 3-objective approach GSEMO3D was unable to produce feasible solutions. On these instance GSEMO2D is clearly outperformed by the sliding window 3-objective approaches.

Results for the instances based on the graphs ca-CSphd, ca-HepPh, ca-AstroPh, ca-CondMat, which consist of 1882, 4158, 6100, 11204, 17903, 21363 nodes are shown in Table 2. Note that the graphs (except ca-CSphd) have more than 10000 nodes and are therefore significantly larger than the ones tested previously. As we are dealing with larger graphs and a smaller fitness evaluation budget of 1M, we also consider the (1+1) EA approach presented in [14]. Here each run of the (1+1) EA tackles each value of α (see Equation (2)) separately with a budget of 1M fitness evaluations, which implies the single-objective approach uses a fitness evaluation budget that is ten times the one of the multi-objective approaches. We observe that Fast SW-GSEMO3D$_0$ overall produces the best results. For the smallest graph ca-CSphd, there is no significant differ-

ence on whether to start with an initial solution chosen uniformly at random or with the search point 0^n. However, for the larger graphs ca-HepPh, ca-AstroPh, ca-CondMat consisting of more than 10000 nodes, starting with the initial search point 0^n in the sliding window approach is crucial for the success of the algorithm. In particular, we can observe that Fast SW-GSEMO3D starting uniformly at random is performing significantly worse than the (1+1) EA and GSEMO2D for the graphs ca-AstroPh, ca-CondMat consisting of 17903 and 21363 nodes, respectively. All observations hold for the uniform as well as the degree-based chance constrained settings.

As mentioned starting with 0^n in our sliding window approach provides a clear benefit when dealing with large graphs. We have already seen in our analysis that the sliding window approach starts at the constraint value of 0 which gives a partial explanation of its benefit. In order to gain additional insights, we provide in Table 3 the maximum population sizes that the approaches Fast SW-GSEMO3D and Fast SW-GSEMO3D$_0$ encounter for the graphs ca-CSphd, ca-HepPh, ca-AstroPh, ca-CondMat. We can observe that the maximum population sizes when starting with the search point 0^n are significantly smaller than when starting with an initial solution chosen uniformly at random. For the graph ca-CondMat, the average maximum population size among the executed 30 runs for Fast SW-GSEMO3D is almost by a factor of 50 larger than for Fast SW-GSEMO3D$_0$ (5196 vs. 107). Given that large populations can significantly slow down the progress of the sliding window approach, we regard the difference in maximum population sizes as a clear explanation why Fast SW-GSEMO3D$_0$ clearly outperforms Fast SW-GSEMO3D on the graphs ca-HepPh, ca-AstroPh, and ca-CondMat.

Conclusions

We have shown how to significantly speed and scale up the 3-objective approach for chance constrained problems introduced in [15]. We have presented a sliding window approach and shown that it provides with high probability the same theoretical approximation quality as the one given in [15] but within a significantly smaller fitness evaluation budget. Our experimental investigations show that the new approach is able to deal with chance constrained instances of the dominating set problem with up to 20,000 nodes (within 1M iterations) whereas the previous approach given in [15] was not able to produce good quality (or even feasible) solutions for already medium size instances of around 4,000 nodes (within 10M iterations).

Acknowledgments. This work has been supported by the Australian Research Council (ARC) through grant FT200100536 and by the Independent Research Fund Denmark through grant DFF-FNU 8021-00260B.

References

1. Doerr, B., Doerr, C., Neumann, A., Neumann, F., Sutton, A.M.: Optimization of chance-constrained submodular functions. In: AAAI, pp. 1460–1467. AAAI Press (2020)
2. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**, 673–697 (2012)
3. 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
4. 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)
5. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Proceedings of CEC '03, vol. 3, pp. 1918–1925 (2003). https://doi.org/10.1109/CEC.2003.1299908
6. Ishii, H., Shiode, S., Nishida, T., Namasuya, Y.: Stochastic spanning tree problem. Discret. Appl. Math. **3**(4), 263–273 (1981)
7. Jansen, T.: Analyzing Evolutionary Algorithms - The Computer Science Perspective. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-17339-4
8. Knowles, J.D., Watson, R.A., Corne, D.W.: Reducing local optima in single-objective problems by multi-objectivization. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 269–283. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44719-9_19
9. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica **65**(4), 754–771 (2013)
10. 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)
11. Neumann, A., Neumann, F.: Optimising monotone chance-constrained submodular functions using evolutionary multi-objective algorithms. In: Bäck, T., Preuss, M., Deutz, A., Wang, H., Doerr, C., Emmerich, M., Trautmann, H. (eds.) PPSN 2020. LNCS, vol. 12269, pp. 404–417. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58112-1_28
12. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Nat. Comput. **5**(3), 305–319 (2006)
13. 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
14. Neumann, F., Witt, C.: Runtime analysis of single- and multi-objective evolutionary algorithms for chance constrained optimization problems with normally distributed random variables. In: IJCAI, pp. 4800–4806. ijcai.org (2022)
15. Neumann, F., Witt, C.: 3-objective pareto optimization for problems with chance constraints. In: GECCO, pp. 731–739. ACM (2023)
16. Neumann, F., Witt, C.: Fast pareto optimization using sliding window selection. In: ECAI. Front. Artif. Intell. Appl. **372**, 1771–1778. IOS Press (2023)
17. Qian, C., Shi, J., Yu, Y., Tang, K.: On subset selection with general cost constraints. In: IJCAI, pp. 2613–2619 (2017). https://doi.org/10.24963/ijcai.2017/364
18. Qian, C., Yu, Y., Zhou, Z.: Subset selection by Pareto optimization. In: NIPS, pp. 1774–1782 (2015)

19. Roostapour, V., Neumann, A., Neumann, F., Friedrich, T.: Pareto optimization for subset selection with dynamic cost constraints. Artif. Intell. **302**, 103597 (2022)
20. Xie, Y., Harper, O., Assimi, H., Neumann, A., Neumann, F.: Evolutionary algorithms for the chance-constrained knapsack problem. In: GECCO, pp. 338–346. ACM (2019)
21. Xie, Y., Neumann, A., Neumann, F.: Specific single- and multi-objective evolutionary algorithms for the chance-constrained knapsack problem. In: GECCO, pp. 271–279. ACM (2020)
22. 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

Runtime Analysis of a Multi-valued Compact Genetic Algorithm on Generalized OneMax

Sumit Adak(✉) and Carsten Witt

DTU Compute, Technical University of Denmark, Kgs. Lyngby, Denmark
{suad,cawi}@dtu.dk

Abstract. A class of metaheuristic techniques called estimation-of-distribution algorithms (EDAs) is employed in optimization as a more sophisticated substitute for traditional strategies like evolutionary algorithms. EDAs generally drive the search for the optimum by creating probabilistic models of potential candidate solutions through repeated sampling and selection from the underlying search space.

Most theoretical research on EDAs has focused on pseudo-Boolean optimization. Jedidia et al. (GECCO 2023) introduced a framework for EDAs for optimizing problems involving multi-valued decision variables. In addition, they conduct a mathematical runtime analysis of a multi-valued UMDA on the r-valued LeadingOnes function. Using their framework, here we focus on the multi-valued compact genetic algorithm (r-cGA) and provide a first runtime analysis of a generalized OneMax function.

To prove our results, we investigate the effect of genetic drift and progress of the probabilistic model towards the optimum. After finding the right algorithm parameters, we prove that the r-cGA solves this r-valued OneMax problem efficiently. We establish that the runtime bound is $O(r^2 n \log^2 r \log^3 n)$ with high probability.

Keywords: Estimation-of-distribution algorithms · multi-valued compact genetic algorithm · genetic drift · OneMax

1 Introduction

The term estimation-of-distribution algorithms (EDAs) refers to an optimization method that creates a probabilistic model that is subsequently utilized to produce new search points based on previous searches. The fundamental distinction from evolutionary algorithms (EAs) is that they evolve a probabilistic model rather than a population. The performance of EDAs can be more efficient than EAs [5,8,18,37]. EDAs are constructed by carrying out three basic steps: first, a population of individuals is sampled using the current probabilistic model; second, the fitness of the population is determined; and finally, a new probabilistic model is generated depending on the fitness of the population.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 53–69, 2024.
https://doi.org/10.1007/978-3-031-70071-2_4

In this framework, various probabilistic models and updating methodologies provide distinct algorithms. The probabilistic model in multivariate EDAs includes inter-variable relationships. Some popular examples of multivariate EDAs include mutual-information-maximization input clustering (MIMIC) [7], bivariate marginal distribution algorithm (BMDA) [29], extended compact genetic algorithm (ecGA) [17], and many more. Another sort of EDA is a univariate EDA, where the positions of the probabilistic model are independent of each other. Examples of univariate EDAs are population-based incremental learning (PBIL) [3], univariate marginal distribution algorithm (UMDA) [26], compact genetic algorithm (cGA) [16]. Because the dependencies in multivariate EDAs are difficult to analyze mathematically, the majority of theoretical studies of EDAs focus on univariate models [21]. This manuscript likewise focuses on univariate EDAs.

In conventional genetic algorithms on bit strings, the frequency of bit values in the population are controlled by the bit's contribution to fitness as well as random fluctuations caused by other bits with a higher influence on fitness. Random fluctuations can even cause particular bits to converge to a single value that differs from the best solution. This effect is known as *genetic drift* [2,19]. Genetic drift is also observed in EDAs. According to Krejca and Witt [21], genetic drift in EDAs is a broad concept of martingales, which are random processes with zero expected change that finally may stop at absorbing bounds of the underlying interval. Furthermore, Witt [36] and Lengler et al. [23] demonstrated that, depending on the parameter, genetic drift might result in a significant performance loss on the OneMax function.

Classical evolutionary algorithms are commonly employed for a variety of search spaces, whereas EDAs have traditionally been used for issues involving binary decision variables. Further, researchers took the initial steps toward applying EDAs for situations involving decision variables with more than two values [4,30]. More specifically, in [4], Jedidia et al. consider univariate EDAs for multi-valued decision factors. They treat a multi-valued problem by introducing r probability values for each variables. Particularly, by building a framework, they analyzed the runtime of multi-valued UMDA on the r-valued LeadingOnes function. This article also deals with EDAs for multi-valued decision variables by using their framework. Particularly, we focus on the r-valued compact genetic algorithm (r-cGA) as defined in [4] without borders on frequencies.

Here we look at the multi-valued OneMax function. On the generalized LeadingOnes studied in [4], the function leads to perfect neutrality in the positions that are not yet relevant and a very strong fitness signal in all other positions. In contrast, on generalized OneMax, all positions have a weak fitness signal, yet the algorithm is able to identify the optimal values for all components if parameters are set appropriately. Therefore, the analysis of generalized OneMax is more complicated than that of generalized LeadingOnes. Informally, one main challenge is determining how these parameters should be configured.

In this paper, we provide the first runtime analysis of the multi-valued cGA and the first analysis of a multi-valued OneMax function. We perform a mathematical runtime of the r-cGA on the r-OneMax function with high probability. In our final analysis, we establish that the typical runtime of the r-cGA on r-OneMax problem is $O(r^2 n \log^2 r \log^3 n)$ (Theorem 5) with high probability. Further, we compare our bound with the bound of binary cGA and mention that our bound probably is not tight.

The article is structured as follows: Sect. 2 summarizes the earlier work on our technical topics. The following section establishes the terminologies and defines the multi-valued OneMax function. Section 4 elaborates on the multi-valued EDA framework for the r-cGA. Sections 5 and 6 include the major technical results, genetic drift analysis, and the runtime evaluation of the r-cGA on the r-valued OneMax. The experiments in Sect. 7 demonstrate the empirical runtime throughout the entire parameter range for the hypothetical population size (K), and we compare the empirical runtimes of the r-cGA on r-OneMax and G-OneMax. Finally, the manuscript concludes with a brief summary.

2 Related Work

This work is separated into three technical topics: the first one is the framework for EDAs, the next one concerns genetic drift, and finally a runtime evaluation on the multi-valued OneMax function. There are numerous theoretical papers on traditional evolutionary algorithms for multi-valued decision variables.

Model-based optimization approaches have enabled EDAs to tackle a wide area of large and complicated problems [13,14,34]. Droste conducted the initial runtime analysis for an EDA [13]. For a simple EDA, the compact genetic algorithm (cGA) was used with the OneMax function. They provide an overall lower and upper bounds for all functions. It was also noticed that EDAs optimize problems differently, as evidenced by the difference in runtime between two linear functions. This stands in contrast to the well-known analysis of how the (1+1) EA optimizes linear functions by Droste et al. [14].

Most theoretical research on EDAs has focused on pseudo-Boolean optimization [6,15,20]. Jedidia et al. [4] recently introduced a framework for EDAs for optimizing problems involving more than two decision variables from the domain $\{0, \ldots, r-1\}^n$. They prove that the multi-valued UMDA solve the r-valued LeadingOnes problem efficiently. Overall, they demonstrate how EDAs can be tailored to multi-valued issues and used to assist define their parameters. At present, there is a very active research area analyzing EDAs on complex problems. Further, in [21,22,28], one can find out more details about theory and practice of EDAs.

3 Preliminaries

We are considering the r-valued compact genetic algorithm (r-cGA) introduced by [4] to maximize an r-valued OneMax function. Here, we're looking at the

maximization of functions of the kind $f\colon \{0,1,\dots,r-1\}^n \to \mathbb{R}$, also known as r-valued fitness functions. We define $f(x)$ as the *fitness* of x, where $x \in \{0,1,\dots,r-1\}^n$ is an individual.

In this work, we discuss two different types of multi-valued OneMax functions: one is r-OneMax and another is G-OneMax. Let $n \in \mathbb{N}_{\geq 1}$ and $r \in \mathbb{N}_{\geq 2}$. In the following, we give the definition of r-OneMax and G-OneMax, where, for all $x = (x_1,\dots,x_n) \in \{0,1,\dots,r-1\}^n$,

$$r\text{-OneMax}(x) := \sum_{i=1}^{n} \mathbb{1}\{x_i = r-1\}$$

$$G\text{-OneMax}(x) := \sum_{i=1}^{n} x_i.$$

In both functions, the single maximum is the string all-$(r-1)$s. A general variant can be derived by selecting a random optimum $a \in \{0,\dots,r-1\}^n$ and defining, for all $b \in \{0,\dots,r-1\}^n$, $r\text{-OneMax}_a(b) = n - d(b,a)$, where $d(b,a) = \sum_{i=1}^{n} \mathbb{1}\{b_i \neq a_i\}$ specifies the distance between the strings a and b. Similarly, we can define the generalized G-OneMax. For an arbitrary optimum $a \in \{0,\dots,r-1\}^n$, and defining, for all $b \in \{0,\dots,r-1\}^n$, $G\text{-OneMax}_a(b) = n\cdot(r-1) - d(b,a)$ where $d(b,a) = \sum_{i=1}^{n} \min\{|a_i - b_i|, r - |b_i - a_i|\}$ specifies the distance between a and b. The main difference between these two functions is that in r-OneMax the values are categorical values and only the right value for a position to contribute to the fitness, whereas in G-OneMax there is a distance metric between values and each position has a fitness signal towards the optimal value. Here, r-OneMax only distinguishes between whether value $r-1$ is taken at a position or not, while G-OneMax takes all $r \in \{0,\dots,r-1\}$ values per position into account. The maximum fitness value for r-OneMax is n, and for G-OneMax it is $n(r-1)$.

A random variable Z is said to *stochastically dominate* another random variable Y, denoted by $Z \succeq Y$, if and only if we have $\Pr[Z \leq \lambda] \leq \Pr[Y \leq \lambda]$ for all $\lambda \in \mathbb{R}$.

4 The Framework

In [4], Jedidia et al. introduced a framework for EDAs for multi-valued decision variables that are not permutation problems. In this paper, we adopt their framework to characterize the underlying probabilistic model. Here, we concentrate on the r-valued cGA.

r-cGA: The Compact genetic algorithm (cGA) [16] is a widely used univariate EDA. The cGA has only one parameter, $K \in \mathbb{N}_{\geq 1}$, which refers to the so-called hypothetical population size [8] and it maintains a vector of probabilities (called frequencies). Each iteration creates two solutions independently. After computing the fitness value, it changes each frequency by $1/K$ so that the frequency of the better sample increases while the frequency of the worse sample decreases.

The r-cGA is an expanded variant of cGA that takes into account multiple variables. The r-cGA, outlined in Algorithm 1, employs marginal probabilities (again denoted as frequencies) $p_{i,j}^{(t)}$ corresponding to the probability of position i and value j at time t. The sampling distribution generates two solutions, x and y, independently in each iteration. After that, the fitter offspring is determined among x and y, and the frequencies are modified by $\pm 1/K$ in the prospective direction towards better offspring for positions where both offspring vary. In this way, $1/K$ indicates the strength of the probabilistic model update.

Algorithm 1: r-valued Compact Genetic Algorithm (r-cGA) for the maximization of $f : \{0, \ldots, r-1\}^n \to \mathbb{R}$

Initialization : $t \leftarrow 0$
$p_{i,0}^{(t)} \leftarrow p_{i,1}^{(t)} \leftarrow p_{i,2}^{(t)} \cdots \leftarrow p_{i,r-1}^{(t)} \leftarrow \frac{1}{r}$ where $i \in \{1, 2, \ldots, n\}$
while *termination criterion not met* **do**
  for $i \in \{1, 2, \ldots, n\}$ **do**
    $x_i \leftarrow j$ with probability $p_{i,j}^{(t)}$ w.r.t. $j = 0, \ldots, r-1$, independently for all i
    $y_i \leftarrow j$ with probability $p_{i,j}^{(t)}$ w.r.t. $j = 0, \ldots, r-1$, independently for all i
  if $f(x) < f(y)$ **then**
    swap $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$
  for $i \in \{1, 2, \ldots, n\}$ **do**
    for $j \in \{0, 1, \ldots, r-1\}$ **do**
      $p_{i,j}^{(t+1)} \leftarrow p_{i,j}^{(t)} + \frac{1}{K}(\mathbb{1}\{x_i = j\} - \mathbb{1}\{y_i = j\})$
  $t \leftarrow t + 1$

The Probabilistic Model: This paragraph depicts the stochastic process in the algorithm. Let $p_{i,j}^{(t)}$ be the marginal probability at time t for arbitrary position i and value j where $(i, j) \in \{1, \ldots, n\} \times \{0, \ldots, r-1\}$. An r-valued EDA's probabilistic model is an $n \times r$ matrix of $(p_{i,j}^{(t)})_{i,j}$ (the frequency matrix), with each row i forming a frequency vector of probabilities that sum to 1. When constructing an individual $x \in \{0, \ldots, r-1\}^n$, for all $i \in \{1, \ldots, n\}$ and all $j \in \{0, \ldots, r-1\}$, the probability of x_i having value j is $p_{i,j}^{(t)}$. For every $y \in \{0, \ldots, r-1\}^n$, we can state that $\Pr[x = y] = \Pi_{i \in \{1,\ldots,n\}} \Pi_{j \in \{0,\ldots,r-1\}} (p_{i,j}^{(t)})^{\mathbb{1}\{y_i = j\}}$, where we assume that $0^0 = 1$. We use the uniform distribution to initialize the frequency matrix, where each frequency gets the value of $1/r$.

After the update, each frequency vector in the r-cGA sums to 1, since one frequency is increased by $1/K$ and one frequency is decreased by the same quantity. We are interested in the number of function evaluations the r-cGA performs before sampling the optimum. This quantity is also known as runtime or optimization time.

Note that, in this work, contrary to the model in [4], the marginal probabilities are not restricted to some specific intervals. The lower and upper borders

on frequencies are 0 and 1. We do not use the borders from [4] since otherwise the analysis will be much more complicated. We make the following *well-behaved frequency assumption*: the r-cGA of any two frequencies can vary by a factor of $1/K$. In the absence of borders, the r-cGA can employ frequencies in $\{0, 1/K, 2/K, \ldots, 1\}$, where $1/r$ is a multiple of $1/K$.

The change of $p_{i,j}^{(t)}$ in one step is defined as $\Delta_{i,j} := \Delta_{i,j}^t := p_{i,j}^{(t+1)} - p_{i,j}^{(t)}$. Therefore, we can write $\Delta_{i,r-1} := p_{i,r-1}^{(t+1)} - p_{i,r-1}^{(t)}$. This change is determined by whether the value of position i influences the decision to update based on the first string x sampled at time t or the second string y. Particularly, we inspect the changes in the r-OneMax value at all positions except i. To achieve this, we define $D_i := \left(\sum_{j\neq i} \mathbb{1}\{x_j = r-1\} - \sum_{j\neq i} \mathbb{1}\{y_j = r-1\}\right)$.

At this point, the r-cGA experiences two different kinds of steps which we discuss below. The following analysis and the terms "rw-steps" and "biased steps" follow closely the one from [34] for the binary cGA.

Random-walk steps: If $|D_i| \geq 2$, position i has no impact on the decision to update with respect to string x or y. With $\Delta_{i,r-1} \neq 0$, it is necessary that position i for value $r-1$ is sampled differently. That means, the value of $p_{i,r-1}^{(t)}$ will be increased or decreased by $1/K$ with equal probability $p_{i,r-1}^{(t)}(1 - p_{i,r-1}^{(t)})$. Otherwise, based on the remaining probability, it holds $p_{i,r-1}^{(t+1)} = p_{i,r-1}^{(t)}$. Now, we can describe this by taking a variable F_i where

$$F_i := \begin{cases} +1/K & \text{with probability } p_{i,r-1}^{(t)}(1 - p_{i,r-1}^{(t)}), \\ -1/K & \text{with probability } p_{i,r-1}^{(t)}(1 - p_{i,r-1}^{(t)}), \\ 0 & \text{with the remaining probability.} \end{cases}$$

A step where $|D_i| \geq 2$ is referred to as a *random-walk step (rw-step)* because the process is a fair random walk (along with self-loops) as $E(\Delta_{i,r-1} \mid p_{i,r-1}^{(t)}, |D_i| \geq 2) = E(F_i \mid p_{i,r-1}^{(t)}) = 0$.

If $D_i = 1$, then $(\sum_{i=1}^n \mathbb{1}\{x_i = r-1\}) \geq (\sum_{i=1}^n \mathbb{1}\{y_i = r-1\})$ and for that strings x and y are never swapped in the r-cGA. So, as previous, here we obtain the same argumentation. In addition, the process also executes a *rw-step*.

Biased steps: If $D_i = -1$, then the strings x and y are swapped unless position i is sampled as $x_i = r-1$ and $y_i \neq r-1$. As a result, both the events of sampling position i raise the $p_{i,r-1}^{(t)}$ value in different ways. So, we obtain $\Delta_{i,r-1} = 1/K$ with probability $2p_{i,r-1}^{(t)}(1 - p_{i,r-1}^{(t)})$ and $\Delta_{i,r-1} = 0$ else.

If $D_i = 0$, then both the events of sampling position i raise the $p_{i,r-1}^{(t)}$ value differently, similar to the previously examined situation ($D_i = -1$). Again, we have $\Delta_{i,r-1} = 1/K$ with the probability $2p_{i,r-1}^{(t)}(1 - p_{i,r-1}^{(t)})$ and $\Delta_{i,r-1} = 0$ else. Let us take a random variable B_i where

$$B_i := \begin{cases} +1/K & \text{with probability } 2p_{i,r-1}^{(t)}(1 - p_{i,r-1}^{(t)}), \\ 0 & \text{with the remaining probability.} \end{cases}$$

For $D_i = -1$ and $D_i = 0$, we can conclude that $\Delta_{i,r-1}$ follows the same distribution as B_i. A *biased step (b-step)* occurs when $E(\Delta_{i,r-1} \mid p^{(t)}_{i,r-1}, D_i \in \{-1,0\}) = E(B_i \mid p^{(t)}_{i,r-1}) = 2p^{(t)}_{i,r-1}(1-p^{(t)}_{i,r-1})/K > 0$.

The event whether a step is a *rw-step* or a *b-step* for position i is merely depending on external factors that are stochastically independent of the outcome of position i. Let R_i represent the occurrence where $D_i = 1$ or $|D_i| \geq 2$. At the end, we get the following equality:

$$\Delta_{i,r-1} = F_i \cdot \mathbb{1}\{R_i\} + B_i \cdot \mathbb{1}\{\overline{R_i}\} \tag{1}$$

which we call *superposition*. Informally, the value change in $p^{(t)}_{i,r-1}$ is a superposition of a unbiased random walk and biased steps.

5 Genetic Drift for the r-Valued cGA

We prove the results of an upper bound on the influence of genetic drift for r-valued EDAs, similar to [4,11]. This enables us to select parameter values for the EDAs that avoid the often undesirable effect of genetic drift. This section provides a general overview of genetic drift, followed by a concentration result for neutral positions. Finally, there is an upper bound for positions with *weak preference*. The results of this section have already been presented for the UMDA [4]. Here, we prove similar results for the r-cGA.

In EDAs, genetic drift occurs when a frequency does not reach extreme values 1 or 0 as a result of a clear signal from the objective function, but rather as a result of random fluctuations caused by the process's stochasticity. Researchers have explored genetic drift in EDAs explicitly [31–33] and conducted numerous runtime analyses [10,12,24,34–36]. We analyze genetic drift for multi-valued EDAs, particularly for the r-cGA, based on insights from [4] and the framework from [11].

Genetic drift is typically examined using a fitness function in the *neutral* position. Let f denote an r-valued fitness function. A position $i \in \{1,\dots,n\}$ is called *neutral* (in relation to f) if and only if x_i has no effect on the value of f for all $x \in \{0,\dots,r-1\}^n$, and we have $f(x) = f(x')$ for each $x, x' \in \{0,\dots,r-1\}^n$ such that $x_j = x'_j$ for all $j \in \{1,\dots,n\} \setminus \{i\}$.

The frequencies of neutral variables in traditional EDAs without margins create martingales, which is useful for analyzing genetic drift [11]. This finding applies to EDAs with binary representation. Furthermore, the concept can be carried over to the r-UMDA [4], too. We make this argument specific to the r-cGA. Due to page restrictions, some of our proofs are not included in this paper. One can find them in the preprint [1].

Lemma 1. *Let f be an r-valued fitness function, and let $i \in \{1,\dots,n\}$ be a neutral position of f. Consider the r-cGA without margins, optimizing f. Then, the frequencies $(p^{(t)}_{i,j})_{t\in\mathbb{N}}$ are a martingale for all $j \in \{0,\dots,r-1\}$.*

In [4], all frequencies of an EDA start at a value $1/r$ and analyses for smaller deviations in both direction up to $1/(2r)$. In this article, for the r-cGA we follow the same frequency setting starting from $1/r$ and tolerate changes by up to $1/(2r)$ in either direction.

We apply a martingale concentration result [25, Theorem 3.13] to exploit the lower sampling variance at the frequencies in $\Theta(1/r)$. In fact, we restate an adjusted version of a theorem by McDiarmid [25, eq. (41)] that was used by Doerr and Zheng [11] and by Jedidia et al. [4].

Theorem 1. *Let $a_1, \dots, a_m \in \mathbb{R}$, and $X_1, \dots, X_m$ be a martingale difference sequence with $|X_k| \leq a_k$ for each k. Then for all $\varepsilon \in \mathbb{R}_{\geq 0}$, it holds that*

$$\Pr\left[\max_{k=1,\dots,m} \left|\sum_{i=1}^{k} X_i\right| \geq \varepsilon\right] \leq 2\exp\left(-\frac{\varepsilon^2}{2\sum_{i=1}^{m} a_i^2}\right).$$

Next, we use Theorem 1 to demonstrate for how much time the frequencies of the r-cGA at neutral places remain around the starting value of $p_{i,j}^{(0)}$ (which is usually $1/r$).

Theorem 2. *Let f be an r-valued fitness function with a neutral position $i \in \{1, \dots, n\}$. Consider the r-cGA optimizing f with population size K. Then, for $j \in \{0, \dots, r-1\}$ and $T \in \mathbb{N}$, we have*

$$\Pr\left[\max_{t \in \{0,\dots,T\}} \left|p_{i,j}^{(t)} - p_{i,j}^{(0)}\right| \geq \frac{1}{2r}\right] \leq 2\exp\left(-\frac{K^2}{8Tr^2}\right).$$

In many situations, for a given fitness function the positions are not neutral. However, we demonstrate that the outcomes for neutral positions apply to the positions in which one value is better than all other values. This is referred to as *weak preference* [11]. An r-valued fitness function f has a weak preference at a position $i \in \{1, \dots, n\}$ for a value $j \in \{0, \dots, r-1\}$ if and only if, for all $x_1, \dots, x_n \in \{0, \dots, r-1\}$, and it holds that

$$f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) \leq f(x_1, \dots, x_{i-1}, j, x_{i+1}, \dots, x_n).$$

Next, we apply Lemma 3 in [11] according to the r-cGA. In the following lemma, we establish the dominance results by comparing two runs of the r-cGA.

Theorem 3. *Let f and g be two r-valued fitness functions to optimize using the r-cGA, such that the first position of f weakly prefers $r-1$ and the first position of g is neutral.*

Let p and q be the corresponding frequency matrices of f and g, both defined by the r-cGA. Then, for all $t \in \mathbb{N}$, it holds that $p_{1,r-1}^{(t)} \succeq q_{1,r-1}^{(t)}$.

Applying Theorem 3 allows us to extend Theorem 2 to positions with weak preference. Since their expected value may raise over time (formally, they are a submartingale), we state the deviation with respect to an arbitrary starting value.

Theorem 4. *Let f be an r-valued fitness function with a weak preference at position $i \in \{1, \ldots, n\}$ for the value $r-1$. Consider the r-cGA optimizing f with parameter K. Let $T \in \mathbb{N}$, then we have*

$$\Pr\left[\min_{t\in\{0,\ldots,T\}} p_{i,r-1}^{(t)} \leq p_{i,r-1}^{(0)} - \frac{1}{2r}\right] \leq 2\exp\left(-\frac{K^2}{8Tr^2}\right).$$

6 Runtime Analysis

This section evaluates the runtime of the r-cGA (Algorithm 1) on r-OneMax. In the preliminaries, we briefly presented the two variants of r-valued OneMax – one is r-OneMax and another is G-OneMax. There is a single local maximum for both functions at the all-$(r-1)$s string, which is also their global optimum. Here, we focus on the runtime evaluation of the r-cGA on r-OneMax only. Further, we leave the analysis of G-OneMax for future work.

With high probability, we bound the runtime of the r-cGA on r-OneMax under the assumption of low genetic drift using drift analysis and then apply Markov's inequality on the time bound. Further we consider the probability of no frequency dropping below $1/(2r)$ at the beginning and, over time, below $k/(2r)$ for a growing k. We prove the following theorem in a similar fashion as Sudholt and Witt [34, Theorem 2] for binary decision variables; however, additional care has to be taken to control genetic drift from the starting value $1/r$ of a frequency. In the following theorem (Theorem 5), we formulate our main results related to runtime.

Theorem 5. *With high probability, the runtime of the r-cGA on r-OneMax with $K \geq cr^2\sqrt{n}(\log r + \log^2 n)$ for a sufficiently large $c > 0$ and $K, r = poly(n)$ is $O(K\sqrt{n}\log r \log n)$. For $K = cr^2\sqrt{n}(\log r + \log^2 n)$, the runtime bound is $O(r^2 n \log^2 r \log^3 n)$.*

The following lemmas are required to prove the Theorem 5. According to the lemmas, the drift grows with an update strength of $1/K$. Furthermore, a high value of $1/K$ also increases genetic drift. We prove the next lemma in a similar way as [27, Lemma 1].

Lemma 2. *Let $p_{i,j}^{(t)}$ denote the frequency vectors of the current iteration of the r-cGA on r-OneMax where $(i,j) \in \{1,\ldots,n\} \times \{0,\ldots,r-1\}$. For a sufficiently large n, we get*

$$\mathrm{P}[D_i = 0] \geq \frac{4}{9\left(2\sqrt{3\left(\sum_{j\neq i} p_{j,r-1}^{(t)}(1-p_{j,r-1}^{(t)})\right)}+1\right)}$$

where $D_i := \left(\sum_{j\neq i} \mathbb{1}\{x_j = r-1\} - \sum_{j\neq i} \mathbb{1}\{y_j = r-1\}\right)$ and $x, y \in \{0,\ldots,r-1\}^n$.

The following lemmas (Lemma 3 and Lemma 4) employ the use of considerations and notation from Lemma 2. In the next lemma, we establish a positive trend towards optimal values for the r-cGA.

Lemma 3. *If* $\frac{1}{K} \leq p^{(t)}_{i,r-1} \leq 1 - \frac{1}{K}$, *then*

$$E(\Delta_{i,r-1} \mid p^{(t)}_{i,r-1}) \geq \frac{8(p^{(t)}_{i,r-1}(1-p^{(t)}_{i,r-1}))}{9K\left(2\sqrt{3\left(\sum_{j\neq i} p^{(t)}_{j,r-1}(1-p^{(t)}_{j,r-1})\right)}+1\right)}.$$

Proof: We get the expected changes using Eq. 1, as

$$E(\Delta_{i,r-1} \mid p^{(t)}_{i,r-1}) = E(F_i \mid p^{(t)}_{i,r-1}) \cdot \mathbb{1}\{R_i\} + E(B_i \mid p^{(t)}_{i,r-1}) \cdot \mathbb{1}\{\overline{R_i}\}.$$

From Sect. 4, we know $E(F_i \mid p^{(t)}_{i,r-1}) = 0$ and $E(B_i \mid p^{(t)}_{i,r-1}) = 2p^{(t)}_{i,r-1}(1 - p^{(t)}_{i,r-1})/K$. Further, from Lemma 2, we got

$$\mathbb{1}\{\overline{R_i}\} \geq \mathrm{P}[D_i = 0] \geq \frac{4}{9\left(2\sqrt{3\left(\sum_{j\neq i} p^{(t)}_{j,r-1}(1-p^{(t)}_{j,r-1})\right)}+1\right)}.$$

By multiplying the results of $E(B_i \mid p^{(t)}_{i,r-1})$ and $\mathrm{P}[D_i = 0]$

$$E(\Delta_{i,r-1} \mid p^{(t)}_{i,r-1}) \geq \frac{8(p^{(t)}_{i,r-1}(1-p^{(t)}_{i,r-1}))}{9K\left(2\sqrt{3\left(\sum_{j\neq i} p^{(t)}_{j,r-1}(1-p^{(t)}_{j,r-1})\right)}+1\right)}. \quad \square$$

The following lemma accumulates the drift of single frequencies in a potential function φ and will be crucial for the proof of Theorem 5. Its proof frequently uses the complementary frequencies $q^{(t+1)}_{i,j} := 1 - p^{(t+1)}_{i,j}$.

Lemma 4. *For any* $t \geq 0$, *let* $\varphi_t := \sum_{i=1}^{n} 1 - p^{(t)}_{i,r-1} = n - \sum_{i=1}^{n} p^{(t)}_{i,r-1}$. *If, for any* $t \geq 0$, *there is some* $s > 0$ *such that for all* $i \in \{1, \ldots, n\}$ *it holds that* $p^{(t)}_{i,r-1} \geq s$ *and furthermore* $\varphi_t \geq 1/2$, *then*

$$E(\varphi_t - \varphi_{t+1} \mid \varphi_t) \geq \frac{2\,s\sqrt{\varphi}}{15K}.$$

Proof. We estimate the expectation of $\varphi' := \varphi_{t+1} = \sum_{i=1}^{n} q^{(t+1)}_{i,r-1}$ based on $\varphi := \varphi_t = \sum_{i=1}^{n} q^{(t+1)}_{i,r-1}$. First, we consider the drift of a single term $q^{(t)}_{i,r-1}$. If $p_{i,r-1} \leq 1 - 1/K$, then by Lemma 3

$$E(q^{(t+1)}_{i,r-1} \mid q^{(t)}_{i,r-1}) \leq q^{(t)}_{i,r-1} - \frac{8(p^{(t)}_{i,r-1}(1-p^{(t)}_{i,r-1}))}{9K\left(2\sqrt{3\left(\sum_{j\neq i} p^{(t)}_{j,r-1}(1-p^{(t)}_{j,r-1})\right)}+1\right)}.$$

We bound $p^{(t)}_{i,r-1}(1-p^{(t)}_{i,r-1})$ from below using our assumption $p^{(t)}_{i,r-1} \geq s$ and the sum from above using

$$\sum_{j\neq i} p^{(t)}_{i,r-1}(1-p^{(t)}_{i,r-1}) \leq \sum_{j=i}^{n}(1-p^{(t)}_{i,r-1}) = \sum_{j=i}^{n} q^{(t)}_{i,r-1} = \varphi.$$

Then,

$$\begin{aligned} E(q^{(t+1)}_{i,r-1} \mid q^{(t)}_{i,r-1}) &\leq q^{(t)}_{i,r-1} - \frac{8}{9}\cdot\frac{q^{(t)}_{i,r-1}}{K}\cdot s\cdot\left(\frac{1}{2\sqrt{3\varphi}+1}\right) \\ &\leq q^{(t)}_{i,r-1}\left(1-\frac{8\,s}{9K}\cdot\frac{1}{2\sqrt{3\varphi}+1}\right). \end{aligned}$$

Putting all together,

$$\begin{aligned} E(\varphi' \mid \varphi) &= \sum_{i=1}^{n} E(q^{(t+1)}_{i,r-1}) \mid q^{(t)}_{i,r-1}) \leq \sum_{i=1}^{n} q^{(t)}_{i,r-1}\left(1-\frac{8\,s}{9rK}\cdot\frac{1}{2\sqrt{3\varphi}+1}\right) \\ &\leq \varphi\left(1-\frac{8\,s}{9K}\cdot\frac{1}{2\sqrt{3\varphi}+1}\right) \leq \varphi - \frac{8\,s}{9K}\cdot\frac{\varphi}{2\sqrt{3\varphi}+1} \\ &\leq \varphi - \frac{8\,s}{9K}\cdot\frac{\varphi^{1/2}}{2\sqrt{3\varphi}+1}\cdot\varphi^{1/2}. \end{aligned}$$

Further, for $\varphi \geq 1/2$, the product of the first two fractions in the negative term can be bounded from below using

$$\frac{8\,s}{9K}\cdot\frac{\varphi^{1/2}}{2\sqrt{3\varphi}+1} \geq \frac{8\,s}{9K}\cdot\frac{\varphi^{1/2}}{2\sqrt{3\varphi}+2\sqrt{3\varphi}} \geq \frac{2\,s}{15K}.$$

By substituting this back,

$$E(\varphi' \mid \varphi) \leq \varphi - \frac{2\,s\varphi^{1/2}}{15K}$$

as suggested. □

With these lemmas, we now provide the proof of the main statement.

Proof (Proof of Theorem 5). By our assumptions on well-behaved frequencies, all frequencies are restricted to $\{0, 1/K, 1/2K, \ldots, 1/r, \ldots, 1-1/K, 1\}$. The main idea is to bound the expected optimization time under the assumption of low genetic drift using additive drift analysis in a sequence of certain phases and then variable drift analysis, using a potential function accumulating all frequencies for value $r-1$. The drift of this potential has already been bounded in Lemma 4 above, using similar estimations as in [34]. The aim is to show that after $\mathrm{O}(K\sqrt{n}\log n \log r)$ iterations the algorithm finds the global optimum, i. e., the string $(r-1)^n$, with high probability, if $K \geq cr^2\sqrt{n}(\log^2 n + \log r)$ for a sufficiently large constant $c > 0$.

Let $p_{i,j}^{(t)}$ denote the marginal probabilities at time t and $q_{i,j}^{(t)} := 1 - p_{i,j}^{(t)}$ where $(i,j) \in \{1,\dots,n\} \times \{0,\dots,r-1\}$. Now, we use the potential function $\varphi_t = \sum_{i=1}^{n} q_{i,r-1}^{(t)}$, which calculates the distance to an ideal setting in which all frequencies for value $r-1$ have reached their maximum. From the definition of $\varphi_t = \sum_{i=1}^{n} 1 - p_{i,r-1}^{(t)} = n - \sum_{i=1}^{n} p_{i,r-1}^{(t)}$, we can note when φ falls, then the sum of the frequencies increases. This will allow us to use increasing bounds for $p_{i,r-1}^{(0)}$ in Theorem 4 as φ falls.

Starting from initialization, we split the run of the r-cGA into phases $k = 1,\dots,r/2$. Phase k starts at the first time where $\varphi_t \le n - kn/r$ and ends just before the first time where $\varphi_t \le n - (k+1)n/r$. We will assume that for each $i \in \{1,\dots,n\}$, at the starting time T_k of phase k it holds that $p_{i,r-1}^{(T_k)} \ge k/(2r)$ and will analyze the probability of the bound holding below when studying genetic drift. Clearly, the bound is true at the starting time $T_1 = 0$ of phase 1.

Note that it is sufficient to reduce the potential by a total amount of n/r to end any phase k. From Lemma 4 we obtain a drift throughout phase k of

$$E(\varphi - \varphi' \mid \varphi) \ge \frac{2\,s\varphi^{1/2}}{15K} \ge \frac{k\sqrt{n/2}}{15Kr}$$

where we use our assumption $s \ge k/(2r)$ and bound $\varphi \ge n/2$ because $k \le r/2$, i. e., we only do the analysis until the potential has decreased to at most $n/2$.

Now, we apply additive drift analysis with overshooting [9, Theorem 4]. An analysis of overshooting is necessary because at the point in time where the potential is at most $n-(k+1)n/r$, it does not have to be exactly $n-(k+1)n/r$ but might be smaller. However, the target cannot be overshoot by much: even if all frequencies change by $1/K$, then the total change is at most n/K. Based on our assumption that $K \ge cr^2(\log^2 n + \log r)$, we can simply pessimistically add the value $n/K \le n/r$ (assuming $c \ge 1$) to the actual distance $d = n/r$ to be overcome. As analyzed above, we have the drift bound $\delta = k\sqrt{n/2}/(15Kr)$ in phase k and the total distance is $(d + n/K)$. Then, the expected time to bridge the distance is $(d+n/K)/\delta \le (2n/r)/\delta = 2d/\delta$. So, we obtain the expected time to conclude phase k is at most

$$\frac{2d}{\delta} = \frac{2n}{r}\frac{15K}{\sqrt{n/2}}\frac{r}{k} = \mathrm{O}(K\sqrt{n}/k).$$

Applying Markov's inequality and a restart argument, the length of phase k is $\mathrm{O}(K\sqrt{n}\log n/k)$ with probability $1 - \mathrm{O}(1/n^{\kappa})$ for any constant $\kappa > 0$. Further, summing over $k = 1,\dots,r/2$, we obtain the total expected time spent in all phases is

$$\mathrm{O}\left(\sum_{k=1}^{r} \frac{K\sqrt{n}}{k}\right) = \mathrm{O}\left(K\sqrt{n}\sum_{k=1}^{r}\frac{1}{k}\right) = \mathrm{O}(K\sqrt{n}\log r).$$

In the same way, by adding up the tail bounds on the phase lengths, we have that the total time spent in phases $1,\dots,r/2$ is $\mathrm{O}(K\sqrt{n}\log r \log n)$ with probability at

least $1-\mathcal{O}(rn^{-\kappa})$, using a union bound. Since $r = poly(n)$, this failure probability is still $o(1)$ if κ is a sufficiently large constant.

After the potential has decreased to at most $n/2$, we can essentially use the same analysis as for the classical (binary) cGA on OneMax [34, Theorem 2], with the exception that we do not use borders on frequencies here and, therefore, the potential φ is non-increasing. We have $\varphi \leq n/2$ as starting point and assume $s \geq 1/2 - 1/(2r) \geq 1/4$. Using the variable drift theorem [34, Theorem 18], we can estimate the expected time it takes for the potential function to drop from $\varphi \leq n/2$ to a value $\varphi \leq 1/2$ (we do not wait for the potential to equal 0 because this would lead to weaker runtime bound). We choose the drift function $h(\varphi) \coloneqq 2s\varphi^{1/2}/(15K) \geq \varphi^{1/2}/(30K)$, for any $\varphi \geq 1/2$, in the variable drift theorem. Also similarly as in [34, Proof of Theorem 2], we merge all the states with potentials $0 < \varphi < 1/2$ with state 0 so that the smallest state larger than 0 is $x_{\min} = 1/2$. This modification can only increase the drift, hence the drift is still bounded from below by $h(\varphi)$ for all states $\varphi \geq x_{\min}$. Moreover, at $x_{\min} = 1/2$, the probability of sampling the optimum is at least $1/2$ by the same arguments related to majorization and Schur-convexity as in [34].

Now, the expected time to reach state 0 in the updated process, or any state with $\varphi < 1/2$ in the actual process, is at most

$$\frac{1/2}{h(1/2)} + \int_{1/2}^{n} \frac{1}{h(\varphi)}\, d\varphi = \mathrm{O}(K) + \mathrm{O}(K) \cdot \int_{1/2}^{n} \varphi^{-1/2}\, d\varphi = \mathrm{O}(K\sqrt{n}).$$

Afterwards, after expected time at most 2, the optimum is sampled. Moreover, again by applying Markov's inequality and a restart argument, the time to sample the optimum is $\mathrm{O}(K\sqrt{n}\log n)$ with probability $1 - o(1)$.

We still have to analyze the effect of genetic drift. In the analysis above, we assume that at the time T_k beginning phase k (when the potential has become at most $n - nk/r$), every frequency $p_{i,r-1}^{(T_k)}$, where $i \in \{1, \dots, n\}$, is bounded from below by $k/(2r)$. By definition, a potential of φ_t corresponds to an average frequency value of $(n - \varphi_t)/n$. Moreover, since all n frequencies $p_{i,r-1}^{(t)}$ describe the same stochastic process due to the symmetry of the r-OneMax function, this average frequency equals the expected frequency at this time t. Deviations of the frequency below the expected value can only happen in rw-steps. The aim is therefore to show that in each phase, the reduction of any frequency in rw-steps is bounded from above by $1/(2r)$. By Theorem 4 (using identifying its starting time 0 with T_k), the probability of a failure in a single phase, i. e., of a reduction more than $1/(2r)$ by genetic drift is at most $2\exp(K^2/(2Tr^2))$. Using $K \geq cr^2\sqrt{n}(\log^2 n + \log r)$ for some sufficiently large constant $c > 0$ and plugging in the above bound on the length of phase k of $T = c'(K\sqrt{n}\log n)/k$ for another constant $c' > 0$, the failure probability is bounded from above by

$$2\exp(Kk/(2c'\sqrt{n}(\log n)r^2)) \leq 2\exp(kc(\log n + \log r)/c') \leq 2n^{-c/c'}r^{-c/c'}.$$

Choosing c large enough and taking a union bound over at most r phases and n frequencies, the failure probability in this analysis of genetic drift is still $o(1)$.

By choosing all constants appropriately, also the sum of all failure probabilities is $o(1)$. □

7 Experiments

In the following section, we show the results of the experiments carried out to check the performance of the proposed algorithm without border restriction. Theoretically, we prove the typical runtime for the r-cGA on the r-OneMax without margins. We used the C programming language and the WELL1024a random number generator to implement the algorithm.

In the experiments, we ran the r-cGA on r-OneMax for $n = 500$ (Fig. 1) without border restriction and all averaged over 3000 runs where $r \in \{3, 4, \ldots, 10\}$. Also, for G-OneMax using $G\text{-OneMax}(x) = \sum_{i=1}^{n} x_i$ for simplicity, we ran the r-cGA with the same configuration. In all cases, we observe the same scenario where the empirical runtime begins at a very high value, takes a minimum and then increases again for the rest of the K. As an example, we got the minimum when K is around 110 (Fig. 1: Left-hand side and $r = 4$). And after that it clearly goes up with K.

We can compare the results according to the variants. If we compare the plots for r-OneMax and G-OneMax, then both variants produce the same structure. From the experimental setup, we can say that the bound of the theoretical analysis is not tight. We observed some empirical findings about the relationship between K and runtime. After the close inspection, we can find the value of K where the minimum of the runtime is reached. Based on the experiments, we believe that the runtime of the r-cGA on G-OneMax is higher than on r-OneMax. In fact, the analysis of G-OneMax is more complicated compared to r-OneMax since there is no benefit in moving to a value close to the optimal one if the new value is not optimal. Therefore, the theoretical analysis of the r-cGA on G-OneMax is one of our future research subjects.

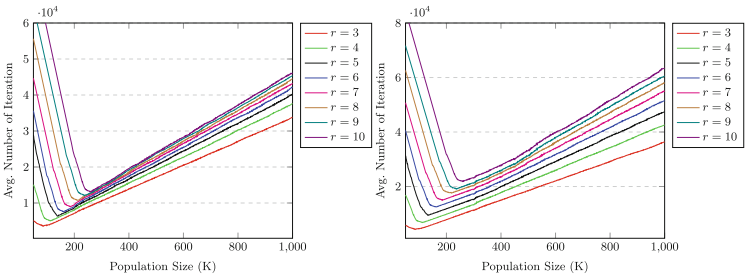


Fig. 1. Left-hand side: empirical runtime of the r-cGA on r-OneMax, right-hand side: empirical runtime of the r-cGA on G-OneMax; for $n = 500$, $K \in \{50, 51, \ldots, 1000\}$ and averaged over 3000 runs.

8 Conclusion

We have performed a runtime analysis of a multi-valued EDA, namely the r-cGA, on a generalized OneMax function. In our analysis, we have bound the runtime of the r-cGA with a high probability. Also, considering the increased complexity of the problem, the resulting runtime is understandable. Since the r-cGA is efficient on generalized OneMax, we believe that the r-cGA is a good algorithm for other, more complex problems, too.

A theoretical analysis of the r-cGA on G-OneMax is one of the future research works. Also, we would like to investigate the r-cGA on a multi-valued OneMax problem where all the frequencies are restricted by a specific upper and lower border value. Based on the experiments, we believe that the runtime of the r-cGA on G-OneMax is higher than on r-OneMax. Further, from the experiments, we believe that our runtime bounds can be improved.

Acknowledgments. This work has been supported by the Danish Council for Independent Research through grant 10.46540/2032-00101B.

References

1. Adak, S., Witt, C.: Runtime analysis of a multi-valued compact genetic algorithm on generalized OneMax (2024). https://arxiv.org/abs/2404.11239
2. Asoh, H., Mühlenbein, H.: On the mean convergence time of evolutionary algorithms without selection and mutation. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) Parallel Problem Solving from Nature — PPSN III, pp. 88–97. Springer, Berlin, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_253
3. Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Carnegie Mellon University Pittsburgh, PA, School of Computer Science (1994)
4. Ben Jedidia, F., Doerr, B., Krejca, M.S.: Estimation-of-distribution algorithms for multi-valued decision variables. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 230–238 (2023)
5. Benbaki, R., Benomar, Z., Doerr, B.: A rigorous runtime analysis of the 2-MMASIB on jump functions: ant colony optimizers can cope well with local optima. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 4–13 (2021)
6. Dang, D.C., Lehre, P.K.: Simplified runtime analysis of estimation of distribution algorithms. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 513–518 (2015)
7. De Bonet, J., Isbell, C., Viola, P.: MIMIC: finding optima by estimating probability densities. In: Advances in Neural Information Processing Systems **9** (1996)
8. Doerr, B.: The runtime of the compact genetic algorithm on jump functions. Algorithmica **83**, 3059–3107 (2021)
9. Doerr, B., Kötzing, T.: Multiplicative up-drift. Algorithmica **83**(10), 3017–3058 (2021)
10. Doerr, B., Krejca, M.S.: The univariate marginal distribution algorithm copes well with deception and epistasis. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, pp. 17–18 (2020)

11. Doerr, B., Zheng, W.: Sharp bounds for genetic drift in estimation of distribution algorithms. IEEE Trans. Evol. Comput. **24**(6), 1140–1149 (2020)
12. Droste, S.: Not all linear functions are equally difficult for the compact genetic algorithm. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 679–686 (2005)
13. Droste, S.: A rigorous analysis of the compact genetic algorithm for linear functions. Nat. Comput. **5**, 257–283 (2006)
14. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoret. Comput. Sci. **276**(1), 51–81 (2002)
15. Friedrich, T., Kötzing, T., Krejca, M.S.: EDAs cannot be balanced and stable. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1139–1146 (2016)
16. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Trans. Evol. Comput. **3**(4), 287–297 (1999)
17. Harik, G.R., Lobo, F.G., Sastry, K.: Linkage learning via probabilistic modeling in the extended compact genetic algorithm (eCGA). In: Scalable Optimization via Probabilistic Modeling, pp. 39–61. Springer (2006). https://doi.org/10.1007/978-3-540-34954-9_3
18. Hasenöhrl, V., Sutton, A.M.: On the runtime dynamics of the compact genetic algorithm on jump functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 967–974 (2018)
19. Kimura, M.: Diffusion models in population genetics. J. Appl. Probab. **1**(2), 177–232 (1964)
20. Krejca, M.S., Witt, C.: Lower bounds on the run time of the univariate marginal distribution algorithm on OneMax. In: Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, pp. 65–79 (2017)
21. Krejca, M.S., Witt, C.: Theory of estimation-of-distribution algorithms. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 405–442. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_9
22. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, vol. 2. Springer Science & Business Media (2001). https://doi.org/10.1007/978-1-4615-1539-5
23. Lengler, J., Sudholt, D., Witt, C.: Medium step sizes are harmful for the compact genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1499–1506 (2018)
24. Lengler, J., Sudholt, D., Witt, C.: The complex parameter landscape of the compact genetic algorithm. Algorithmica **83**, 1096–1137 (2021)
25. McDiarmid, C.: Concentration. In: Habib, M., McDiarmid, C., Ramirez-Alfonsin, J., Reed, B. (eds.) Probabilistic Methods for Algorithmic Discrete Mathematics, pp. 195–248. Springer, Berlin, Heidelberg (1998). https://doi.org/10.1007/978-3-662-12788-9_6
26. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) Parallel Problem Solving from Nature — PPSN IV, pp. 178–187. Springer, Berlin, Heidelberg (1996). https://doi.org/10.1007/3-540-61723-X_982
27. Neumann, F., Sudholt, D., Witt, C.: A few ants are enough: ACO with iteration-best update. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 63–70 (2010)

28. Pelikan, M., Hauschild, M.W., Lobo, F.G.: Estimation of Distribution Algorithms. In: Kacprzyk, J., Pedrycz, W. (eds.) Springer Handbook of Computational Intelligence, pp. 899–928. Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-43505-2_45
29. Pelikan, M., Muehlenbein, H.: The bivariate marginal distribution algorithm. In: Roy, R., Furuhashi, T., Chawdhry, P.K. (eds.) Advances in Soft Computing, pp. 521–535. Springer, London (1999). https://doi.org/10.1007/978-1-4471-0819-1_39
30. Santana, R., Larranaga, P., Lozano, J.A.: Protein folding in simplified models with estimation of distribution algorithms. IEEE Trans. Evol. Comput. **12**(4), 418–438 (2008). https://doi.org/10.1109/TEVC.2007.906095
31. Shapiro, J.L.: The sensitivity of PBIL to its learning rate, and how detailed balance can remove it. In: FOGA, pp. 115–132 (2002)
32. Shapiro, J.L.: Drift and scaling in estimation of distribution algorithms. Evol. Comput. **13**(1), 99–123 (2005)
33. Shapiro, J.L.: Diversity loss in general estimation of distribution algorithms. In: Runarsson, T.P., Beyer, H.-G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) Parallel Problem Solving from Nature - PPSN IX: 9th International Conference, Reykjavik, Iceland, September 9-13, 2006, Proceedings, pp. 92–101. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11844297_10
34. Sudholt, D., Witt, C.: On the choice of the update strength in estimation-of-distribution algorithms and ant colony optimization. Algorithmica **81**, 1450–1489 (2019)
35. Witt, C.: Domino convergence: why one should hill-climb on linear functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1539–1546 (2018)
36. Witt, C.: Upper bounds on the running time of the univariate marginal distribution algorithm on OneMax. Algorithmica **81**, 632–667 (2019)
37. Witt, C.: How majority-vote crossover and estimation-of-distribution algorithms cope with fitness valleys. Theoret. Comput. Sci. **940**, 18–42 (2023)

Faster Optimization Through Genetic Drift

Cella Florescu, Marc Kaufmann(✉), Johannes Lengler, and Ulysse Schaller

Department of Computer Science, ETH Zürich, Zürich, Switzerland
{cella.florescu,marc.kaufmann,johannes.lengler,
ulysse.schaller}@inf.ethz.ch

Abstract. The compact Genetic Algorithm (cGA), parameterized by its hypothetical population size K, offers a low-memory alternative to evolving a large offspring population of solutions. It evolves a probability distribution, biasing it towards promising samples. For the classical benchmark OneMax, the cGA has two different modes of operation: a conservative one with small step sizes $\Theta(1/(\sqrt{n}\log n))$, which is slow but prevents genetic drift, and an aggressive one with large step sizes $\Theta(1/\log n)$, in which genetic drift leads to wrong decisions, but those are corrected efficiently. On OneMax, an easy hill-climbing problem, both modes lead to optimization times of $\Theta(n \log n)$ and are thus equally efficient.

In this paper we study how both regimes change when we replace OneMax by the harder hill-climbing problem Dynamic BinVal. It turns out that the aggressive mode is not affected and still yields quasi-linear runtime $O(n \operatorname{polylog} n)$. However, the conservative mode becomes substantially slower, yielding a runtime of $\Omega(n^2)$, since genetic drift can only be avoided with smaller step sizes of $O(1/n)$. We complement our theoretical results with simulations.

Keywords: compact Genetic Algorithm · Genetic Drift · Estimation-of-Distribution Algorithm · Dynamic Binary Value

1 Introduction

Estimation-of-distribution algorithms (EDAs) are a family of randomized optimization heuristics in which the algorithm evolves a probability distribution over the search space.[1] In each iteration, it samples solutions from this distribution, evaluates their quality (also called fitness), and updates the probability distribution accordingly. Examples in discrete domains include the cGA, UMDA, PBIL, ant colony systems like the MMAS, and multivariate systems like hBOA [24] or MIMIC [2], see [16] for a survey. EDAs turn out to be a powerful alternative to population-based heuristics like evolutionary and genetic algorithms. They have the advantage that they often sample from a wider region

[1] Proofs in this submission are omitted due to the page limit. A full version with detailed proofs can be found in the arXiv version of this article [10].

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 70–85, 2024.
https://doi.org/10.1007/978-3-031-70071-2_5

of the search space than their population-based alternatives, which makes them less susceptible to local deviations like (frozen or non-frozen) noise and local optima [3,11,12,18,27].

EDAs have been used for several decades, but theoretical investigations of EDAs have only started to gain momentum a few years ago. While for some aspects a clear picture has emerged, like that EDAs are able to cope with large amounts of noise [11], there is one aspect for which researchers have found a complex and ambiguous pattern: genetic drift. Genetic drift is the tendency of an algorithm to move through the fitness landscape even in absence of a clear signal-to-noise ratio.[2] While it is possible to avoid genetic drift by tracking the signal-to-noise ratio [5,8], this conservative attempt of avoiding mistakes could potentially make the algorithm slow and inflexible. An alternative approach might be to embrace genetic drift, allow the algorithm to swiftly move through the search space, and let it correct mistakes as they appear.

Indeed, these two alternatives are exemplified by the behaviour of the compact Genetic Algorithm cGA on the pseudo-Boolean function OneMax [23]. The OneMax function assigns to a bit string $x \in \{0,1\}^n$ the number of one-bits in x. It is one of the simplest and most classical hill-climbing benchmarks. The cGA maintains for each of the n coordinates a *frequency* p_i, which encodes the probability that the i-th bit is set to one in the distribution. In each iteration, it samples two solutions, and for each component i it shifts the frequency p_i by $1/K$ towards the value of the fitter of the two solutions. Droste was the first to prove an expected runtime bound of $\Theta(\sqrt{n}K)$ on OneMax and $\Theta(Kn)$ on BinVal whenever $K = \Omega(n^{1/2+\varepsilon})$ [9]. The latter assumption on K was necessary in order to avoid frequencies getting trapped at 0 or at 1. In the version of the algorithm that recent research has therefore focused on, frequencies are capped in the interval $[\bar{p}, 1-\bar{p}]$ in order to avoid that, where $\bar{p} = \bar{p}(n) \in (0, \frac{1}{2})$ is a boundary parameter, usually set to $\bar{p} = \frac{1}{n}$, see Sect. 2 for full details. The step size $1/K$ determines how aggressively or conservatively the algorithm updates. It is well understood that the size of K determines whether genetic drift happens to a relevant extent on OneMax or not. If $K = \omega(\sqrt{n}\log n)$ then the frequencies move so slowly that the signal exceeds the noise, and all frequencies move slowly but steadily towards the upper boundary. This corresponds to the regime where genetic drift is avoided, and we refer to this as *conservative* regime. On the other hand, if $K = o(\sqrt{n}\log n)$ then the signal is weaker than the noise, and some bits move to the wrong boundary due to genetic drift. In the subsequent optimization process, these mistakes are then slowly corrected. We call this the *aggressive* regime.

It turns out that both regimes are equally efficient on OneMax. For suitable $K = C\log n$ with a large constant C, errors are corrected so quickly that the opti-

[2] The term *drift* is also used in the context of *drift analysis*, where it means the expected change, which is almost the opposite concept. The term "genetic drift" should not be confused with this other meaning of the term "drift".

mum is sampled in $O(n \log n)$ iterations.[3] On the other hand, if $K = C\sqrt{n} \log n$ with a large constant C, then the algorithm moves more slowly, but does not make any errors, which yields the same asymptotic runtime $O(n \log n)$. Both parameter settings are brittle with respect to smaller K: if either $K = C \log n$ or $K = C\sqrt{n} \log n$ are decreased only slightly, this results in a sudden loss of performance. On the other hand, if the parameter K is increased from either $K = C \log n$ or $K = C\sqrt{n} \log n$, then the performance deteriorates slowly but steadily. Hence, there are two optimal parameter settings for the cGA on OneMax, a conservative one which avoids genetic drift and an aggressive one which embraces genetic drift.

Since the mentioned analysis was limited to OneMax, it remained open whether both modes of the algorithm also show comparable performance for other hill-climbing tasks. In this paper, we give a negative answer and show that for the harder hill-climbing problem Dynamic BinVal, the aggressive mode still finds the optimum in quasi-linear time, while the conservative mode needs time $\Omega(n^2)$.

1.1 Our Results

We investigate the cGA on the function Dynamic BinVal, or DynBV for short. This function, introduced in [20], builds on the classical linear test function Binary Value that assigns to each binary string the integer that is represented by it in the binary number system. DynBV is obtained by drawing at each iteration a random permutation of the weights and then evaluating all solutions with the permuted Binary Value function, see Sect. 2 for a formal definition. Binary Value is conjectured to be the hardest linear function[4], and DynBV is known to be the hardest dynamic linear function[5] [20,21]. This makes it the perfect benchmark for a hard hill-climbing task. For more discussion of the benchmark, see Sect. 1.2.

Our proofs rely heavily on drift theorems, a standard toolbox from the analysis of evolutionary algorithms. These allow to transform statements about the expected one-iteration change of a potential function, a proxy for the function to be optimized, into runtime bounds. For an overview see [19].

The Conservative Regime is Slow. Our first main result is the following lower runtime bound, which holds for all $K = O(\text{poly}(n))$. In this range the runtime will at first increase quadratically in K, until K reaches the dimension of the search space - at which point the runtime dependency becomes $\Omega(K \cdot n)$.

[3] This was only shown formally for the UMDA in [27] and [1], not for the cGA. However, [23] contains an informal argument why the results should also apply to the cGA.

[4] For the $(1 + 1)$-EA. It is a famous open problem to prove this formally [13].

[5] It is not formally a dynamic linear function in the sense of [22], but can be obtained as a limit of such functions [20].

Theorem 1. *Let $\bar{p} \in (0, \frac{1}{2})$ be arbitrary and consider the* cGA *with parameter $K = O(\text{poly}(n))$ and boundaries at $\bar{p}$ and $1 - \bar{p}$ on* DynBV. *Then with high probability, the optimum is not sampled during the first $\Omega(K \cdot \min\{K, n\})$ iterations.*

The reason for this is captured in Lemma 9, which states that in this period, there are always linearly many bits which stay in some constant interval around their initialization value, so it is exponentially unlikely to sample the optimum.

If we want to avoid genetic drift, we have to choose a rather large K (small step sizes) to overcome the small signal-to-noise ratio that is inherent to DynBV. The following theorem states that any $K = O(n)$ will lead to substantial genetic drift and hence belongs to the aggressive regime. This agrees with the guidelines from [3] on how to avoid genetic drift and is similar to LeadingOnes [6].

Theorem 2. *For every $\rho > 0$ and $\beta \in (\bar{p}, \frac{1}{2})$ there is $\delta > 0$ such that the following holds. Consider the* cGA *with parameter $K \leq \rho n$ on* DynBV. *Then with high probability as $K \to \infty$ at least δn frequencies drop below β during optimization.*

The proof sketches of Theorem 1 and Theorem 2 can be found in Sect. 4. Theorem 2 shows that indeed the only possibility to avoid substantial genetic drift is to set $K = \omega(n)$, which leads to a runtime of $\omega(n^2)$ by Theorem 1. Hence, DynBV can not be optimized in quadratic time by any parameter setting of the cGA that avoids genetic drift. As we will see below, aggressive parameter settings that allow genetic drift are much more efficient. Before we come to this other regime, we complement Theorem 1 with a matching upper bound that holds when we are safely in the conservative regime with $K = \Omega(n \log n)$. To simplify the proof, we require a slight adjustment of the boundary values. More precisely, we set them to $\frac{1}{cn}, 1 - \frac{1}{cn}$ for a large enough constant $c > 0$. As our simulations, which are all conducted with boundaries $\frac{1}{n}, 1 - \frac{1}{n}$, show, this choice does not affect the asymptotic behaviour.

Theorem 3. *Consider the* cGA *with parameter $K = \text{poly}(n)$ and boundaries at $\frac{1}{cn}$ and $1 - \frac{1}{cn}$ on* DynBV. *If $K \geq c' \cdot n \log n$, and the constants $c, c' > 0$ are large enough, then the expected optimization time is $O(Kn)$.*

The main ingredient for this proof, whose sketch can be found in Sect. 5, is to show that in the first polynomially many rounds, all frequencies stay bounded away from the lower boundary (Proposition 10). Hence, the proof of Theorem 3 is similar to other proofs of upper runtime bounds in conservative regimes [8,26].

Together, Theorems 1 and 3 give tight runtime bounds of $\Theta(Kn)$ in the conservative regime. This implies in particular that the runtime in this regime is much larger for DynBV than for OneMax, where the runtime is $O(K\sqrt{n})$ [25].

We remark that the different transition points between conservative and aggressive regime ($K = \Theta(n)$ in Theorem 2 and $K = \Theta(n \log n)$ in Theorem 3) are natural because there are different possible definitions of the conservative regime: that no frequency drops below 1/3 (or any other fixed constant below

1/2), that no frequency reaches the lower boundary, or that the number of frequencies hitting the lower boundary is sublinear. All these variants lead to different transition points between the conservative and aggressive regime.

The Aggressive Regime is Fast. Our second result shows that in contrast, the optimization time of the cGA remains quasi-linear for small K, i.e. linear up to a poly-logarithmic factor. This corresponds to the aggressive regime where many frequencies reach the wrong boundary, but those errors are corrected efficiently. To make the analysis simpler, similar as for Theorem 3, we do not set the two boundary values at their standard values $1/n$ and $1 - 1/n$, but this time we even set them to $1/(n \operatorname{polylog} n)$ and $1 - 1/(n \operatorname{polylog} n)$. Moreover, we do not use the smallest possible (most aggressive) parameter choice $K = C \log n$ for the aggressive regime, but rather choose the slightly more conservative $K = \Theta(\log^2 n)$. Then we prove the following result (proof sketch in Sect. 6).

Theorem 4. *Consider the* cGA *with parameter* $K = \Theta(\log^2 n)$ *and boundaries* $1/(n \log^7 n)$ *and* $1 - 1/(n \log^7 n)$ *on* DynBV. *Then with high probability the optimum is sampled in* $O(n \cdot \operatorname{polylog}(n))$ *iterations.*

We hide in the notation $O(n \cdot \operatorname{polylog}(n))$ our explicit derived bound of $O(n \log^{16} n)$ iterations. But this is not tight with regard to log-factors in various places, so we did not optimize for the exponent of the logarithm. We further note that we made no effort to optimize the exponent 7 of the poly-logarithmic factor in the boundaries. We conjecture that the true runtime for optimal parameters is $O(n \log n)$, and that this is achieved with the standard boundaries $1/n$ and $1 - 1/n$. Notably, this would mean that there is no substantial runtime difference in the aggressive regime with optimal parameters between OneMax and DynBV, in stark contrast to the conservative regime. We do not quite show this statement, but we show it up to poly-logarithmic factors.

Compared to this conjecture, our analysis is likely not tight in several ways. Firstly, even for the given parameters we believe that the poly-logarithmic exponent of our runtime bound could be reduced at the cost of a more technical analysis. Secondly, both the conservative choice of K and the non-standard choice of the threshold likely bring us away from the optimal parameter. This simplifies the proof, but costs us performance, even though only logarithmic factors. We suspect that the optimal parameter setup is indeed the standard setup of $K = C \log n$ for a large constant C and boundaries $1/n$ and $1 - 1/n$. This is supported by the experiments presented in Sect. 7.

1.2 Discussion of the Setup and Related Work

Signal Steps and DynBV. In order to understand genetic drift, a key question is how often each frequency receives a signal step. We call an iteration a *signal step* for frequency p_i if both solutions differ in this position i and the two values of this

position are necessary to decide which of the two solution is fitter. When both solutions differ, but their values are irrelevant for identifying the fitter solution, then we call the iteration a *random walk step* for frequency p_i. In the initial phase of the cGA on OneMax, the probability of a signal step is $\Theta(1/\sqrt{n})$. For DynBV, the signal probability is considerably weaker, namely of order $\Theta(1/n)$. In fact, in each iteration exactly one frequency receives a signal step, except when the two offspring sampled by the algorithm agree in every single bit.

As mentioned, DynBV is the hardest dynamic linear function. This also holds in terms of the signal strength: when comparing two non-equal solutions, then exactly one frequency gets a signal, while all other frequencies perform a random walk step. This is the weakest signal strength among all dynamic linear functions, and even the hardest among all dynamic monotone functions [14], since every monotone function, static or dynamic, will always provide a signal step to at least one frequency when comparing two solutions.

Although it is a dynamic function, DynBV provides a hill-climbing task in the sense that in each iteration and at any position, a one-bit gives a higher fitness than a zero-bit. Thus, pure hill-climbing heuristics such as Random Local Search (RLS) can be highly efficient on this function. Moreover, DynBV is more symmetric than the classical Binary Value function, which makes the analysis simpler. All these properties make DynBV the perfect benchmark for a theoretical runtime analysis of a hard hill-climbing task.

Related Work. It has been shown that Dynamic BinVal is harder to optimize by evolutionary algorithms than static monotone functions in various ways. The $(1, \lambda)$-EA with self-adapting offspring population size fails on Dynamic BinVal while succeeding on OneMax if the hyperparameters are not set correctly [15]. Furthermore, a "switching" variant of Dynamic BinVal minimizes drift in the number of zeros at every search point for the $(1 + 1)$-EA for any mutation rate at every search point, making it harder to optimize than any *static* monotone function [14].

We do not claim that the aggressive mode of the cGA is generally superior to the conservative mode. However, our results show that the other extreme position of avoiding genetic drift at all costs, does cost performance for Dynamic BinVal. On the other hand, the conservative mode was shown to be superior on the function DeceptiveLeadingBlocks for some parameter settings [7,17], though a discussion at a Dagstuhl seminar shows that opinions are split about the implications of these results [4]. We hope that further research will give a clearer and more nuanced picture on the benefits and drawbacks of genetic drift.

2 Setting

Our search space is always $\{0,1\}^n$. We say that an event $\mathcal{E} = \mathcal{E}(n)$ holds *with high probability* or *whp* if $\Pr[\mathcal{E}] \to 1$ as $n \to \infty$. We may for simplicity omit the parameter t indicating the iteration when it is clear from context.

2.1 The Algorithm: The cGA with Hypothetical Population Size K

We begin with an intuitive description of the cGA. Before the start of the algorithm, we fix a capping probability $\bar{p} < \frac{1}{2}$ and a hypothetical population size K, which we may think of as an inverse update strength. At every iteration, the algorithm generates two offspring x and y independently of each other by the same sampling procedure. At iteration t, the i-th bit of the offspring to be sampled is set (independently of all other bits and of all previous iterations) to 1 with probability $p_{i,t}$ and set to 0 otherwise. The probabilities are initialized to $\frac{1}{2}$ for all bits and evolve according to the following procedure: At each iteration, the fitness of x and y are compared according to the fitness function f - in our case this will be Dynamic BinVal. If the fitter offspring contains a 1 at position i, we increase the probability of sampling a 1 bit, p_i, by $\frac{1}{K}$ for the next iteration, otherwise we decrease it by the same amount. If there is no strictly fitter offspring, i.e. $f(x) = f(y)$, all probabilities $p_{i,t}$ remain unchanged in the next iteration. To ensure the algorithm does not get stuck by fixing one of the bits, i.e. sampling a 1 (or a 0) with probability 1, we restrict the possible values for probabilities $p_{i,t}$ to the interval $[\bar{p}, 1 - \bar{p}]$. If an update step would make a $p_{i,t}$ exceed these bounds, we set it to the boundary value instead. The algorithm stops when the optimum has been sampled (as one of the two offspring in a given iteration). The pseudocode is provided in Algorithm 1.

2.2 The Benchmark: Dynamic BinVal

In our considered benchmark Dynamic BinVal, at each iteration t, we draw uniformly at random (and independently of everything else) from the set of bijections from $\{1, \dots, n\}$ onto itself an element $\pi_t \colon \{1, 2, \dots, n\} \to \{1, 2, \dots, n\}$. Note that this can be seen as a permutation of the bits of the search point. The fitness function for iteration t is then given by

$$f_t(x) = \sum_{i=1}^{n} 2^{n-i} \cdot x_{\pi_t(i)}.$$

Intuitively, the offspring which has a 1 bit at the most significant position (given by the permutation π_t) at which the two offspring differ is considered fitter.

2.3 Terminology

Signal Step. A signal step of a bit $i \in \{1, 2, \dots, n\}$ is the increase in the marginal probability of bit i during an iteration t where the value of this bit in the two offspring was decisive. In other words, bit i performs a signal step at iteration t if and only if the two offspring differ at bit i and are equal at any other bit i' for which $\pi_t(i') < \pi_t(i)$.

Random Step. A random step of a bit $i \in \{1, 2, \dots, n\}$ is the change in the marginal probability of bit i during an iteration where the value of this bit in the two offspring was not decisive. Thus, all changes in a bit's marginal probability that are not signal steps are random steps.

Algorithm 1. cGA$(f, K, \bar{p})$

$t \leftarrow 0$
$p_{1,t} \leftarrow p_{2,t} \leftarrow \cdots \leftarrow p_{n,t} \leftarrow \frac{1}{2}$
while optimum has not been sampled **do**
  for $i \in \{1, 2, \ldots, n\}$ **do**
    $x_i \leftarrow 1$ with probability $p_{i,t}$ and 0 otherwise
    $y_i \leftarrow 1$ with probability $p_{i,t}$ and 0 otherwise
  end for
  if $f(x) = f(y)$ **then**
    $t \leftarrow t + 1$
    continue
  else if $f(x) < f(y)$ **then**
    swap x and y
  end if
  for $i \in \{1, 2, \ldots, n\}$ **do**
    if $x_i > y_i$ **then**
      $p'_{i,t+1} \leftarrow p_{i,t} + \frac{1}{K}$
    else if $x_i < y_i$ **then**
      $p'_{i,t+1} \leftarrow p_{i,t} - \frac{1}{K}$
    else
      $p'_{i,t+1} \leftarrow p_{i,t}$
    end if
    $p_{i,t+1} \leftarrow \min\left\{\max\left\{\bar{p}, p'_{i,t+1}\right\}, 1 - \bar{p}\right\}$
  end for
  $t \leftarrow t + 1$
end while

Sampling Variance. The sampling variance at time t is the variance of the binomial distribution induced by the probabilities $p_{i,t}, i = 1, \ldots n$. We denote it by $V_t := \sum_{i=1}^{n} p_{i,t}(1 - p_{i,t})$. Intuitively, it is the sum at time t of the variances contributed by the probability of each frequency in the generating distribution.

Lower/Upper Boundary. Recall from Subsect. 2.1 that the possible values for the probabilities $p_{i,t}$ are restricted to an interval $[\bar{p}, 1 - \bar{p}] \subsetneq [0, 1]$ for some value $\bar{p} = O(\frac{1}{n})$ to ensure that the algorithm does not get stuck. The values $\bar{p}$ and $1 - \bar{p}$ are going to be referred to as lower and upper boundary respectively. For example, we say a frequency $p_{i,t}$ is at its upper boundary if $p_{i,t} = 1 - \bar{p}$. Note that the distances of the lower and upper boundaries from 0 and 1 is always the same, respectively. $\bar{p}$ is fixed for the entire execution of the algorithm. It will always either be $\frac{1}{cn}$ or $\frac{1}{n \log^c n}$ for constant $c > 0$, and it will usually be clear from context which value of $\bar{p}$ is assumed.

3 Dynamics of the Marginal Probabilities

We start by analyzing how the cGA behaves on the Dynamic BinVal at the level of a single iteration. The first proposition computes the probability that

a bit i at which the two offspring differ gets a signal step (in some iteration t). This proposition captures the main difference between DynBV and OneMax. For OneMax, a position which differs in the two offspring has probability $1/\max\{\sqrt{V_t}, 1\}$ to perform a signal step [26], and performs a random step otherwise. For DynBV, the probability is $1/\max\{V_t, 1\}$, so the term $\sqrt{V_t}$ is replaced by V_t. Hence, signal steps are more likely for OneMax, and the signal-to-noise ratio of OneMax is larger than for DynBV. This corresponds to the fact that OneMax is a particularly easy function to optimize.

Proposition 5. *Let $K \geq 1$ and $\bar{p} \in (0, \frac{1}{2})$ be arbitrary, and consider the algorithm* cGA$(K, \bar{p})$ *on* DynBV *and some bit $i \in [n]$ and iteration t. For the permutation π_t drawn at iteration t, we denote by $S_{i,t}$ the event that all bits $i' \in [n]$ that appear before i in the permutation, i.e. such that $\pi_t(i') < \pi_t(i)$, are equal in the two offspring. Then it holds that*

$$\Pr[S_{i,t}] = \Theta\left(\frac{1}{\max\{V_t, 1\}}\right).$$

The idea of the proof is that the expected number of bits differing in the two offspring is $\sum_{i=1}^{n} 2p_{i,t}(1-p_{i,t}) = 2V_t$, and each of these bits is equally likely to be the first bit differing in the two offspring, yielding a probability of $\Theta(1/\max\{V_t, 1\})$ (the maximum with 1 ensures the expression stays constant).

Using Proposition 5, one can describe the transition matrix of the marginal probabilities, which then allows to compute the drift of these marginal probabilities. As for Proposition 5, the same formulas would hold for OneMax with V_t replaced by $\sqrt{V_t}$.

Proposition 6. *Let $K \geq 1$ and $\bar{p} \in (0, \frac{1}{2})$ be arbitrary, and consider the algorithm* cGA$(K, \bar{p})$ *on* DynBV*. Then for all $i \in [n]$ and $t \in \mathbb{N}$ we have $p_{i,t+1} = \min\left\{\max\left\{\bar{p}, p'_{i,t+1}\right\}, 1-\bar{p}\right\}$ where*

$$p'_{i,t+1} = \begin{cases} p_{i,t}, & \text{with probability } 1 - 2p_{i,t}(1-p_{i,t}) \\ p_{i,t} + \frac{1}{K}, & \text{with probability } \left(\frac{1}{2} + \Theta\left(\frac{1}{\max\{V_t,1\}}\right)\right) 2p_{i,t}(1-p_{i,t}) \\ p_{i,t} - \frac{1}{K}, & \text{with probability } \left(\frac{1}{2} - \Theta\left(\frac{1}{\max\{V_t,1\}}\right)\right) 2p_{i,t}(1-p_{i,t}) \end{cases}.$$

This implies $\mathbb{E}[p_{i,t+1} - p_{i,t} \mid p_{i,t}] = \Theta\left(\frac{p_{i,t}(1-p_{i,t})}{K \cdot \max\{V_t,1\}}\right)$, where the lower bound requires $p_{i,t} < 1 - \bar{p}$ and the upper bound requires $p_{i,t} > \bar{p}$.

4 Lower Bound on the Runtime

In this section, we sketch the proof for the lower bounds on the runtime of the compact Genetic Algorithm on Dynamic BinVal when K is polynomial in the number of bits. The idea is to bound the number of signal steps that a given bit makes over a certain number of iterations, and use this bound to show that a linear number of frequencies stay a constant distance away from the boundaries

for $\Omega(K \cdot \min\{K, n\})$ iterations. As long as this is the case, the probability to sample the optimum in any given iteration is exponentially small, and hence a union bound over the iterations gives us a high probability lower bound on the runtime. We start by upper bounding the probability that a fixed frequency gets a signal step in a given iteration, under the condition that enough bits have marginal probabilities far away from the boundaries.

Corollary 7. *Let $K \geq 1$ and $\bar{p} \in (0, \frac{1}{2})$ be arbitrary, and consider the algorithm* cGA$(K, \bar{p})$ *on* DynBV. *Assume that at iteration t there are at least γn bits whose marginal probabilities are within $[\frac{1}{6}, \frac{5}{6}]$, for some constant $\gamma > 0$. Then the probability of having a signal step on any fixed bit is $O(1/n)$.*

The following lemma guarantees, using Chernoff bounds, that the displacement of the marginal probabilities caused by $O(K^2)$ random steps is bounded in absolute value by $\frac{1}{6}$ for a linear number of bits.

Lemma 8. *Let $K \geq 1$ and $\bar{p} \in (0, \frac{1}{2})$ be arbitrary, and consider the algorithm* cGA$(K, \bar{p})$ *on* DynBV. *Consider a fixed frequency $i \in [n]$ and let $t \leq \alpha K^2$, where $\alpha > 0$ is a small enough constant. With probability $\Theta(1)$, the first t random steps of frequency i lead to a total change of the bit's marginal probability that is within $\left[-\frac{1}{6}, \frac{1}{6}\right]$.*

Moreover, for a small enough constant $\gamma > 0$, the probability that the above holds for less than γn bits among the first $\frac{n}{2}$ bits is $2^{-\Omega(n)}$, regardless of the decisions made on the last $\frac{n}{2}$ bits.

We assemble things in the next lemma and show that there is a constant fraction of bits whose marginal probabilities stay bounded away from the boundaries (and hence receive few signal steps). The proof bounds the accumulated effect of signal steps using Chernoff bounds.

Lemma 9. *Let $K \geq 1$ and $\bar{p} \in (0, \frac{1}{2})$ be arbitrary, and consider the algorithm* cGA$(K, \bar{p})$ *on* DynBV. *There exist constants $\alpha, \gamma > 0$, such that the following holds with high probability, regardless of the last $\frac{n}{2}$ bits (i.e. an adversary may choose the value of those bits in the offspring). There is a subset S of γn bits among the first $\frac{n}{2}$ bits such that during the first $t := \alpha K^2$ iterations:*

i) the marginal probabilities of all bits in S always lie in the interval $\left[\frac{1}{6}, \frac{5}{6}\right]$;
ii) the total number of signal steps for each bit in S is bounded by $\frac{K}{6}$, leading to a displacement of at most $\frac{1}{6}$.

Theorem 1 is then a direct consequence of the above lemma. Theorem 2 follows in a similar manner, by showing that there is a linear subset of the other $\frac{n}{2}$ bits which make $\Omega(K^2)$ random steps and at most εK signal steps, and the random steps lead them to a value below $\beta - \varepsilon$. We omit the details.

5 Upper Bound on the Runtime for the Conservative Regime

This section is dedicated to an upper bound (Theorem 3) on the runtime for population sizes $K = \Omega(n \log n)$ that matches Theorem 1. We slightly adjust the boundaries by setting $\bar{p} = \frac{1}{cn}$ for some large enough constant c.

Our first proposition states that for $K = \Omega(n \log n)$, the marginal probability of any bit above at least some constant threshold β will not drop below a fixed constant $0 < \alpha < \beta$ in the next polynomially many iterations. In other words, even moderately high marginal probabilities do not drop by much for a while. The proof is a straightforward application of the negative drift theorem.

Proposition 10. *Let $\bar{p} \in (0, \frac{1}{2})$ be arbitrary and let $K \geq 1$, and consider the algorithm* cGA$(K, \bar{p})$ *on* DynBV. *Let $\bar{p} < \alpha < \beta < 1-\bar{p}$ and $\gamma > 0$ be constants. There exists a constant $c' > 0$ (possibly depending on α, β, and γ) such that for a specific bit the following holds: If the bit has marginal probability at least β and $K \geq c' \cdot n \log n$, then the probability that during the following n^γ iterations the marginal probability decreases below α is at most $O(n^{-\gamma})$.*

The next lemma gives an upper bound on the time until the marginal probability reaches any threshold τ, irrespective of where the marginal probability starts. It allows us to bound the "recovery time" needed for bits whose marginal probabilities travel to the lower boundary.

Lemma 11. *Let $c > 0$ be a constant and let $K \geq 1$. Let $\tau \in \left[\frac{1}{cn}, 1 - \frac{1}{cn}\right]$ and consider some fixed bit i in the* cGA$(K, \frac{1}{cn})$ *on* DynBV *with an arbitrary value for the initial marginal probability. Then the expected time until the marginal probability p_i of this bit reaches at least $p_i = \tau$ is $O(Kn^2)$.*

To finish the proof of Theorem 3, we now show that with probability $\Omega(1)$, after $O(Kn)$ iterations either the global optimum has been found (*success*) or at least one of the frequencies - which were initially all at least $\frac{1}{2}$ - has dropped below some constant $\eta < \frac{1}{2}$ (*failure*). From a *failure*, we can recover quickly in expectation (they are unlikely by Proposition 10 and recovery is fast by Lemma 11). One then concludes by an application of the variable drift theorem.

6 Upper Bound on the Runtime for the Aggressive Regime

In this section, we analyze the cGA when $K = \Theta(\log^2 n)$, culminating in Theorem 4. Throughout, we consider a capping probability of $\bar{p} := \frac{1}{n \log^c n}$ for some constant $c > 0$, but our simulations indicate that the result should also hold for the classical $\frac{1}{n}$ capping probability. Theorem 4 is stated with $c = 7$ but the proof would go through for any $c \geq 7$. It would be possible to reduce this constant, but we aimed for simpler proofs and have not tried to optimize it.

The proof of Theorem 4 proceeds in four main steps. First, we show that due to the high genetic drift, the frequencies essentially start by executing random walks until they reach one of the boundaries. As a consequence, the sampling variance V_t drops from $\Theta(n)$ to $O(\log n)$ during the first $O(\text{polylog}(n))$ iterations, and then stays below $O(\log n)$ for the remainder of the optimization time with high probability. We call this initial phase the *burn-in phase*.

Proposition 12. *For $K = \Theta\left(\log^2 n\right)$ consider the algorithm $\text{cGA}(K, \frac{1}{n \log^7 n})$ on DynBV. After the first $O\left(K^3 \log n\right)$ iterations, with high probability the sampling variance V_t will stay below $O(\log n)$ for at least n^2 consecutive iterations.*

The proof of the above proposition requires the following lemma that bounds the time until a given frequency reaches one of the boundaries, and is proved using coupling to a fair random walk and standard drift analysis tools.

Lemma 13. *Let $c > 0$ be a constant and $K = \omega(1)$, and consider the frequency $p_{i,t}$ of a bit i of the algorithm $\text{cGA}(K, \frac{1}{n \log^c n})$ on DynBV. Let T denote the first time that $p_{i,t}$ reaches one of the boundaries. Then for every initial value $p_{i,0}$ and all $r \geq 8$, $\mathbb{E}[T \mid p_{i,0}] \leq 4K^2 \ln K$ and $\Pr[T \geq rK^2 \ln K \mid p_{i,0}] \leq 2^{-\lfloor r/8 \rfloor}$.*

With that result, the proof of Proposition 12 consists of dividing the optimization into phases of length $O\left(K^3 \log n\right)$, and showing that with high probability, during a single such phase, all the frequencies that were not at the boundaries at the start of the phase will return to one of the boundaries, and the number of frequencies that detach from the boundaries during that same phase is $O(\log n)$.

Given the bound on the sampling variance $V_t = O(\log n)$, we show, using the negative drift theorem, that frequencies at the upper boundary are unlikely to drop below a constant. This basically ensures that, while frequencies from the lower boundary may reach the upper boundary, the converse does not happen.

Lemma 14. *Let $\bar{p} \in (0, \frac{1}{2})$ be arbitrary and let $K \geq 1$, and consider the algorithm $\text{cGA}(K, \bar{p})$ on DynBV. Let $\bar{p} < \alpha < \beta < 1 - \bar{p}$ and $\gamma > 0$ be constants. Assume that $V_t = O(\log n)$ holds for the variance throughout the optimization time. Then there exists a constant $c' > 0$ (possibly depending on α, β, and γ) such that for a specific bit the following holds: If the bit has marginal probability at least β and $K \geq c' \cdot \log^2 n$, then the probability that during the following n^γ iterations the marginal probability decreases below α is at most $O(n^{-\gamma})$.*

We then argue that indeed all the frequencies starting from the lower boundary after the burn-in phase reach the upper boundary (at least once) within $O(n \cdot \text{polylog}(n))$ iterations using the following proposition. The proof uses Lemma 13 and a coupling of the process $p_{i,t}$ with a fair gambler's ruin random walk with self-loops.

Proposition 15. *Let $c > 0$ be a constant and $K \geq 1$, and consider the algorithm $\text{cGA}(K, \frac{1}{n \log^c n})$ on DynBV. Let $i \in \{1, \ldots, n\}$ be an arbitrary bit at the lower boundary, and assume that $V_t = O(\log n)$ for the rest of the optimization. Then, the expected number of iterations until the frequency of bit i reaches the upper boundary is in $O(K^4 n \log^c n)$.*

This puts us in a situation where $n - O(\text{polylog}(n))$ frequencies are at the upper boundary, and the remaining $O(\text{polylog}(n))$ frequencies are lower-bounded by a constant. We now show that with high probability all $O(\text{polylog}(n))$ frequencies reach the upper boundary while no frequency detaches from the upper boundary, and this process only takes $O(\text{polylog}(n))$ iterations. Hence we finally reach a state where $p_{i,t} = 1 - \frac{1}{n \log^c n}$ holds for all positions i, from which the optimum is then sampled with high probability in a single iteration, and that leads to the termination of the algorithm.

7 Simulations

In this section, we provide simulations that complement our theoretical analysis. All figures depict the optimization of DynBV for varying hypothetical population size K.[6] The dimension of the search space is always $n = 300$. The probabilities $p_i, i = 1, \ldots, n$ are initialized with $\frac{1}{2}$ as in the pseudocode of the algorithm. The lower and upper boundary are set to $\frac{1}{n}$ and $1 - \frac{1}{n}$. The algorithm stops when the optimum has been sampled, or after $200'000$ iterations if the optimum has not been sampled at that point. The code for the simulations is provided on request.

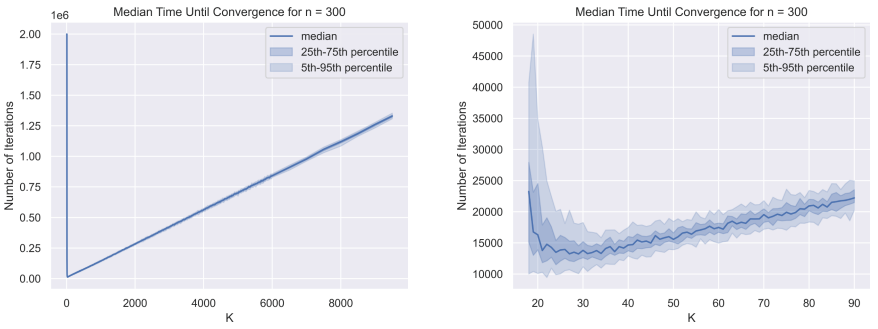


Fig. 1. Number of iterations for the optimization of Dynamic BinVal with the cGA when $6 \leq K \leq 10000$. The right plot shows the subinterval $18 \leq K \leq 90$. The median over 50 runs is plotted.

The regime of small population sizes is shown in the right plot of Fig. 1. Even at a small search space dimension of $n = 300$, the asymptotic speed-up of small K is clearly visible. For $K = 6, 7$, the optimum is not reached before the number of iterations are capped. This is in line with the observation that even for OneMax, when $K = o(\log(n))$, the runtime of the cGA becomes exponential. However,

[6] For $6 \leq K \leq 420$ all integer values of K are simulated, for $421 \leq K \leq 1000$ all integer multiples of 5, for $1001 \leq K \leq 6000$ integer multiples of 20, for $6001 \leq K \leq 10000$ integer multiples of 500.

for $10 \leq K \leq 20$ we observe a phase transition, with the minimal runtime attained for hypothetical population sizes K around 30. Due to the small problem dimension, it is difficult to tell if the threshold is located at $K = \Theta(\log n)$, at $K = \Theta(\text{polylog}(n))$, or even $K = \Theta(n^c)$ for some small $c < 1$. But the data is consistent with the theoretical result that the optimum is obtained for the sublinear K regime of genetic drift. For $K = \Omega(n \log n)$, Theorems 9 and 3 show an asymptotically tight runtime bound of $\Theta(Kn)$. Figure 1 covers a range of $K = 6$ up to $K = 10000$, exceeding the search space dimension of $n = 300$ by 2–3 orders of magnitude. We see indeed that the runtime increases proportionally to K, thus confirming our theoretical findings. In particular we see that, contrary to the optimization of OneMax, there are no local minima after the transition from the exponential to the polynomial regime. Furthermore, the plot indicates that the runtime scales linearly with K in practice much earlier than our theoretical bound from Theorem 3.

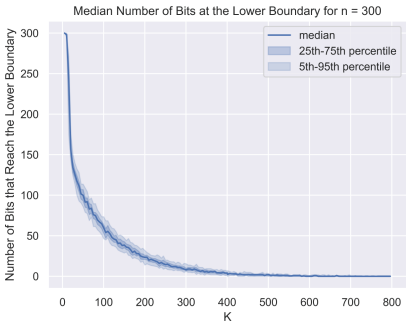


Fig. 2. Number of bits that reach the lower boundary $1 - \frac{1}{n}$ for the range $5 \leq K \leq 800$. The median over 20 runs is plotted.

In Fig. 2, we see that after an initial exponential decrease, which is similar to the initial exponential runtime decrease in Fig. 1, the number of frequencies ever reaching the lower boundary tapers off only slowly. In particular, for the empirically optimal value $K \approx 30$ from Fig. 1 still many frequencies reach the lower boundary, confirming that this is in the aggressive regime of strong genetic drift. Until $K = n$, there is still a double-digit number of bits which reach the lower boundary. Only after approximately $K = 500 = \frac{5}{3}n$ the median drops to zero.

Acknowledgments. M.K. and U.S. were supported by the Swiss National Science Foundation [grant number 200021_192079]. The Dagstuhl seminar 22182 "Estimation-of-Distribution Algorithms: Theory and Applications" gave inspiration for this work.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Dang, D.C., Lehre, P.K., Nguyen, P.T.H.: Level-based analysis of the univariate marginal distribution algorithm. Algorithmica **81** (2019)
2. De Bonet, J., Isbell, C., Viola, P.: Mimic: Finding optima by estimating probability densities. In: Advances in Neural Information Processing Systems, vol. 9 (1996)
3. Doerr, B.: The runtime of the compact genetic algorithm on jump functions. Algorithmica **83**, 3059–3107 (2021)
4. Doerr, B., Krejca, M., Lehre, P.K.: Estimation-of-distribution algorithms: theory and applications. Panel discussion (2022). https://doi.org/10.4230/DagRep.12.5.17
5. Doerr, B., Krejca, M.S.: Significance-based estimation-of-distribution algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1483–1490 (2018)
6. Doerr, B., Krejca, M.S.: A simplified run time analysis of the univariate marginal distribution algorithm on LeadingOnes. Theoret. Comput. Sci. **851**, 121–128 (2021)
7. Doerr, B., Krejca, M.S.: The univariate marginal distribution algorithm copes well with deception and epistasis. Evol. Comput. **29**(4), 543–563 (2021)
8. Doerr, B., Zheng, W.: Sharp bounds for genetic drift in estimation of distribution algorithms. IEEE Trans. Evol. Comput. **24**(6), 1140–1149 (2020)
9. Droste, S.: A rigorous analysis of the compact genetic algorithm for linear functions. Nat. Comput. **5**, 257–283 (2006). https://doi.org/10.1007/s11047-006-9001-0
10. Florescu, C., Kaufmann, M., Lengler, J., Schaller, U.: Faster optimization through genetic drift. arXiv preprint arXiv:2404.12147 (2024)
11. 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**(3), 477–490 (2016)
12. Friedrich, T., Kötzing, T., Neumann, F., Radhakrishnan, A.: Theoretical study of optimizing rugged landscapes with the cGA. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) Parallel Problem Solving from Nature – PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part II, pp. 586–599. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-14721-0_41
13. Gießen, C.: Theory of randomized optimization heuristics (Dagstuhl Seminar 17191). Dagstuhl Rep. **7**(5), 22–55 (2017). https://doi.org/10.4230/DagRep.7.5.22
14. Kaufmann, M., Larcher, M., Lengler, J., Sieberling, O.: Hardest monotone functions for evolutionary algorithms (2023)
15. Kaufmann, M., Larcher, M., Lengler, J., Zou, X.: OneMax is not the easiest function for fitness improvements. In: Pérez Cáceres, L., Stützle, T. (eds.) Evolutionary Computation in Combinatorial Optimization: 23rd European Conference, EvoCOP 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings, pp. 162–178. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30035-6_11
16. Krejca, M.S., Witt, C.: Theory of estimation-of-distribution algorithms. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 405–442. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_9

17. Lehre, P.K., Nguyen, P.T.H.: On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help. In: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, pp. 154–168 (2019)
18. Lehre, P.K., Nguyen, P.T.H.: Runtime analysis of the univariate marginal distribution algorithm under low selective pressure and prior noise. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1497–1505 (2019)
19. Lengler, J.: Drift analysis. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 89–131. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_2
20. Lengler, J., Meier, J.: Large population sizes and crossover help in dynamic environments. Nat. Comput. **23**(1), 115–129 (2024). https://doi.org/10.1007/s11047-022-09915-0
21. Lengler, J., Riedi, S.: Runtime analysis of the (μ+ 1)-EA on the dynamic BinVal function. Evol. Comput. Comb. Optim. **12692**, 84–99 (2021)
22. Lengler, J., Schaller, U.: The (1+ 1)-EA on noisy linear functions with random positive weights. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 712–719. IEEE (2018)
23. Lengler, J., Sudholt, D., Witt, C.: The complex parameter landscape of the compact genetic algorithm. Algorithmica **83**, 1096–1137 (2021)
24. Pelikan, M., Lin, T.-K.: Parameter-less hierarchical BOA. In: Deb, K. (ed.) Genetic and Evolutionary Computation – GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part II, pp. 24–35. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24855-2_3
25. Sudholt, D., Witt, C.: On the choice of the update strength in estimation-of-distribution algorithms and ant colony optimization. Algorithmica **81**, 1450–1489 (2019)
26. Witt, C.: Upper bounds on the running time of the univariate marginal distribution algorithm on OneMax. Algorithmica **81**, 632–667 (2019)
27. Witt, C.: How majority-vote crossover and estimation-of-distribution algorithms cope with fitness valleys. Theoret. Comput. Sci. **940**, 18–42 (2023)

Greedy Versus Curious Parent Selection for Multi-objective Evolutionary Algorithms

Denis Antipov[1], Timo Kötzing[2], and Aishwarya Radhakrishnan[2(✉)]

[1] University of Adelaide, Adelaide, Australia
denis.antipov@adelaide.edu.au
[2] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
{timo.koetzing,aishwarya.radhakrishnan}@hpi.de

Abstract. From the literature we know that simple evolutionary multi-objective algorithms can optimize the classic two-objective test functions OneMinMax and CountingOnesCountingZeroes in $O(n^2 \log n)$ expected time. We extend this result to any pair of generalized OneMax functions and show that, if the optima of the two functions are d apart, then (G)SEMO has an expected optimization time of $O(dn \log(n))$.

In an attempt to achieve better optimization times, some algorithms consider parent selection. We show that parent selection based on the curiosity-based novelty search can improve the optimization time to $O(n^2)$ on OneMinMax. By contrast, we show that greedy parent selection schemes can be trapped with an incomplete Pareto front for super-polynomial time.

Finally, we provide experimental results on the two-objective optimization of linear functions.

Keywords: Evolutionary Algorithm · Multi-Objective Optimization · Run time analysis

1 Introduction

While evolutionary algorithms [16] might be most famous for applications on single-objective problems, the setting of optimizing multiple criteria at once is particularly suitable for an approach with population-based methods, since different candidate solutions might be incomparable: while one solution is better than another in terms of the first criterion, the situation is reversed in terms of the second criterion, and so on. Thus it makes sense to retain all non-dominated solutions (where no other solution is better in *all* objectives), naturally giving a population of solutions. The analysis of the search behavior and search performance has been the subject of significant theoretical analysis [10–12,17,23,28].

Core starting point of all theoretical research is the classic benchmark problem OneMax, which was extended to the setting of two objectives in two ways. The first uses the classic OneMax function as one objective and the direct opposite, *minimizing* the number of 1s instead of maximizing it, as the other objective.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 86–101, 2024.
https://doi.org/10.1007/978-3-031-70071-2_6

Using these two objectives is called OneMinMax. The second extension considers less conflicting bits: While the second half of the bits stay in conflict, the first half are shared. This is called CountingOnesCountingZeroes (COCZ).

The behavior of the classic SEMO (Simple Evolutionary Multi-Objective) and GSEMO (Global Simple Evolutionary Multi-Objective) algorithms is well-understood on these two problems (see [4,20,22,26]). The expected run time of SEMO and GSEMO to cover the whole Pareto front while minimizing OneMinMax and COCZ is $\Theta(n^2 \log(n))$ and the theoretical analyses can be found in [4,20,22].

For the case of single-objective optimization, the broader class of linear functions [14] gives an important extension of the simple OneMax function, extending it to a sizable class of functions. This was a driver for further development of the field [27]. While OneMax as a member of this class has been analyzed for the two-objective case, no other linear functions where considered.

With this paper, we first provide a general definition of two-objective problems where both objectives are derived from the OneMax test function; we call this OMC, the OneMax function class. Analogously, we define LFC, the linear function class. In Sect. 4 we introduce and discuss these function classes and study some of their properties. We also include a proof of the expected run time of (G)SEMO on two elements of OMC being $O(dn \log(n))$ in dependence on the distance $d > 0$ of the optima of the two objective functions. Note that this shows the smooth transition of run time $O(n \log(n))$ when using twice the same objective and $O(n^2 \log(n))$ for complementary objectives (as in OneMinMax) and also recovers the run time bound for COCZ.

The considered algorithms typically waste a lot of time reconsidering old search points which are already optimal and where no more progress can be made in the proximity. This inspires algorithms based on considering new search points rather than reconsidering old ones. This paradigm is called novelty search and in the literature the algorithm is known as fair evolutionary multi-objective optimizer (FEMO) [18,22,23]. In Sect. 6 we consider a simple variant of such an algorithm which maintains, for each phenotype, a counter of how often it was considered for creating offspring. Each iteration, an individual with minimal counter is considered for creating offspring. We show, in Theorem 4, that this algorithm has an expected optimization time of $O(n^2)$ on OneMinMax, improving over (G)SEMO.

Novelty search modifies which individuals are considered for creating offspring, while leaving the rest of the algorithm as is. This is called a *parent selection scheme* and the literature knows a variety of other mechanisms [4,5]. These schemes typically rank all individuals of the population according to how promising they are to create relevant offspring and then prefer more promising ones over less promising ones. In Sect. 7 we show that, for many ranking schemes and too radically greedy preference of more promising points, we get super-polynomial optimization on OneMinMax with constant probability (see Theorem 6). We consider this as a cautionary tale that parent selection schemes

need to reconsider less promising search points from time to time, even on very easy fitness landscapes (such as OneMinMax).

Finally, in Sect. 8, we provide experimental evidence for the expected run time performance of GSEMO on two anti-aligned LFC functions without any shared bits and GSEMO on two anti-aligned LFC functions without any conflicted bits. Our results hint at an asymptotic run time of $O(n^2 \cdot \log(n))$ for anti-aligned LFC functions without any shared bits and $O(n \cdot \log(n))$ for anti-aligned LFC functions without any conflicted bits.

The remainder of this paper first gives some discussion on further related work (see Sect. 2). We give important definitions in Sect. 3. We introduce OMC formally (along with the extension to linear functions) in Sect. 4 and analyze the run time of (G)SEMO on OMC in Sect. 5. We analyze novelty search in Sect. 6 and greedy parent selection in Sect. 7. We conclude with some experiments in Sect. 8. Many proofs are not included into this document, but can be found in the supplementary material [1].

2 Related Works

In [23], a first run time analysis was conducted on the simple multi-objective optimization algorithm (SEMO) on minimizing LeadingOnesTrailingZeros (LOTZ). This work was extended in [22] to the fair and the greedy multi-objective optimization algorithms (FEMO, GEMO) and the multi-start (1+1) EA on COCZ and LOTZ.

The global simple multi-objective optimization algorithm (GSEMO) was analyzed on LOTZ in [19] along with a lower bound on GSEMO for a general class of pseudo-boolean functions. GSEMO and GSEMO with asymmetric mutation operator on plateau functions and set cover instances were studied in [3,17]. In [18], the performance of GSEMO and Global-FEMO algorithms on plateaus, plateaus with gap and dual path were analyzed. In [25], analysis of GSEMO with mixed strategy (mixing selection mechanisms) on ZPLG (ZeroMax, a plateau, and a path with little gaps) and SPG (shortest path and gaps) can be found.

The algorithms SEMO and GSEMO with crossover operators on COCZ and minimum spanning tree (MST) problems were studied in [26]. The first analysis of SEMO optimizing OneMinMax was given in [20]. The OneMinMax function was again analyzed in [8], but using the $(\mu + 1)$-SIBEA$_D$ algorithm. The decomposition-based multi-objective evolutionary algorithms (MOEAs) were introduced in [24] and analyzed on COCZ and LOTZ.

A diversity-based parent selection mechanism (based on hypervolume contribution) for SEMO and GSEMO was given in [4,5] and studied on minimizing OneMinMax and LOTZ. In [12], SEMO and GSEMO were studied on optimizing the OneJumpZeroJump function. An offspring selection mechanism which uses the total Hamming distance as a diversity measure was given in [2] and the OneMinMax function was again analyzed in this setting.

3 Preliminaries

In this section we give some definitions, the algorithms we analyze and some notations which we use throughout the paper. We use the following theorem in some of our proofs.

Theorem 1 (Multiplicative Drift Theorem [21]). *Let $(X_t)_{t\in\mathbb{N}}$ be a random process over $\mathbb{R}$, $x_{\min} > 0$, $\delta > 0$ and let $T = \min\{t \mid X_t < x_{\min}\}$. Furthermore, suppose that*

1. *$X_0 \geq x_{\min}$ and, for all $t \leq T$, it holds that $X_t \geq 0$, and that*
2. *for all $t < T$, we have $X_t - E[X_{t+1} \mid X_0, \ldots, X_t] \geq \delta X_t$.*

Then

$$E[T \mid X_0] \leq \frac{1 + \ln\left(\frac{X_0}{x_{\min}}\right)}{\delta}.$$

We analyze the simple multi-objective optimizer (SEMO) and the global multi-objective optimizer (GSEMO) algorithms (see Algorithm 1) on different bi-objective functions in this paper. The only difference between SEMO and GSEMO is the mutation step. In SEMO, at the mutation step, a bit position is chosen uniformly at random and flipped (one bit mutation). In GSEMO, each bit position is flipped with probability $1/n$ (standard bit mutation).

The initial population has only one individual chosen at random from $\{0,1\}^n$. An individual x dominates another individual y ($x \succeq y$) if and only if $f_1(x) \leq f_1(y)$ and $f_2(x) \leq f_2(y)$. Note that we use slightly different Pareto dominance relation which prefers the offspring if the offspring has the same fitness as any of the other individuals existing in the population. We use the term genotype to refer to the individuals in the input domain and the term phenotype to refer to the fitness vector.

ïż£

Algorithm 1: (Global) Simple Evolutionary Multi-objective Optimizer ((G)SEMO) minimizing $f = (f_1, f_2)$.

```
1 x ← choose u.a.r from {0,1}^n, P ← {x};
2 while termination criteria not met do
3 |   select parent x from P u.a.r;
4 |   x' ← mutate(x);
5 |   P ← P \ {z ∈ P | x' ⪰ z};
6 |   if ∄z ∈ P s.t (z ⪰ x') then P ← P ∪ {x'}
```

When discussing greedy parent selection schemes, we use the following (simplified) definition of the hypervolume contribution for 2 objectives, usually used in minimization problems.

Definition 1. *Consider a bi-objective function $f = (f_1, f_2)$ and a population of points $P = (x_1, \ldots, x_\mu)$ which do not dominate each other (in terms of f) and are sorted in the ascending order of their f_1 value. Let $r = (r_1, r_2)$ be a reference point such that $r_1 \geq f_1(x_\mu)$ and $r_2 \geq f_2(x_1)$. For all $i \in [1..\mu]$ let $a_i = f_1(x_i)$ and let $b_i = f_2(x_i)$. Let also $a_{\mu+1} = r_1$ and $b_0 = r_2$. Then for all $i \in [1..\mu]$ the hypervolume contribution (HVC) of point $x_i \in P$ is $(a_{i+1} - a_i) \cdot (b_{i-1} - b_i)$.*

4 Linear Multi-objective Functions

In this section we analyze two classes of functions: first, the OneMax function class, where each fitness function measures the Hamming distance to some optimal bit string. Second, the linear function class, where each bit has a weight and fitness is the sum of the weights of incorrect bits.

Formally, for each $a \in \{0,1\}^n$, we let

$$\mathrm{OM}_a : \{0,1\}^n \to \mathbb{R}, x \mapsto H(a,x),$$

where H is the Hamming distance between two bit strings. We define the OneMax class as

$$\mathrm{OMC} = \{\mathrm{OM}_a \mid a \in \{0,1\}^n\}.$$

Note that the OneMax class has been studied before in the context of black-box optimization [7,13,15]. The most famous example from OMC is $\mathrm{OM} := \mathrm{OM}_{1^n}$ which is minimal at 1^n. For two-objective optimization, we also care for the exact opposite, ZeroMax, denoted as $\mathrm{ZM} := \mathrm{OM}_{0^n}$.

Similarly, we can define the linear function class LFC as follows. For each $w \in \mathbb{R}^{n+1}$, we let

$$f_w : \{0,1\}^n \to \mathbb{R}, x \mapsto w_{n+1} + \sum_{i=1}^{n} w_i \, x_i.$$

Note that we use the constant w_{n+1} (a) as an offset, so that all function values are non-negative and can be more nicely depicted in a diagram; and (b) so that it is formally true that each OneMax function is a linear function, which they intuitively are (for example, for $a = 1^n$ we need $\forall i \in [1..n] : w_i = -1$ and $w_{n+1} = n$).

In the literature we frequently find the additional restrictions $w_{n+1} = 0$ and $\forall i \leq n : w_i > 0$; or even $\forall i < n : w_i > w_{i+1} > 0$. These can be assumed without loss of generality to simplify the exposition or the proof in the context of single objective optimization. However, in the context of optimizing two such functions simultaneously, and with the algorithm potentially making decisions not just based on the ranking of search points (but, for example, also based on hypervolume covered), we prefer this more general definition here.

We define the linear function class as

$$\mathrm{LFC} = \left\{ f_w \mid w \in \mathbb{R}^{n+1} \right\}.$$

We have the following theorem about the two function classes and the proof can be found in the supplementary material [1].

Theorem 2. *We have* OMC $\subseteq$ LFC*, and both* OMC *and* LFC *are closed under isomorphisms of the hypercube.*

We use the following definition to talk about the fitness landscape of two linear fitness functions.

Definition 2. *Let two linear functions f_w, f_v be given. We call the set $I = \{i \in [n] \mid w_i \cdot v_i \geq 0\}$ the* shared bits*, since there is a bit setting which is optimal for both f_w and f_v. We call $[n] \setminus I$ the* conflicted bits*. We call the number of conflicted bits the* Pareto dimension*, since all elements on the Pareto front agree on the shared bits and only differ on conflicted bits (discarding the case of weights of* 0*). We frequently denote the Pareto dimension by d.*

If all n bits are conflicted, then we call f_w and f_v complementary *(since their unique global optima are complementary).*

We call f_w and f_v anti-aligned *if ordering the bits descendingly according to $|w_i|$-value leads to an ascending ordering according to $|v_i|$-value. In other words: the more significant bit positions of w are, the less significant bit positions of v (and vice versa).*

5 SEMO and GSEMO on OMC

Here we give a generalization of the OneMinMax and COCZ analysis to the situation where the optima can share any number of bits (rather than either 0 bits as for OneMinMax or $n/2$ bits as in COCZ). Let $\log^+(x) = \max\{\log(x), 1\}$.

Theorem 3. *Let $a, b \in \{0,1\}^n$, $a \neq b$, let $d = d_H(a,b)$ and $0 < d < n$. Then (G)SEMO minimizing $(\mathrm{OM}_a, \mathrm{OM}_b)$ takes $O(dn \log(n))$ function evaluations in expectation to discover the full Pareto front of size $d+1$.*

Proof. First we show that the expected time for (G)SEMO to find an individual on the Pareto front is $O(dn \log^+(n-d))$ using the multiplicative drift theorem (see Theorem 1).

Let T_1 be the time taken by (G)SEMO to find an individual on the Pareto front, and let I be the set of all shared bits, i.e., $I = \{i \in [n] \mid a_i = b_i\}$. Since $d = d_H(a,b)$, $|I| = n - d$. Also, an individual x is on the Pareto front if and only if all shared bits of a and b (elements of I) are set correctly, i.e., $\sum_{i \in I} |a_i - x_i| = \sum_{i \in I} |b_i - x_i| = 0$.

For any $t > 0$, let P^t be the parent population at iteration t. For any $t < T_1$, let $X^t = \arg\min_{x \in P^t}\{\sum_{i \in I} |a_i - x_i|\}$. Then we claim that, $X^t \geq X^{t+1}$, i.e., an individual with less correct shared bits will not dominate an individual with more correct shared bits. If an individual x has $1 < i \leq n$ more shared bits set correctly than another individual y, then for y to dominate x the individual y should have i more conflicted (non-shared) bits (than x) set correctly with respect to a and i more conflicted bits set correctly with respect to b. This is not possible, since setting a conflicted bit correctly with respect to a implies that this conflicted bit is set incorrectly with respect to b. Therefore, for all t we have, $X^t \geq X^{t+1}$.

We claim that at any time t, the population P^t has at most $d+1$ individuals. Since there are only d conflicted bits, if there are $d+2$ individuals in the population then, by the pigeonhole principle, there is an $i \in [0..d]$ such that there are two distinct individuals in the population in which i conflicted bits are set correctly with respect to a. Since both the individuals exist in the population, their fitness is different. Therefore, one of them has more shared bits set correctly than the other, which implies that one individual dominates the other. This contradicts the definition of (G)SEMO since it only stores non-dominated individuals in its population.

Now we claim that $\Pr(X^t - X^{t+1} = 1 \mid X^t) \geq \frac{X^t}{e(d+1)n}$. As the maximum size of the population is $d+1$ and the mutation operator can choose an individual contributing to the potential X^t and flip exactly one of the X^t positions where $a_i \neq x^t$ with probability $\frac{1}{n}$ in the case of SEMO and with probability at least $\frac{1}{en}$ in the case of GSEMO.

By the multiplicative drift theorem (Theorem 1) and since we have $X^0 \leq n-d$, the expected time taken by (G)SEMO to find an individual on the Pareto front $E[T_1]$ is $O(dn \log^+(n-d))$.

Next, we show that the expected time to cover the Pareto front after the algorithm finds an individual on the Pareto front is $O(dn \ln(d))$. At time T_1, when the algorithm finds an individual x^T on the Pareto front for the first time, the fitness of this first individual cannot be more than d in both objectives, since, as we mentioned before, an individual is on the Pareto front if and only if all shared bits of a and b are set correctly. Let the fitness of this first individual be $(i, d-i)$, where $0 \leq i \leq d$.

Let $Y_t = -1$, if all fitness vectors from $(0, d)$ to $(i, d-i)$ are in population P^t, and let it be the maximum $j < i$ such that we do not have fitness $(j, d-j)$ in population P^t otherwise. Similarly, let Z_t be $d+1$, if we have all fitness vectors from $(i, d-i)$ to $(d, 0)$ in the population P^t, and let it be the minimum $k > i$ such that $(k, d-k)$ is not in P^t otherwise. Then the Pareto front is covered, iff $Y_t = -1$ and $Z_t = d+1$. Consider first time T_2' until Y_t reaches -1. If $Y_t = j$, then to decrease it we can choose an individual with fitness $(j+1, d-j-1)$ (which exists in the population) with probability at least $\frac{1}{d+1}$ and flip exactly one one-bit in it with probability $\frac{j+1}{n}$ or $\frac{j+1}{en}$ for SEMO or GSEMO respectively. Since for each value of j we decrease Y_t only once, the total expected time until we have $Y_t = -1$ is at most $E[T_2'] \leq \sum_{j=0}^{i-1} \frac{en(d+1)}{j+1} = O(nd \log^+(i))$. Similarly, we can show that the expected time T_2'' until Z_t reaches $d+1$ is at most $O(nd \log^+(d-i))$. We then have that $T_2 \leq T_2' + T_2''$ and therefore, it is at most $O(nd \log(d))$.

Overall, (G)SEMO takes $O(dn \log^+(n-d) + dn \log(d)) = O(dn \log(n))$ iterations in expectation while minimizing $(\mathrm{OM}_a, \mathrm{OM}_b)$ to discover the full Pareto front of size $d+1$. □

6 Novelty Search

Consider as an order scheme the ranking of all individuals by how often they have been considered for creating offspring since entering the population, from least

to most frequently considered. In a sense, we want to explore under-explored parts of the search space, and we want to find novel areas. This can lead to speed-ups in exploration of the Pareto front, as the next theorem shows.

We distinguish two cases for novelty search: resetting the offspring counter when an individual was replaced by one with the exact same phenotype, and not doing such a reset. That is, when we reset the counter, we actually counting the number of times the *genotype* has been selected as a parent, and when we do not reset the counter, we count the number of such times for the *phenotype*. The following theorem shows a speed-up of the latter approach on OneMinMax compared to the standard parent selection. We believe that resetting the counter when replacing individuals with the same phenotype leads to a behavior much like uniform parent selection, which is not interesting for us.

Theorem 4. *(G)SEMO paired with the the novelty ranking without resets which always chooses a parent that has been considered the smallest number of times finds the whole Pareto-front on* OneMinMax *in* $O(n^2)$ *expected iterations.*

Proof. All individuals are on the Pareto-front, since more 1s contribute positively to minimize the objective ZM and negatively to the objective OM and vice versa in the case of more 0s. Any two individuals $x, y \in \{0,1\}^n$ either have the same number of 1s, which leads to the same fitness, or one of them has more 1s than the other, which implies that neither x dominates y nor y dominates x. Therefore, the set $\{(i, n-i) \mid 0 \leq i \leq n\}$ is the set of all possible fitness values which corresponds to $n+1$ individuals on the Pareto-front.

For reasons of space, in the rest of this proof we consider SEMO. The proof for GSEMO uses the same arguments, but has slightly different constants.

We break down the total run time into time taken for the following events to happen. For any $0 \leq i \leq n-1$, let X_i be the random variable which denotes the time taken to find an individual x with fitness $(i+1, n-i-1)$ after the algorithm has found an individual with fitness $(i, n-i)$ and the algorithm has chosen this individual at least n times for offspring creation. For any $1 \leq i \leq n$, let Y_i be the random variable which denotes the time taken to find an individual y with fitness $(i-1, n-i+1)$ after the algorithm has found an individual with fitness $(i, n-i)$ and has chosen this individual at least n times for offspring creation. At any given iteration, when individual with fitness $(i, n-i)$ is chosen as a parent, this individual has either not yet been selected for offspring selection n times or the algorithm tries to find a new individual in the Pareto front which is not in the population by mutation. Therefore, the expected time T to find all the elements on the Pareto-front is

$$E[T] \leq \sum_{i=0}^{n-1}(E[X_i]+n) + \sum_{i=1}^{n}(E[Y_i]+n) \leq \sum_{i=0}^{n-1} E[X_i] + \sum_{i=1}^{n} E[Y_i] + 2n^2. \quad (1)$$

Now we calculate upper bounds on the expectation of the random variables X_i and Y_i for a given i. The probability that an individual with fitness $(i, n-i)$ does not lead to an offspring with fitness $(i+1, n-i-1)$ after being chosen n times

is $(1-\frac{n-i}{n})^n \leq e^{i-n}$ and the probability that an individual with fitness $(i, n-i)$ does not lead to an offspring with fitness $(i-1, n-i+1)$ after being chosen n times is $(1-\frac{i}{n})^n \leq e^{-i}$. The number of function evaluations needed for an individual with fitness $(i, n-i)$ to produce an offspring with fitness $(i+1, n-i-1)$ by flipping exactly one bit follows the $\mathrm{Geo}(\frac{n-i}{n})$ geometric distribution and each failure costs at most n function evaluations since every other individual in the population must be selected at least as many times as this desired individual for offspring selection before this individual can be selected again. We note that if a new fitness appears in the population at this stage, then we can wait for more than n iterations before we chose our individual with fitness $(i, n-i)$ again, however those iterations when we choose an individual with this new fitness do not go to the cost of our mistake, but to the n iterations which are allocated for that new fitness in the corresponding term in eq. (1) or they go to the price of our previous mistakes. Similarly, the number of function evaluations needed for an individual with fitness $(i, n-i)$ to produce an offspring with fitness $(i-1, n-i+1)$ by flipping exactly one bit follows the $\mathrm{Geo}(\frac{i}{n})$ with the cost of at most n function evaluations for each failure. Therefore, for $0 \leq i \leq n-1$, $X_i \leq ne^{i-n} \cdot \mathrm{Geo}(\frac{n-i}{n})$ and for $1 \leq i \leq n$, $Y_i \leq ne^{-i} \cdot \mathrm{Geo}(\frac{i}{n})$. Thus, from eq. (1) we have

$$\begin{aligned} E[T] &\leq \sum_{i=0}^{n-1} E\left[ne^{i-n} \cdot \mathrm{Geo}\left(\frac{n-i}{n}\right)\right] + \sum_{i=1}^{n} E\left[ne^{-i} \cdot \mathrm{Geo}\left(\frac{i}{n}\right)\right] + 2n^2 \\ &= n^2 \sum_{i=0}^{n-1} \frac{1}{(n-i)(e^{n-i})} + n^2 \sum_{i=1}^{n} \frac{1}{ie^i} + 2n^2 = O(n^2). \end{aligned}$$

□

We have the following corollary on strictly monotone increasing functions and the proof can be found in the supplementary material [1].

Corollary 5. *Let $f, g\colon \mathbb{R} \to \mathbb{R}$ be strictly monotone increasing. Then the novelty ranking paired without reset which always chooses an individual that has been considered least number of times (top individual) leads to a run time of $O(n^2)$ on minimizing $(f(OM), g(ZM))$.*

7 Counter-Example for Phenotype-Based Methods

In this section we show that the parent selection methods which are based on the phenotype of the points in the population might be decisive even on very simple problems. We consider GSEMO with an exaggerated greedy phenotype-based parent selection: it always chooses one of the two points with the largest HVC (Definition 1) as a parent, each with probability $\frac{1}{2}$. We call this algorithm GSEMO_2 for brevity.

We study this algorithm on OneMinMax with a reference point $(2n, 2n)$, so that the largest HVC is always yielded by the two *edge* points in the population

(the points with the largest and the smallest numbers of one-bits), and therefore one of them is always chosen as a parent. The main result of this section is the following theorem, which demonstrates an ineffectiveness of the GSEMO_2.

Theorem 6. *With probability $\Omega(1)$ the $GSEMO_2$ optimizing* OneMinMax *with reference point $(2n, 2n)$ does not find all points in the Pareto front in polynomial time.*

We split the proof of Theorem 6 into three stages. The first stage of the proof shows that a run of the algorithm with high probability occurs in a particular initial state, where the two edge points are in linear distance from each other (in phenotype space), but they are still not too far away from the initial search point. In the second stage we show that starting from the initial state, we are very likely to create a *hole* in the population, when we get an edge point in distance at least two (again, in the phenotype space) from the nearest other point in the population. In the last stage we show that once we get a hole, with constant probability it stays in the population for a super-polynomial time. For reasons of space, we omit the analysis of the first two stages, but it can be found in supplementary material [1].

Theorem 6 resembles Theorem 8.1 in [5], where a similar result was proven for the GSEMO with a similar (but artificially modified) greedy parent selection on LOTZ. The main difference of our result is that we use a much more simple function, for which all points in the search space are Pareto optimal, thus we do not need to modify the selection mechanism as in [5]. Another significant difference is that in the third stage of our proof extending the front is less likely than covering the hole, while for LOTZ these events are equally likely. Despite this, the hole is also likely to stay on OneMinMax.

We use the following notation. By x_t we denote the individual in the population with the maximum OneMax value after iteration t, and by y_t the one with the minimum OneMax value. Note that in iteration $t+1$ we always choose as a parent either x_t or y_t, since they are the edge points. In our proofs we also use an arbitrary small constant ε, which can be any value in $(0, \frac{1}{10})$. For simplicity we also assume that n is even and εn is an integer. We start the proof with several auxiliary results.

Lemma 7. *Let ω_t, $t \in \mathbb{N}$, be a sequence of random experiments. Let also A_t and B_t be sequences of events over the corresponding probabilistic spaces. Let C_t be another sequence of events such that $C_t = \cap_{j=1}^{t-1} \overline{A_j}$ (that is, C_t is the event that A_j did not occur before time t) and let τ be the first time when A_t occurs, that is, $\tau = \min\{t \mid \omega_t \in A_t\}$ and assume that $\Pr[\tau = +\infty] = 0$.*

(a) If there exists p such that, for all $t \in \mathbb{N}$, we have $\Pr[B_t \mid A_t \cap C_t] \leq p$, then $\Pr[B_\tau] \leq p$.
(b) If there exists q such that, for all $t \in \mathbb{N}$, we have $\Pr[B_t \mid A_t \cap C_t] \geq q$, then $\Pr[B_\tau] \geq q$.

Proof. We prove only (a), since the proof of (b) is analogous. Event $\tau = t$ occurs, iff A_t occurs and all A_j for $j \in [1..t-1]$ do not occur, that is, it is equal to event

$$A_t \cap \left(\bigcap_{j=1}^{t-1} \overline{A_j} \right) = A_t \cap C_t,$$

hence by condition we have $\Pr[B_t \mid \tau = t] \leq p$.

Since $\Pr[\tau = +\infty] = 0$, we can use the law of total probability.

$$\Pr[B_\tau] = \sum_{t=1}^{+\infty} \Pr[\tau = t] \Pr[B_t \mid \tau = t] \leq \sum_{t=1}^{+\infty} \Pr[\tau = t] \cdot p = p.$$

□

We now show that if we create a hole in our population, which is not too far from, but also not too close to the center of the Pareto front, then with at least a constant probability we move our edge points in a linear distance from this hole before we fill it.

Lemma 8. *Consider a run of the GSEMO$_2$ on* OneMinMax. *Assume that at some iteration t_0 we have some $i \in [\frac{n}{2} + 2\varepsilon n..\frac{n}{2} + 4\varepsilon n]$ such that*

(1) we do not have fitness $(i, n-i)$ in population,
(2) $\mathrm{OM}(x_{t_0-1}) > i$, and
(3) $\mathrm{OM}(y_{t_0-1}) < i - \varepsilon n$.

Then with at least a constant (that is, $\Omega(1)$) probability we get $x_t > i + \varepsilon n$ before we generate an offspring with i one-bits.

Without proof we note that such iteration t_0 exists with probability $1 - e^{-\Omega(n)}$, which is shown in the supplementary material [1].

Proof. Assume that, at some iteration t', we have $\mathrm{OM}(x_{t-1}) = i + k$ for some $k \in [1..\varepsilon n]$. For all $t \geq t'$ let A_t be an event that we either have $\mathrm{OM}(x_t) > i + k$ or we generate an offspring with exactly i one-bits in generation t. Let B_t be an event that we have $\mathrm{OM}(x_t) > i + k$. Let also C_t be $\bigcap_{j=t'}^{t-1} \overline{A_j}$. Then by Lemma 7 and since B_t is a sub-event of A_t, the probability p_k that we get $\mathrm{OM}(x_t) > i + k$ before we cover the fitness value $(i, n-i)$ is at least

$$\begin{aligned}\Pr[B_t \mid A_t \cap C_t] &= \frac{\Pr[B_t \mid C_t]}{\Pr[A_t \mid C_t]} = \frac{\Pr[B_t \mid C_t]}{\Pr[B_t \cup (A_t \setminus B_t) \mid C_t]} \\ &= \frac{\Pr[B_t \mid C_t]}{\Pr[B_t \mid C_t] + \Pr[A_t \setminus B_t \mid C_t]} = \frac{1}{1 + \frac{\Pr[A_t \setminus B_t \mid C_t]}{\Pr[B_t \mid C_t]}}.\end{aligned}$$

Event $A_t \setminus B_t$ conditional on C_t is the event when we create an individual with exactly i one-bits. If we chose y_{t-1} as a parent, then to do this we would need to

flip at least εn bits, the probability of which is $e^{-\Omega(n)}$ by Chernoff bounds. If we choose x_{t-1} as a parent, then we need to flip at least k one-bits, the probability of which is at most $\binom{i+k}{k}(\frac{1}{n})^k$ by Lemma 1.10.37 in [6]. Consequently, we have

$$\begin{aligned}\Pr[A_t \setminus B_t \mid C_t] &\leq \frac{1}{2} \cdot e^{-\Omega(n)} + \frac{1}{2} \cdot \binom{i+k}{k} \left(\frac{1}{n}\right)^k \\ &\leq \frac{e^{-\Omega(n)} + \frac{n^k}{k!n^k}}{2} = \frac{e^{-\Omega(n)} + \frac{1}{k!}}{2}.\end{aligned}$$

The probability of B_t conditional on C_t is at least the probability that we chose x_{t-1} as a parent and flip only one zero-bit in it, that is,

$$\Pr[B_t \mid C_t] \geq \frac{1}{2} \cdot \frac{n-i-k}{n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{n - \frac{n}{2} - 5\varepsilon n}{2en} = \frac{1 - 10\varepsilon}{2e}.$$

Hence, we have

$$\Pr[B_t \mid A_t \cap C_t] \geq \frac{1}{1 + \frac{e^{-\Omega(n)} + \frac{1}{k!}}{2} \cdot \frac{2e}{1-10\varepsilon}} = \frac{1}{1 + c\left(e^{-\Omega(n)} + \frac{1}{k!}\right)},$$

where $c = \frac{e}{1-10\varepsilon} = \Omega(1)$, if $\varepsilon < \frac{1}{10}$.

The probability that we reach $\mathrm{OM}(x_t) > i + \varepsilon n$ before we cover the hole is at least the probability that for each OneMax value visited by x_t we increase this value before we cover the hole. By the law of total probability used inductively over all values of k from 1 to εn, this probability is at least

$$\begin{aligned}\prod_{k=1}^{\varepsilon n} \frac{1}{1 + c\left(e^{-\Omega(n)} + \frac{1}{k!}\right)} &= \frac{1}{\exp\left(\ln \prod_{k=1}^{\varepsilon n} \left(1 + c\left(e^{-\Omega(n)} + \frac{1}{k!}\right)\right)\right)} \\ &= \frac{1}{\exp\left(\sum_{k=1}^{\varepsilon n} \ln\left(1 + c\left(e^{-\Omega(n)} + \frac{1}{k!}\right)\right)\right)} \\ &\geq \frac{1}{\exp\left(\sum_{k=1}^{\varepsilon n} c\left(e^{-\Omega(n)} + \frac{1}{k!}\right)\right)} \\ &\geq \frac{1}{\exp\left(c\varepsilon n e^{-\Omega(n)} + ce\right)} = \frac{1}{e^{ce+o(1)}} = \Omega(1).\end{aligned}$$

□

We are now in position to prove the main result of this section, Theorem 6.

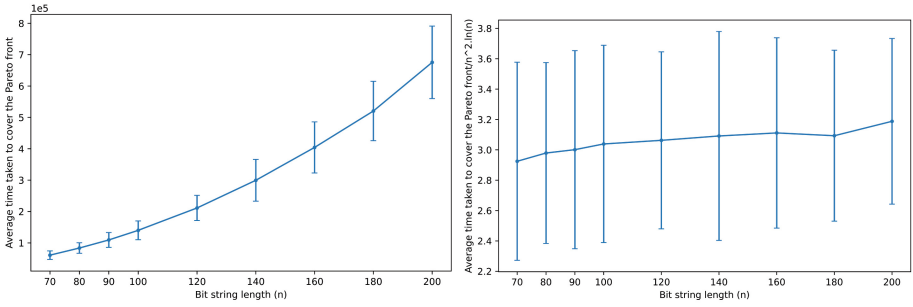
Proof (Proof of Theorem 6). By Lemma 8, assuming that with high probability its conditions are satisfied at some iteration t_0, with probability at least $\Omega(1)$, a run of GSEMO_2 is in a situation where there is some fitness value $(i, n-i)$ which is not present in the population, $\mathrm{OM}(x_t) > i + \varepsilon n$ and $\mathrm{OM}(y_t) < i - \varepsilon n$. Therefore, in all consequent iterations, to generate an individual with exactly i one-bits we need to flip at least εn bits in the parent (independently on which edge point we chose), the probability of which by the Chernoff bound is $e^{-\Omega(n)}$. Hence, the expected time until we cover the whole Pareto front is at least $e^{\Omega(n)}$. Since this happens with at least a constant probability, the total expected run time of the GSEMO_2 is also $e^{\Omega(n)}$, that is, it is super-polynomial.

8 Anti-aligned LFC

We use experimental results to extend our analyses to anti-aligned fitness functions from LFC. For the OMC, many individuals on the Pareto front have many neighbors on the Pareto front, so the exploration of the Pareto front is efficient. For anti-aligned fitness functions from LFC, we are only guaranteed a single neighbor (in each possible direction).

We start our analyses with two anti-aligned LFC functions without any shared bits. We assume, without loss of generality, that the optimum (minimum) of f_w is 1^n and that the weights are sorted based on the absolute value in descending order and thus the optimum of f_v is 0^n and the weights are sorted in ascending order. The lemma below is about how an individual on the Pareto-front looks like while optimizing anti-aligned *LFC* functions without any shared bits and the proof can be found in the supplementary material [1].

Proposition 9. *Let $w, v \in \mathbb{R}^n$ be such that w has only negative values, v has only positive values and the values are in ascending order. Then the Pareto dimension of the multi-objective function (f_w, f_v) is n and the Pareto front is $\{1^i 0^{n-i} \mid 0 \leq i \leq n\}$.*



(a) Average run time

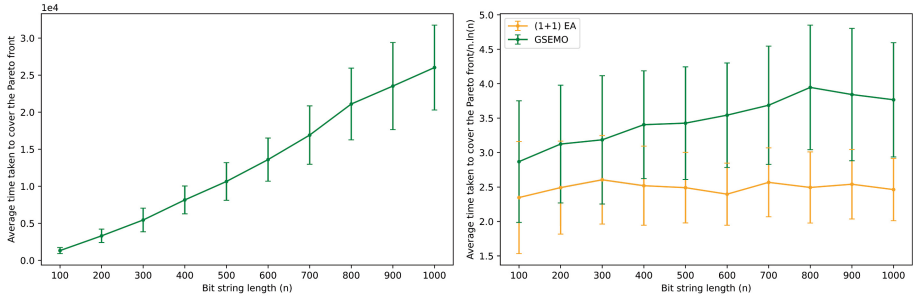
(b) Average run time normalized by $n^2 \cdot \ln(n)$

Fig. 1. Average run time of GSEMO on anti-aligned LFC functions with no shared bits.

We now empirically analyze the performance of GSEMO on minimizing two different types of anti-aligned LFC functions. First, we look at (f_w, f_v), where f_w, f_v are two anti-aligned LFC functions without any shared bits. That is, the optimum of f_w and the optimum of f_v differ in each bit position. Let n be the length of the bit string. The values of the weight vectors w and v are randomly chosen from $(-1, 0)$ and $(0, 1)$, respectively and are sorted in ascending order. We consider the mean of 100 independent runs of GSEMO on (f_w, f_v). In Fig. 1(a), for each n, we have the mean and the standard deviation of total number of

iterations required by the each run of GSEMO to cover the whole Pareto front and in Fig. 1(b) this value is divided by $n^2 \cdot \ln(n)$. We can observe from the Fig. 1(b), since the mean of the run time is almost a constant line, that GSEMO minimizing two anti-aligned LFC functions without any shared bits appears to have an expected run time of $O(n^2 \cdot \log(n))$.

We next look at (f_w, f_v), where f_w, f_v are two anti-aligned LFC functions without any conflicted bits. That is, f_w and f_v have the same global optimum. The values of the weight vectors w and v are randomly chosen from $(0, 1)$. Without loss of generality, let the weights be positive and the weight vector w be sorted in descending order and the weight vector v be sorted in ascending order. Note that the Pareto front is $\{0^n\}$. In the case of SEMO, the optimization process is similar to the random local search algorithm optimizing the ZeroMax function. Since in each iteration only one bit is flipped, the offspring gets rejected if a 0 bit is flipped to 1 and the offspring replaces the parent if a 1 bit is flipped to 0. This guarantees that SEMO has exactly one individual in the population at each iteration. However, in the case of GSEMO, the population size could be more than one, and the individuals in the population need not have the same number of 0s. We are interested in whether these possibilities slow down search compared with the run time required for the single objective optimization of any of these fitness functions.



(a) Average run time

(b) Average run time of GSEMO and (1+1) EA on f_w normalized by $n \cdot \ln(n)$

Fig. 2. Average run time of GSEMO on anti-aligned LFC functions with no conflicted bits.

We empirically analyze the performance of GSEMO minimizing (f_w, f_v) by considering the mean of 100 independent runs. In Fig. 2(a), for each n, we have the mean and standard deviation of total number of iterations required by each run of GSEMO to cover the whole Pareto front and in Fig. 2(b) this value is divided by $n \cdot \ln(n)$. In Fig. 2(b), we also have the average run time of the (1+1) EA on minimizing f_w, one of the objectives of the two objectives considered for GSEMO. We can observe from the Fig. 2(b), that GSEMO on two anti-aligned LFC functions without any conflicted bits appears to have an expected run time

of $O(n \cdot \log(n))$, while being a constant factor slower than just on one of the two functions.

References

1. Antipov, D., Kötzing, T., Radhakrishnan, A.: Supplementary material - greedy versus curious parent selection for multi-objective evolutionary algorithms (2024). https://zenodo.org/records/10990807
2. Antipov, D., Neumann, A., Neumann, F.: Rigorous runtime analysis of diversity optimization with GSEMO on OneMinMax. In: Foundations of Genetic Algorithms (FOGA 2023), pp. 3–14. ACM (2023)
3. Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: Do additional objectives make a problem harder? In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation GECCO 2007, pp. 765–772. ACM (2007)
4. Covantes Osuna, E., Gao, W., Neumann, F., Sudholt, D.: Speeding up evolutionary multi-objective optimisation through diversity-based parent selection. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2017), pp. 553–560. ACM (2017)
5. Covantes Osuna, E., Gao, W., Neumann, F., Sudholt, D.: Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. Theor. Comput. Sci. **832**, 123–142 (2018)
6. Doerr, B.: Probabilistic tools for the analysis of randomized optimization heuristics. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation - Recent Developments in Discrete Optimization, pp. 1–87. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_1
7. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theoret. Comput. Sci. **567**, 87–104 (2015)
8. Doerr, B., Gao, W., Neumann, F.: Runtime analysis of evolutionary diversity maximization for oneminmax. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO 2016), pp. 557–564. ACM (2016)
9. Doerr, B., Kötzing, T.: Lower bounds from fitness levels made easy. Algorithmica **86**(2), 367–395 (2024)
10. Doerr, B., Qu, Z.: A first runtime analysis of the NSGA-II on a multimodal problem. IEEE Trans. Evol. Comput. **27**(5), 1288–1297 (2023)
11. Doerr, B., Qu, Z.: Runtime analysis for the NSGA-II: provable speed-ups from crossover. In: Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023), pp. 12399–12407. AAAI Press (2023)
12. Doerr, B., Zheng, W.: Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives: (hot-off-the-press track at GECCO 2021). In: Proceedings of the 2021 Annual Conference on Genetic and Evolutionary Computation GECCO 2021, pp. 25–26. ACM (2021)
13. Doerr, C., Lengler, J.: Onemax in black-box models with several restrictions. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO 2015), pp. 1431–1438. ACM (2015)
14. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theor. Comput. Sci. **276**(1–2), 51–81 (2002)
15. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Theory Comput. Syst. **39**(4), 525–544 (2006)

16. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, 2nd edn. Springer, Heidelberg (2015)
17. Friedrich, T., Hebbinghaus, N., Neumann, F.: Plateaus can be harder in multi-objective optimization. In: 2007 IEEE Congress on Evolutionary Computation CEC, pp. 2622–2629 (2007)
18. Friedrich, T., Horoba, C., Neumann, F.: Illustration of fairness in evolutionary multi-objective optimization. Theor. Comput. Sci. **412**(17), 1546–1556 (2011)
19. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: The 2003 Congress on Evolutionary Computation CEC, vol. 3, pp. 1918–1925 (2003)
20. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimization. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation GECCO 2006, pp. 651–658. ACM (2006)
21. Kötzing, T., Krejca, M.S.: First-hitting times under drift. Theoret. Comput. Sci. **796**, 51–69 (2019)
22. 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)
23. Laumanns, M., Thiele, L., Zitzler, E., Welzl, E., Deb, K.: Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature PPSN 2002, pp. 44–53. Springer (2002)
24. Li, Y.L., Zhou, Y.R., Zhan, Z.H., Zhang, J.: A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **20**(4), 563–576 (2016)
25. Qian, C., Tang, K., Zhou, Z.H.: Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization. In: Parallel Problem Solving from Nature PPSN 2016. Springer (2016)
26. Qian, C., Yu, Y., Zhou, Z.H.: An analysis on recombination in multi-objective evolutionary optimization. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO 2011), pp. 2051–2058. ACM (2011)
27. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Comb. Probab. Comput. **22**(2), 294–318 (2013)
28. Zheng, W., Doerr, B.: Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). Artif. Intell. **325**, 104016 (2023)

How Population Diversity Influences the Efficiency of Crossover

Sacha Cerf[1(✉)] and Johannes Lengler[2]

[1] Ecole Polytechnique, Paris, France
sacha.cerf@inria.fr
[2] ETH Zürich, Zürich, Switzerland

Abstract. Our theoretical understanding of crossover is limited by our ability to analyze how population diversity evolves. In this study, we provide one of the first rigorous analyses of population diversity and optimization time in a setting where large diversity and large population sizes are required to speed up progress. We give a formal and general criterion which amount of diversity is necessary and sufficient to speed up the $(\mu+1)$ Genetic Algorithm on LeadingOnes. We show that the naturally evolving diversity falls short of giving a substantial speed-up for any $\mu = O(\sqrt{n}/\log^2 n)$. On the other hand, we show that even for $\mu = 2$, if we simply break ties in favor of diversity then this increases diversity so much that optimization is accelerated by a constant factor.[3](Proofs in this submission are mostly omitted due to the page limit. A full version with detailed proofs can be found in the arXiv version of this article [2], but reviewers are not required to consult that version or to check correctness of those proofs.)

1 Introduction

One of the central aspects of genetic algorithms (GAs) is their ability to recombine existing solutions via crossover. This is considered crucial and important in practical applications [30]. In order for crossover to be helpful, it is vital that the population remains diverse, which gives a very specific setting for the exploration/exploitation dualism. Unfortunately, our ability to mathematically analyze population diversity and its impact on runtime has been limited to situations of small populations and/or small diversity, as we will review below. To already give one example, in easy hillclimbing settings like OneMax[1], a tiny Hamming distance of 2 between two parents of equal fitness is already beneficial for crossover. In such situations, crossover has been proven to be helpful [28].

In this paper, we will treat a situation that was not amenable for analysis with previous techniques, because crossover will only be beneficial if the population diversity is quite large. More precisely, we will study the LeadingOnes function $\mathrm{LO}(x)$, which returns for $x \in \{0,1\}^n$ the number of one-bits before the first zero-bit in x, see Sect. 2 for the formal definition. For a string x with $\mathrm{LO}(x) = k$, in order to improve its fitness it is necessary to flip the $(k+1)$st bit in x. Thus, it is rather hard to find such an improvement. OneMax and LeadingOnes are the most common theoretical benchmarks for

[1] For $x \in \{0,1\}^n$, the OneMax function is defined via $f(x) = \sum_{i=1}^{n} x_i$.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 102–116, 2024.
https://doi.org/10.1007/978-3-031-70071-2_7

hillclimbing in discrete search spaces, where ONEMAX is supposed to be particularly easy[2] and LEADINGONES is designed to be particularly hard [26]. By construction of LEADINGONES, a crossover between two bit-strings x and y, where $\mathrm{LO}(x) = k$, can only be fitter than x if the two parents differ specifically in the $(k+1)$st position. This is a quite strong requirement and this situation will usually only occur if the population is very diverse. In contrast, on ONEMAX an offspring of x and y can be fitter than x if the parents differ in *any* position where x has a zero-bit, which happens even with minimal population diversity. The main contribution of this paper is that we develop a method to track the population diversity even if it is large, and that we give a general criterion to translate population diversity into runtime[3] for the $(\mu + 1)$ GA on LEADINGONES.

1.1 Our Results

We analyze the runtime of the elitist $(\mu + 1)$ Genetic Algorithm (or $(\mu + 1)$ GA for short) on LEADINGONES. We use standard options for mutation and crossover operators: standard bit mutation with rate χ/n for some constant $\chi > 0$ and uniform crossover with uniform parent selection, see Sect. 2 for details. It was known before that without crossover, the expected runtime of the $(\mu + 1)$ GA on LEADINGONES is $(1 + o(1))\frac{e^\chi - 1}{2\chi^2} n^2$ for all $\mu = o(n/\log n)$ [1,31].

Our first result is that this runtime stays the same for the $(\mu + 1)$ GA for any $\mu = O(\sqrt{n}/\log^2 n)$, up to a $(1 + o(1))$ factor.[4] The core contribution of the proof lies in showing that the population diversity, measured by the average Hamming distance of two randomly selected parents, is bounded by $O(\mu)$. We show in a general setting that this diversity is too small to speed up the runtime by any constant factor. Our technique builds on a recent result by Jorritsma, Lengler and Sudholt [16], who analyzed how population diversity of the $(\mu+1)$ GA evolves in the absence of selective pressure, i.e., for a flat fitness function. Hence, for moderately large population sizes, the $(\mu+1)$ GA lacks population diversity.

Our second result shows that this problem can be overcome easily by simply breaking ties between equally fit individuals in favor of diversity, then even for $\mu = 2$ the average Hamming distance increases to $\Omega(n)$. This speeds up optimization by a constant factor.

Intuitive Explanation of the Results

Preparation: Runtime Without Crossover. Let us first recapitulate where the runtime for $\mu = 1$ comes from (without crossover, as this does not make sense for $\mu = 1$). When the current search point x has fitness $\mathrm{LO}(x) = k$, then for an improvement it is necessary to flip the $(k + 1)$st bit of x, which happens with probability χ/n. The expected time until this happens is n/χ. There is a second condition: the bits $1, \ldots, k$ must not be flipped. It can be shown that this second condition leads to an aggregated

[2] In fact, it can be mathematically proven that ONEMAX is the easiest problem with unique optimum for many algorithms [9,16,27,32].

[3] We measure the runtime as the number of function evaluations until the optimum is evaluated.

[4] For ease of terminology we will ignore $(1 + o(1))$ factors in the rest of this exposition.

factor of $(e^\chi - 1)/\chi$. This is not completely obvious, but is also not hard with the modern tools of *drift analysis* that have been developed in the last decade [20].

The two aforementioned conditions for fitness improvement would lead to a runtime of $(1 + o(1))\frac{e^\chi - 1}{\chi^2}n^2$ if it was necessary to visit all n fitness levels, but this is not necessary. When the $(k+1)$st bit is flipped, then it may happen by chance that the $(k+2)$nd bit is already set to one, in which case the algorithm will skip fitness level $k+1$. This happens with probability $1/2$, and in this case the $(k+2)$nd bit is called a *free-rider*. There can be more than one free-rider at once, and the number of free-riders is well-understood: in expectation only every second fitness level is visited, which reduces the runtime by a factor of 2, and leads to the overall runtime of $(1 + o(1))\frac{e^\chi - 1}{2\chi^2}n^2$.

Without crossover, the above explanation remains essentially unchanged for larger μ, up to $\mu = o(n/\log n)$: once the first individual reaches fitness level k, it only takes time $\Theta(\mu \log \mu) = o(n)$ until all individuals are on this level. This time is negligible compared to the time that is needed for the next improvement. Once all individuals have reached fitness k, all parents have the same chance to produce an offspring of larger fitness, so the effect of the larger population size is negligible. The discussion up to this point was known from previous work.

Extra Free-Riders Through Crossover. Our main insight lies in the following. With crossover there is an additional chance to make progress. Consider the situation that the whole population is at fitness level k, and an offspring x reaches a new fitness level for the first time. Assume for simplicity that there are no free-riders in this step, so $\mathrm{LO}(x) = k+1$. Then x has a one-bit at position $k+1$ and a zero-bit at position $k+2$. All other individuals have a zero-bit at position $k+1$ because they all have fitness k. But it is possible that some other individual y has a one-bit at position $k+2$. If x and y perform a crossover, then there is a chance of $1/4$ that it gets the one-bit at position $k+1$ from x, and the one-bit at position $k+2$ from y, combining the best from the two parents. This effectively gives an *extra free-rider*. If this scenario happens for a constant fraction of all levels, this reduces the runtime by a constant factor.

There are two key question for the runtime analysis:

1. Conditional on $\mathrm{LO}(x) = k+1$, how likely is it that there is an individual y in the population with a one-bit in position $k+2$?
2. If there exists such y, how likely is it that y transfers its gene to x before it is replaced by individuals of higher fitness?

The answer to the second question is more positive than might seem on first glance, because in each generation the probability that y passes on its gene is only $O(1/\mu)$. However, in order to replace the old population by fitter individuals, the algorithm needs some time: it must select x or its equally fit descendants at least μ times. (Here we omit the unlikely case that the level is reached a second time by mutation.) Intuitively, this corresponds to μ chances to select y as the second parent and perform the gene transfer, each with probability $1/\mu$. The real situation is more complex since y could be replaced earlier, but it suffices if the gene continues to exist in the population until half of the population has reached fitness at least $k+1$. In this case, it already has a chance of $\Omega(1)$ to be passed on in form of an extra free-rider. We will not need this argument

directly for the proofs, but believe that it provides the right intuition: genes that exist are efficiently transferred into extra free-riders.

Connection to Diversity. For point 1, recall that $\mathrm{LO}(x) = k + 1$ means in particular that x has a zero-bit at position $k + 2$. This is key to the situation: we want to obtain a one-bit in a specific position where x can not provide the one-bit by itself. Thus, the probability to understand is: how likely is it that the bit value of y differs from the bit value of x in position $k+2$? This is closely connected to the Hamming distance between x and y and thus, to the diversity. In fact, the LEADINGONES function has a high level of symmetry, and the bits $k + 2, \dots, n$ do not have any effect on the fitness before the creation of x. Hence, if x and y have Hamming distance d, then the bits in which they differ are uniformly at random among $k + 2, \dots, n$ (plus the two special position k and $k + 1$). Thus, we can compute the probability that x and y differ in position $k + 2$ from their Hamming distance, which is directly connected to the population diversity.

Let us quantify the effect in terms of μ. Once a new fitness level is reached, the old population is replaced, which represents a genetic bottleneck that reduces diversity. Afterwards, the average Hamming distance starts growing again. If given enough time, it will grow until it reaches $\Theta(\mu)$, at which point it maxes out because diversity may also get lost again whenever individuals are removed from the population. These *equilibrium dynamics* were recently discovered and quantified in [16]. For $\mu = o(\sqrt{n})$, this means that the average Hamming distance stays at $\Theta(\mu)$, and the probability that a fixed individual y differs in position $k+2$ from x is only $O(\mu/n)$. By a union bound over all μ individuals, the probability that the desired one-bit exists somewhere in the population is at most $O(\mu^2/n) = o(1)$. Since this one-bit typically does not exist, crossover has no chance of providing an extra free-rider. We prove this formally, where for technical reasons we make the slightly stronger assumption $\mu = o(\sqrt{n}/\log^2 n)$.

If we modify the $(\mu + 1)$ GA to break fitness ties in favor of larger diversity, then the equilibrium dynamics changes. We show for $\mu = 2$ that the equilibrium shifts from $\Theta(1)$ to $\Theta(n)$. Moreover, the time required to reach diversity $\Theta(n)$ is only $O(n)$. This is fast enough to give an expected constant number of extra free-riders per fitness level, which leads to a constant factor speed-up.

Although we do not examine the case in this paper, let us briefly speculate on the case $\mu = \Omega(\sqrt{n}) \cap o(n/\log n)$, without diversity-increasing tie-breaker. This may look promising since the aforementioned equilibrium dynamics remain true: the average Hamming distance is $\Theta(\mu)$, so it seems conceivable that point 1 from above has a high probability. However, we conjecture that this is not the case, and that the probability is $o(1)$, because the diversity is generated by $o(n)$ positions who differ in many pairs, while $n - o(n)$ positions are identical throughout the population. Nevertheless, we believe that this setting is worth exploring, since a mechanism for increasing diversity in this regime could potentially lead to runtime $o(n^2)$. We leave the exploration of this regime to future work.

1.2 Related Work

There is a very long history of theoretical work on crossover, and we only give a brief overview. A thorough overview of the theoretical study of population diversity is the

review by Sudholt [29]. For the more specific question how diversity can provably decrease runtime, a more detailed discussion can be found in [7].

Our result on LEADINGONES is by far not the first setting in which crossover is provably beneficial. Historically one of the first rigorous mathematical results were for functions that were specifically tailored to make crossover beneficial, such as the REALROYALROAD function [15]. A non-tailored example is the ONEMAX function mentioned above. Sudholt [28] proved that crossover accelerates a non-standard version of the $(2+1)$ GA by a constant factor on ONEMAX, and Corus and Oliveto [3] showed that a constant factor speedup is also obtained for the standard $(2+1)$ GA. However, their analyses rely on the fact that crossover between *any* two different search points is helpful for ONEMAX. So it sufficed to show that the diversity is not literally zero. Experiments in [3] indicated that larger population sizes than 2 might be helpful, but so far it could not be mathematically shown that higher population sizes $\mu = \omega(1)$ (or even $\mu > 2$) leads to substantially larger diversity that speeds up optimization on ONEMAX.

Another important benchmark problem is the JUMP function, where the optimum is surrounded by a fitness valley of size k. There has been a long and rich line of research for crossover on this function, particularly on the $(\mu+1)$ GA and some variations [4,7,14,17,22,25]. It had been understood early that mutation can increase diversity substantially [17], but it remained unclear how crossover influences the population dynamics. Hence, polynomial runtime bounds independent of k (for constant k) could only be shown if crossover happens so rarely that it does not influence the dynamics of population diversity [14,17], or if the process is amended with diversity-enhancing mechanisms [4]. Without such mechanisms and for larger crossover probabilities, analyses were for a long time limited to minimal amounts of diversity [7,25]. Even so recent results as the work by Doerr, Echarghaoui, Jamal and Krejca from 2023 [7] could only make use of Hamming distances of at least one, i.e., the proof relied on showing that crossover is frequently performed between two individuals which are not identical to each other. However, very recently Lengler, Opris and Sudholt [22] could show a tight bound by proving that the typical Hamming distances are $2k$, which is the maximal possible Hamming distance on the plateau of local optima. However, they could only show their result for a modified version of the $(\mu+1)$ GA in which the parents produce several offspring at the same time, and proceed with the fittest. Nevertheless, the result was the first to show analytically high amounts of diversity on JUMP in a setting with frequent crossover-based and without diversity-enhancing mechanism. Notably, the result in [22] built on the same techniques from 2023 in [21] that we also build upon.

Other theoretical work has shown benefits of problem-specific crossover operators [8,24], of special ways of applying crossover as in the successful design of the $(1+(\lambda,\lambda))$ GA [6], and of crossover that is enhanced by diversity-preserving mechanisms [4,19,23]. A discussion of those and further results can be found in [7] and [29].

2 Preliminaries

In this section, we formally introduce the optimization problem and algorithm studied in the paper. Then, we introduce the notations that will be used in our analysis. We also

Algorithm 1: The $(\mu + 1)$-GA for maximizing a fitness function f.

$t \leftarrow 0$; Generate initial population $P_0 \in (\{0,1\}^n)^\mu$.
repeat
  With probability p_c, choose a random parent pair of parents in P_t which do not have the same index and generate the offspring y via crossover. Otherwise, choose a random parent $x \in P_t$ and copy it to get y.
  Apply mutation on y to get y'.
  Choose z uniform at random among the individuals in P_t with minimal fitness.
  if $f(y') > f(z)$ **then**
    $P_{t+1} \leftarrow P_t \backslash \{z\} \cup \{y'\}$.
  if $f(y') = f(z)$ *and tie-breaker decides for* y' **then**
    $P_{t+1} \leftarrow P_t \backslash \{z\} \cup \{y'\}$.
  $t \leftarrow t + 1$
until forever;

explain the concept of unbiased operators from [18] since some of our results hold for arbitrary unbiased mutation and crossover operators.

LeadingOnes. Let $n \in \mathbb{N}$. The LEADINGONES fitness of a bit-string $x \in \{0,1\}^n$ is the number of consecutive ones from the left of x,

$$\mathrm{LO}(x) = \textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j = \max\{1 \leq i \leq n \mid \forall 1 \leq j \leq i : x_j = 1\}.$$

We will sometimes write f instead of LEADINGONES for the sake of conciseness.

The Algorithm. The $(\mu + 1)$ Genetic Algorithm, or $(\mu + 1)$ GA for short, is described in Algorithm 1 for arbitrary mutation and (binary) crossover operators and for arbitrary tie-breaking rules. Its *runtime* $T^\mu = T^\mu_n$ is the number of function evaluations on LEADINGONES before the optimum is found. Our main result will use *standard bit-mutation* and *uniform crossover*: standard bit-wise mutation with mutation rate χ flips every bit of a bit-string independently with probability χ/n, and uniform crossover consists in taking each bit from one of the two parents, with equal probability and independently from each other. Moreover, we will always break ties in favor of the offspring except for Sect. 3.2, where we explicitly study a variant of the $(2+1)$ GA which uses a diversity-increasing tie-breaking mechanism. Some of our results, in particular in Sect. 2.1 are true for more general mutation and crossover operators and tie-breakers.

General Notation. We consider the LEADINGONES function on the search space $\{0,1\}^n$ for $n \to \infty$, and all Landau notation like $O(.), \Omega(.), \ldots$ is with respect to this limit. We denote search points by $x = (x_1, ..., x_n)$. For any two search points x, y, the *Hamming distance* $H(x, y)$ of x and y is the number of positions $1 \leq i \leq n$ such that $x_i \neq y_i$. For $x \in \{0,1\}^n$, we define $0_x = \{1 \leq i \leq n \mid x_i = 0\}$, and $1_x = \{1 \leq i \leq n \mid x_i = 1\}$. For $S \subseteq \{1, \ldots, n\}$, we define $x_S = (x_i)_{i \in S}$. For the special case where S is an integer interval with lower and upper bounds m and M, we may write $x_{[m:M]}$, or even $x_{[m:]}$ if $M = n$. Finally, for two random variables X and Y,

we write $X \preceq Y$ for "X is stochastically dominated by Y". We use the same notation if Y is a probability distribution. We write $\mathcal{G}(p)$ for the geometric distribution with mean $1/p$.

$(\mu+1)$GA Process. We denote by χ the expected number of bits flipped by any mutation operator, and assume $\chi = \Theta(1)$ throughout the paper. The best fitness of an individual in the t-th population $P_t = \{x_1^t, \dots, x_\mu^t\}$ is denoted $\mathrm{LO}_t = \max\{\mathrm{LO}(x), x \in P_t\}$. We call C_t the event "a crossover was performed at iteration t", and M_t^i the event "bit i is mutated at time t".

For $0 \leq i \leq n$, we will use the expression "level i is *visited*" to signify "$\exists t \in \mathbb{N}, \mathrm{LO}_t = i$". We denote by T_i^{in} the time of reaching fitness level at least i, by T_i^{o} the time of exceeding fitness i, and by Succ_i the fitness level at time $T_i^{\mathrm{o}} + 1$. We say that a population P is *consolidated* if all individuals in P have the same fitness i. We call the *consolidation time* for fitness level i the time from reaching this level until a consolidated population on this fitness level occurs. Note that there may be no consolidated population on fitness level i, in which case we set $T_i^c = T_i^o$. As in [12], "fitness level i is *essential*" means that it is visited, and left by mutation. We call this event E_i. If an essential fitness level i is left before *consolidation* ($T_i^c > T_i^{\mathrm{o}}$), it is called "strange", which is denoted S_i. An essential fitness level i which is not strange is called "normal", which is denoted by N_i. We also denote ESucc_i the smallest essential fitness level after i, i.e. $\mathrm{ESucc}_i := \min\{j > i \mid E_j\}$. We set $\mathrm{ESucc}_i := n+1$ if there is no $j > i$ with E_j.

Diversity Measure. Following [21], we define for a population P the sum of pairwise Hamming distances in the population $S(P) = \sum_{x \in P} \sum_{y \in P} H(x,y)$, and for $x \in \{0,1\}^n$, the sum $S_P(x) = \sum_{y \in P} H(x,y)$ of Hamming distances between x and all individuals in P. Note that the average Hamming distance between two individuals (without repetition) is $\frac{S_P(x)}{\mu(\mu-1)}$. When it is clear, we will omit the index of the population P we are summing over: $S_P(x) = S(x)$. Finally, we call "diversity of the population at time t", the quantity $d_t = \frac{S(P'_t)}{\mu(\mu-1)(n-\mathrm{LO}_t-1)}$, where $P'_t = \{x_{[\mathrm{LO}_t+2:]}, x \in P_t\}$. This is the average pairwise Hamming distance of the non-optimized parts of the bit-strings in the population (not counting the bit just after the current fitness level), normalized by the size of the non-optimized part of a fit bit-string.

2.1 Unbiased Offspring Generation Mechanisms

Our analysis builds on the fact that, for a given individual in the population, the bits between $\mathrm{LO}_t + 2$ and n are uniformly distributed in the space of bit-strings of size $n - \mathrm{LO}_t - 1$, and that the bits in which two individuals differ are evenly distributed in this range. As we will show below, this is generally true if mutation, crossover, and tie-breaker are *unbiased* operators. The notion of unbiased operators has been introduced in [18] as operators which are invariant under automorphisms of the hypercube.

The unbiased framework has been very successful especially in the context of black-box complexity [11]. Most of the standard mutation operators, such as standard bit-wise mutation or the heavy-tailed mutation used in *fast GAs* [10] are unbiased. Many

crossover operators are unbiased, like uniform crossover, but some, like the single-point crossover, are not, see [13] for details. Our default in this paper are standard bit-wise mutation and uniform crossover, both of which are unbiased. Note that a tie-breaker can be considered as a $(\mu+2)$-ary operator, taking as an input the full population of size μ and two additional search points between which we want to break ties, one of which it needs to return.

For the $(\mu+1)$ GA, we call the combination of crossover (if applied), mutation and tie-breaking as *offspring generation mechanism*, and we call such a mechanism unbiased if all three operators that constitute it are unbiased. We call the algorithm $(\mu+1)$ GA unbiased if its offspring generation mechanism is unbiased and if it is initialized with a 0-ary unbiased operator. We may now prove some useful results that are true for an unbiased $(\mu+1)$ GA on the LEADINGONES problem. Define, for every automorphism of the hypercube π, and every population P, $\pi(P) = \{\pi(x), x \in P\}$, and $F(P) = \max\{\mathrm{LO}(y), y \in P\}$. Our first result states that the population P_t is invariant under automorphisms that keep the first $F(P_t)+1$ bits fixed. We omit the proof.

Lemma 1. *Consider an unbiased $(\mu+1)$ GA. For all $t \geq 0$, all $P \in \{0,1\}^\mu$ and all automorphism π of the hypercube such that for all bit-string x, and all $j \leq F(P)+1$, $\pi(x)_j = x_j$,*

$$\Pr(P_t = P) = \Pr(P_t = \pi(P)).$$

Lemma 1 has some interesting consequences, the first one being on the distribution of the non-optimized part of a single bit-string in the population.

Corollary 2. *For all $1 \leq j \leq \mu$, $x^t_{j_{[\mathrm{LO}_t+2:]}}$ is uniformly distributed on $\{0,1\}^{n-\mathrm{LO}_t-1}$.*

We also get the following very useful result that helps us bound the size of the fitness jumps.

Corollary 3. *For all $t \geq 0$, for all $2 \leq j$, $\Pr(\mathrm{LO}_{t+1} - \mathrm{LO}_t \geq j \mid \mathrm{LO}_{t+1} - \mathrm{LO}_t \geq 1) = 2^{-(j-1)}$.*

Finally we give a result on the distribution of the bits that differ between two individuals in the population at a given iteration. Recall that d_t is the average density of non-equal bits in the population when we restrict to the non-optimized part of the bit-string $[\mathrm{LO}_t+1:n]$.

Corollary 4. *Let $t \geq 0$, and $1 \leq i, i' \leq \mu$ with $i \neq i'$. Let $d := x^t_{i_{[\mathrm{LO}_t+2:]}} \oplus x^t_{i'_{[\mathrm{LO}_t+2:]}}$ be the string which has a 1 where x^t_i and $x^t_{i'}$ differ, and a 0 otherwise, truncated to the non-optimized part. Then, for any pair of bit-strings $s, s' \in \{0,1\}^{n-\mathrm{LO}_t-1}$ of equal Hamming weight, we have $\Pr(d=s) = \Pr(d=s')$. In particular, for any $j \in [\mathrm{LO}_t+2:n]$ and for all $H \geq 0$,*

$$\Pr\left((x^t_i)_j \neq (x^t_{i'})_j \mid H(x^t_{i_{[\mathrm{LO}_t+2:]}}, x^t_{i'_{[\mathrm{LO}_t+2:]}}) = H\right) = \tfrac{H}{n-\mathrm{LO}_t-1}.$$

Moreover, if the tie-breaker is symmetric with respect to permutations of the population and the initial population is uniformly random, then

$$\Pr((x^t_i)_j \neq (x^t_{i'})_j \mid d_t) = d_t.$$

Preliminary Results on the Consolidation Process. The following lemma bounds the consolidation time of raising the whole population to fitness at least i, after this fitness level has been found. This time is well-known to have expectation $O(\mu \log \mu)$, e.g. [31]. Here we provide a tail bound. Note that the crossover probability p_c and the parameter p_{clone} that appears in the following lemma are not included into the index of C_β because those are part of the algorithm, which we consider as fixed.

Lemma 5. *Consider the $(\mu+1)$ GA with a mutation operator that has probability p_{clone} of duplicating the parent. Let $1 \leq i \leq n$ be any fitness level on* LEADINGONES. *For $t \geq 0$, let X_t be the number of individuals of fitness at least i in $P_{T_i^{in}+t}$. Then, for any constant $\beta > 0$, $p_{clone} > 0$ and $p_c < 1$, there exists $C_\beta > 0$ such that for n big enough the following holds for all $C > C_\beta$.*

$$\Pr(X_{C\mu \log \mu} < \mu) \leq \mu^{-\beta}. \tag{1}$$

In particular, for any fitness level $1 \leq i \leq n$, $\mathbb{E}[T_i^c - T_i^{\text{in}}] = O(\mu \log \mu)$ and

$$\Pr(T_i^c - T_i^{\text{in}} > C\mu \log \mu) \leq \mu^{-\beta}.$$

3 Analysis of the $(\mu + 1)$ GA on LEADINGONES for Different Population Regimes

In this section, we will show that the number of extra free-riders determines the expected runtime of the $(\mu + 1)$ GA. Throughout the section, we will assume standard bit mutation where the mutation rate χ and the crossover probability $p_c < 1$ are constants. However, at first the algorithm may use any respectful[5] unbiased crossover operator and any unbiased tie-breaking rule.

Our strategy is based on the notion of *extra free-riders* due to crossover, as introduced in Sect. 2. This term is derived from the term "free-rider" originating in [12], which we define as $F_i := \text{Succ}_i - i - 1$ if the i-th fitness level is essential, and as $F_i := 0$ otherwise. In other words, a free-rider is a fitness level that is skipped due to a mutation. Extending on this idea, we define "extra free-riders" as additional leading ones that are obtained with crossover. Recall that E_i is the event that the i-th fitness level is essential, i.e., is left via mutation, and that ESucc_i is the smallest essential fitness level after level i.

Definition 6 (Extra free-riders). For $0 \leq i \leq n-1$, we denote as EF_i and call "extra free-riders associated to level i" the following random variable.

$$\text{EF}_i = \begin{cases} \text{ESucc}_i - \text{Succ}_i & \text{if } E_i \\ 0 & \text{otherwise} \end{cases}$$

[5] In a respectful crossover, if both parents have the same bit at some position i, the offspring also has the same bit at position i.

Note that if the crossover operator is *respectful*, which is the case for most common crossover operators (see [21] for a classification), then extra free-riders must come from one of the parent. Thus, just like normal free-riders, these 1-bits already *accidentally* exist among the population (that is, they are not here because of an optimization choice of the algorithm, but because of genetic drift), and allow us to overcome some fitness plateaus in negligible time.

The typical scenario for the acquisition of extra free-riders is that, after the whole population is brought to a common fitness plateau, diversity accumulates on the non-optimized trailing part of the bit-strings. Then, when an individual x reaches a higher fitness level, some individuals in the lower levels may happen to have a 1-bit at the position corresponding to the next fitness level. If a crossover between one of these individuals and an individual of fitness $f(x)$ is performed, then with a good probability we get extra free-riders. These extra free-riders can be obtained from multiple successive crossovers, until the next fitness level is left via mutation. We introduce a useful definition that stems from this observation.

The two following lemmas draw a link between the expected value of EF_i for any reached fitness level i in a given implementation of the $(\mu+1)$ GA, and the expected runtime of the algorithm. Recall that we consider the $(\mu+1)$ GA with standard bit-wise mutation with mutation rate $\chi = \Theta(1)$, $p_c < 1$, any respectful crossover operator, and any unbiased tie-breaker, but the proof may be adapted to most known mutation mechanisms.

Lemma 7. *Consider the $(\mu+1)$ GA with standard bit mutation and any respectful unbiased mutation operator. Suppose that there exists a sequence of functions $(m_n)_{n\in\mathbb{N}}$ defined on $[0,1]$, uniformly convergent to a function m, and $k(n) = \omega(1)$, such that $\varepsilon_{m_n} = \max\{|m_n(x) - m_n(y)| \mid |x-y| \leq \frac{1}{k}\} = o(1)$, and, for all fitness levels $0 \leq i \leq n-1$, $\Pr(\mathrm{EF}_i \geq 1 \mid N_i) \geq m_n(\frac{i}{n})$. Then:*

$$\mathbb{E}[T^\mu] \leq \frac{n^2}{\chi}\int_0^1 \frac{e^{\chi x}}{2+m(x)}dx + o(n^2).$$

Lemma 8. *Consider the $(\mu+1)$ GA with standard bit mutation and any respectful unbiased mutation operator. Suppose that there exists a sequence of functions $(M_n)_{n\in\mathbb{N}}$ defined on $[0,1]$, uniformly convergent to a function M, and $k(n) = \omega(1)$, such that $\varepsilon_{M_n} = \max\{|M_n(x) - M_n(y)| \mid |x-y| \leq \frac{1}{k}\} = o(1)$, and, for all fitness levels $0 \leq i \leq n-1$, $\mathbb{E}[\mathrm{EF}_i \mid N_i] \leq M_n(\frac{i}{n})$. Suppose also that the event A : "for all $1 \leq i \leq n, \mathrm{ESucc}_i - i = o(\frac{n}{\max(k,\mu\log\mu)})$" holds with high probability. Then:*

$$\mathbb{E}[T^\mu] \geq \frac{n^2+o(n^2)}{\chi}\int_0^1 \frac{e^{\chi x}}{2+M(x)}dx.$$

Observation 1 *Note that for the case where m_n and M_n are constant, the inequality from Lemmas 7 and 8 respectively become:*

$$\mathbb{E}[T^\mu] \leq \frac{1}{2+m_n}\frac{e^\chi-1}{\chi^2}n^2 + o(n^2) \quad \textit{and} \quad \mathbb{E}[T^\mu] \geq \frac{1}{2+M_n}\frac{e^\chi-1}{\chi^2}n^2 + o(n^2).$$

Observation 2 *When $p_c = 0$, one can simply set $m_n = M_n = 0$, and obtain $\mathbb{E}[T^1] = \frac{e^\chi-1}{2\chi^2}n^2 + o(n^2)$, which is up to $o(n^2)$ the runtime of the $(1+1)$ EA.*

These two lemmas are stated in a much stronger form than what we will use in this paper, where M_n or m_n are constant. This is because we believe they can be a useful tool for whoever would like to study other variants of the $(\mu+1)$ GA, where these two quantities might depend on the current fitness level.

To prove these lemmas, we will need two preparatory results. One will allow us neglect strange fitness levels, the second one to restrict to normal fitness level.

Lemma 9. *For any fitness level* $0 \le i \le n-1$, $\Pr(S_i) = O\big(\frac{\mu \log \mu}{n}\big)$.

Lemma 10. *For* $\mu = o(n/\log n)$ *the following equality holds:*

$$\mathbb{E}[T^\mu] = \sum_{i=0}^{n-1} \mathbb{E}[\mathbb{1}_{N_i}(T_i^{\mathrm{o}} - T_i^c)] + o(n^2).$$

With this preparation, we can now prove Lemma 7. We omit the proof of Lemma 8, since it mostly follows the same strategy, although a bit more work is needed to exclude exceedingly large jumps.

Proof (of Lemma 7). By Lemma 10 we may focus on normal levels. We observe that on any normal fitness level i, we have $T_i^{\mathrm{o}} - T_i^c \sim \mathcal{G}((1-\frac{\chi}{n})^i \frac{\chi}{n})$. Indeed, after consolidation, at each iteration, regardless of whether we use crossover or not (since the crossover operator is respectful, so the first i 1-bits are always copied to the offspring), the probability of producing an offspring with fitness level higher than i is determined only by the mutation phase, where we have to keep the first i bits untouched and mutate the $(i+1)$-st bit.

We also need to know how many normal levels there are in a local window of fitness levels. The more extra free-riders we get, the sparser normal fitness levels are, and the faster the optimization is. To make this precise, let us define, for $0 \le j \le k$, $i_j = \frac{jn}{k}$. Then, for $i_j \le i < i_{j+1}$, define the truncated number of extra free-riders associated to fitness level i as

$$\widetilde{\mathrm{EF}}_i := \min(i_{j+1}, \mathrm{ESucc}_i) - \mathrm{Succ}_i.$$

First, we claim that for all $0 \le j \le k-1$:

$$\sum_{i=i_j}^{i_{j+1}-1} \mathbb{1}_{N_i}(1 + F_i + \widetilde{\mathrm{EF}}_i) \le \sum_{i=i_j}^{i_{j+1}-1} \mathbb{1}_{E_i}(1 + F_i + \widetilde{\mathrm{EF}}_i) \le \tfrac{n}{k}. \tag{2}$$

The first inequality stems from $N_i \subset E_i$. The second holds because the sum in the middle is simply $i_{j+1} - L_j$, where L_j is the first essential level in $[i_j, i_{j+1}]$, if such a level exists. In particular the sum is at most $i_{j+1} - i_j = \frac{n}{k}$. If no essential level exists then the sum is zero and the bound is still true.

Note that $\mathbb{E}[\widetilde{\mathrm{EF}}_i \mid N_i] \ge \Pr(\mathrm{EF}_i \ge 1 \mid N_i) \ge m_n(\frac{i}{n})$. Using $(F_i \mid N_i) \preceq \mathcal{G}(\frac{1}{2})-1$ (which is a direct implication of Corollary 3), this yields for all level i:

$$\mathbb{E}[\mathbb{1}_{N_i}(1 + F_i + \widetilde{\mathrm{EF}}_i)] \ge \mathbb{E}[\mathbb{1}_{N_i}](2 + m_n(\tfrac{i}{n})).$$

Hence, by definition of ε_{m_n}, for $i_j \le i \le i_{j+1} - 1$,

$$\mathbb{E}[\mathbb{1}_{N_i}(1 + F_i + \mathrm{EF}_i)] \ge \mathbb{E}[\mathbb{1}_{N_i}](2 + m_n(\tfrac{i_j}{n}) - \varepsilon_{m_n}) = \mathbb{E}[\mathbb{1}_{N_i}](2 + m_n(\tfrac{j}{k}) - \varepsilon_{m_n}).$$

Plugging this last inequality into the left part of (2), we get the desired bound for the expected number of essential levels in the local window $i_j \leq i < i_{j+1}$.

$$\sum_{i=i_j}^{i_{j+1}-1} \mathbb{E}[\mathbb{1}_{N_i}] \leq \tfrac{n}{k(2+m_n(j/k)-\varepsilon_{m_n})}$$

By Lemma 10 and since $T_i^o - T_i^c \sim \mathcal{G}((1-\frac{\chi}{n})^i \frac{\chi}{n})$, it suffices to bound $\sum_{i=0}^{n-1} \mathbb{E}[\mathbb{1}_{N_i} \mathcal{G}((1-\frac{\chi}{n})^i \frac{\chi}{n})]$. Decomposing this sum into sums over the local windows yields:

$$\sum_{j=0}^{k-1} \sum_{i=i_j}^{i_{j+1}-1} \mathbb{E}[\mathbb{1}_{N_i} \mathcal{G}((1-\tfrac{\chi}{n})^i \tfrac{\chi}{n})] \leq \sum_{j=0}^{k-1} \sum_{i=i_j}^{i_{j+1}-1} \mathbb{E}[\mathbb{1}_{N_i}](1-\tfrac{\chi}{n})^{-i} \tfrac{n}{\chi}$$

$$\leq \sum_{j=0}^{k-1} \frac{n}{k(2+m_n(j/k)-\varepsilon_{m_n})} \frac{n}{\chi}(1-\tfrac{\chi}{n})^{-i_{j+1}} \leq \frac{n^2}{\chi}\frac{1}{k}\sum_{j=0}^{k-1} \frac{((1-\frac{\chi}{n})^{-n})^{(j+1)/k}}{2+m_n(j/k)-\varepsilon_{m_n}}.$$

Thanks to the uniform convergence $m_n \to m$ and $(1-\frac{\chi}{n})^{-nx} \to e^{\chi x}$ for $x \in [0,1]$ when $n \to \infty$, the sum converges to $\int_0^1 \frac{e^{\chi x}}{2+m(x)}$ as $k \to \infty$, as required. □

3.1 The Vanilla $(\mu+1)$ GA Is Not Faster Than the Vanilla $(1+1)$ EA for $\mu \in O(\sqrt{n}/\log^2 n)$.

In this section, we show that the vanilla $(\mu+1)$ GA is not faster than the $(1+1)$ EA for any population size $\mu = O(\sqrt{n}/\log^2 n)$, where "vanilla" means standard bit-wise mutation, uniform crossover, and uniform tie-breaker. This is because the diversity at equilibrium is not big enough to provide extra free-riders.

We will make the simplifying assumption that the population is initialized with the all-0 string. This allows us to start with a normal fitness level, and to prove inductively that gaps between fitness levels are not larger than $O(\log n)$. We first show that the expected diversity when leaving a fitness level is small.

Lemma 11. *Consider the $(\mu+1)$ GA with $\mu = o(\sqrt{n})$. Let $1 \leq i \leq n - \omega(\mu)$ and suppose that P_{t_0} is consolidated for some $t_0 \geq 0$, and $f(P_{t_0}) = i$. Then*

$$\mathbb{E}[d_{T_i^o}] = O\big(\tfrac{\mu^2}{n} d(P_{t_0}) + \tfrac{\mu}{n-i}\big).$$

Proof. We use a result from [21] (Corollary 3.4 and Theorem 4.3) to show that after truncating the population P_t to the non-optimized part, called Q_t, satisfies

$$\mathbb{E}[S(Q_{t+1}) \mid S(Q_t)] = (1 - \tfrac{2}{\mu^2} + \tfrac{4(\mu-1)\chi}{\mu^2(n-i)})S(Q_t) + 2(\mu-1)\chi.$$

Solving this recursion explicitly and summing over all possible exit times t^* from this fitness level yields the lemma. We omit the details. □

We call a fitness level i *good* if there is a consolidated population on level i and $d(P) \leq 1/\mu$. Next we show inductively that the algorithm frequently encounters good fitness levels. This allows us to bound any gain form non-normal fitness levels by $O(\log n)$.

Lemma 12. *Let $\mu = O(\sqrt{n}/\log^2 n)$ and let $i = n - \omega(\mu^2)$ be a good fitness level. Then with probability $1 - o(1/n)$ there is a good fitness level $i' = i + O(\log n)$.*

Now, we can prove that the expected number of extra free-riders is negligible.

Lemma 13. *Consider the $(\mu+1)$ GA with $\mu = O(\sqrt{n}/\log^2 n)$ starting in the all-zero string. Then for all $1 \leq i \leq n - \omega(\mu^2)$, $\mathbb{E}[\mathrm{EF}_i \mid N_i] = o(1)$.*

Proof. Under some good event E (the time t^* spend on level i is not too small and the diversity approaches equilibrium at least once before t^*), we show

$$\mathbb{E}[d_{t^*} \mid E] = O(\tfrac{\mu}{n-i}),$$

and Lemma 12 allows us to bound the contribution of the event $\bar{E}$. The statement follows then from Lemma 11. □

3.2 Breaking Ties Towards Diversity Speeds up the $(2+1)$ GA

In this section, we consider the tie-breaker that, when two individuals are in a tie, chooses the one that has the highest S-value relative to the rest of the population. This tie-breaker is unbiased. It is similar to that studied by [5] (Sect. 5.5), who proved that it significantly enhances the optimization of the JUMP functions. We first observe that this tie-breaker indeed improves d_t when the offspring produced at t is not better than one of the individuals.

Lemma 14. *For the above tiebreaker, suppose that the offspring y produced at time t is not fitter than any of the individuals in P_t. Then $d_{t+1} \geq d_t$.*

The first step is to study the diversity evolution process on a fitness plateau. The proof is similar to the proof of Lemma 11 and is omitted.

Lemma 15. *Consider a run of the $(2+1)$ GA using the diversity-improving tie-breaker and standard bit-wise mutation with any mutation rate χ and uniform crossover. Let $0 \leq i \leq n-1$. Then, $\mathbb{E}[d_{T_i^\circ} \mid N_i] \geq \frac{1-p_c}{3-2p_c} + o(1)$.*

This lemma guarantees a larger expected number of extra free-riders.

Lemma 16. *For $0 \leq i \leq n$, $\Pr[\mathrm{EF}_i \geq 1 \mid N_i, d_{T_i^\circ}] \geq \frac{p_c}{8} d_{T_i^\circ}$,*

With this lemma, one can show our main result of this section.

Theorem 17. *Consider the $(2+1)$ GA using standard bit-wise mutation, uniform crossover with constant probability p_c, and the diversity-improving tie-breaker. Then its runtime T satisfies:*

$$\mathbb{E}[T] \leq \tfrac{2}{2+p_c(1-p_c)/(24-16p_c)} \cdot \tfrac{e^\chi - 1}{2\chi^2} n^2 + o(n^2)$$

Note that the first factor is a constant strictly smaller than 1, and the second factor is up to a $(1+o(1))$ the runtime of a vanilla $(2+1)$ GA or a $(1+1)$ EA. This means that the diversity-improving tie-breaker brings a constant factor improvement. Empirically, we found a speedup about twice as large as our bound.

This bound is optimized for $p_c \approx 0.6$, which is also experimentally the optimal static crossover value. However, our proof yields an adaptive mechanism for the crossover probability, which is to set $p_c = 1$ when the population is not consolidated, and $p_c = 0$ otherwise. This predicted improvement was also confirmed experimentally.

4 Conclusion

In this work, we examined the connection between population diversity and progress on the LeadingOnes problem by the $(\mu + 1)$ GA. We have shown that the naturally evolving diversity for any $\mu = o(\sqrt{n}/\log^2 n)$ is not enough to improve the runtime by more than a $(1 + o(1))$ factor. On the other hand, even for $\mu = 2$ simple tie-breaking in favor of diversity leads to so much diversity in the population that the runtime decreases by a constant factor.

There are many question that we had to leave open. The most interesting is what happens for $\mu = \Omega(\sqrt{n}/\log^2 n) \cap o(n/\log n)$. We conjecture that the vanilla version can not create enough diversity to give a constant factor speed-up, even though the average Hamming distance between parents increases further. We conjecture further that the problem is not that differences in bits are never created, but instead the problem is that they also get lost again. So, if there was a way of preventing these differences to get lost, we may even hope for algorithms which get extra free-riders on *most* fitness levels, which could lead to asymptotically optimization time $o(n^2)$. We believe that this is a very interesting setting to explore more systematically potential diversity-preserving mechanisms.

Disclosure of Interests. The author do not have conflicting interests.

References

1. 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.) Parallel Problem Solving from Nature, PPSN XI, pp. 1–10. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_1
2. Cerf, S., Lengler, J.: How population diversity influences the efficiency of crossover. arXiv preprint (2024)
3. Corus, D., Oliveto, P.S.: Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. IEEE Trans. Evol. Comput. **22**, 720–732 (2018)
4. Dang, D.C., et al.: Escaping local optima using crossover with emergent diversity. IEEE Trans. Evol. Comput. **22**(3), 484–497 (2017)
5. Dang, D., et al.: Escaping local optima using crossover with emergent diversity. IEEE Trans. Evol. Comput. **22**, 484–497 (2018)
6. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theoret. Comput. Sci. **567**, 87–104 (2015)
7. Doerr, B., Echarghaoui, A., Jamal, M., Krejca, M.S.: Lasting diversity and superior runtime guarantees for the $(\mu + 1)$ genetic algorithm. CoRR **abs/2302.12570** (2023)
8. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. Theoret. Comput. Sci. **425**, 17–33 (2012)
9. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**, 673–697 (2012)
10. 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)
11. Doerr, C.: Complexity theory for discrete black-box optimization heuristics. Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 133–212 (2020)
12. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoret. Comput. Sci. **276**, 51–81 (2002)

13. Friedrich, T., et al.: Crossover for cardinality constrained optimization. In: Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 1399–1407. ACM (2022)
14. Jansen, T., Wegener, I.: On the analysis of evolutionary algorithms — a proof that crossover really can help. In: Nešetřil, J. (ed.) Algorithms - ESA' 99, pp. 184–193. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48481-7_17
15. Jansen, T., Wegener, I.: Real royal road functions-where crossover provably is essential. Discret. Appl. Math. **149**(1–3), 111–125 (2005)
16. Jorritsma, J., Lengler, J., Sudholt, D.: Comma selection outperforms plus selection on OneMax with randomly planted optima. In: Genetic and Evolutionary Computation Conference (GECCO) (2023)
17. Kötzing, T., Sudholt, D., Theile, M.: How crossover helps in pseudo-Boolean optimization. In: Proceedings of GECCO'11, pp. 989–996. ACM (2011)
18. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Algorithmica **64**, 623–642 (2012)
19. Lehre, P.K., Yao, X.: Crossover can be constructive when computing unique input-output sequences. Soft. Comput. **15**, 1675–1687 (2011)
20. Lengler, J.: Drift analysis. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 89–131. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_2
21. Lengler, J., Opris, A., Sudholt, D.: Analysing equilibrium states for population diversity. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1628–1636 (2023)
22. Lengler, J., Opris, A., Sudholt, D.: A Tight $O(4^k/p_c)$ runtime bound for a $(\mu + 1)$ GA on Jump_k for realistic crossover probabilities. In: Proceedings of the Genetic and Evolutionary Computation Conference, p to appear (2024)
23. Neumann, F., Oliveto, P.S., Rudolph, G., Sudholt, D.: On the effectiveness of crossover for migration in parallel evolutionary algorithms. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 1587–1594 (2011)
24. Oliveto, P.S., He, J., Yao, X.: Analysis of population-based evolutionary algorithms for the vertex cover problem. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 1563–1570 (2008). https://doi.org/10.1109/CEC.2008.4631000
25. Oliveto, P.S., Sudholt, D., Witt, C.: Tight bounds on the expected runtime of a standard steady state genetic algorithm. Algorithmica., 1–56 (2022)
26. Rudolph, G.: Convergence Properties of Evolutionary Algorithms. Verlag Dr, Kovăc (1997)
27. Sudholt, D.: A new method for lower bounds on the running time of evolutionary algorithms. IEEE Trans. Evol. Comput. **17**(3), 418–435 (2012)
28. Sudholt, D.: How crossover speeds up building block assembly in genetic algorithms. Evol. Comput. **25**, 237–274 (2017)
29. Sudholt, D.: The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 359–404. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_8
30. Whitley, D.: Next generation genetic algorithms: a user's guide and tutorial. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics, pp. 245–274. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4_8
31. Witt, C.: Runtime analysis of the (μ + 1) EA on simple pseudo-Boolean functions. Evol. Comput. **14**, 65–86 (2006)
32. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Comb. Probab. Comput. **22**(2), 294–318 (2013)

Overcoming Binary Adversarial Optimisation with Competitive Coevolution

Per Kristian Lehre and Shishen Lin(✉)

University of Birmingham, Birmingham B15 2TT, UK
{p.k.lehre,sxl1242}@cs.bham.ac.uk

Abstract. Co-evolutionary algorithms (CoEAs), which pair candidate designs with test cases, are frequently used in adversarial optimisation, particularly for binary test-based problems where designs and tests yield binary outcomes. The effectiveness of designs is determined by their performance against tests, and the value of tests is based on their ability to identify failing designs, often leading to more sophisticated tests and improved designs. However, CoEAs can exhibit complex, sometimes pathological behaviours like disengagement. Through runtime analysis, we aim to rigorously analyse whether CoEAs can efficiently solve test-based adversarial optimisation problems in an expected polynomial runtime.

This paper carries out the first rigorous runtime analysis of $(1, \lambda)$-CoEA for binary test-based adversarial optimisation problems. In particular, we introduce a binary test-based benchmark problem called DIAGONAL problem and initiate the first runtime analysis of competitive CoEA on this problem. The mathematical analysis shows that the $(1, \lambda)$-CoEA can efficiently find an ε approximation to the optimal solution of the DIAGONAL problem, i.e. in expected polynomial runtime assuming sufficiently low mutation rates and large offspring population size. On the other hand, the standard $(1, \lambda)$-EA fails to find an ε approximation to the optimal solution of the DIAGONAL problem in polynomial runtime. This illustrates the potential of coevolution for solving binary adversarial optimisation problems.

Keywords: Adversarial Optimisation · Theory of Computation · Competitive Coevolution

1 Introduction

CoEAs are a class of algorithms that have been applied in various game-theoretic and strategic optimisation scenarios. There are two main types of CoEAs: cooperative and competitive CoEAs. Competitive CoEAs can be applied to problems that involve adversaries, such as MAXIMIN optimisation problems. With the widespread use and development of GANs [7], there are recent successful

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 117–132, 2024.
https://doi.org/10.1007/978-3-031-70071-2_8

applications of competitive CoEAs for GANs [1,28] and co-evolutionary learning [19]. Competitive CoEAs share similarities with neural network-based adversarial models but require less information, e.g., a gradient. However, despite their potential, the application of CoEAs is challenging. One of the main difficulties is that these algorithms often exhibit pathological behaviours, such as cyclic behaviours, disengagement, and over-specialisation [22,29]. More precisely, cyclic behaviour means that the solution A dominates B, B dominates C, but C can dominate A. This leads to the problem of the algorithm forgetting the optimal solution previously found; for example, RLS-PD suffers from an evolutionary forgetting issue for finding Nash Equilibrium [6]. Disengagement and over-specialisation mean one of the population is too strong, and the other barely learns or optimises from coevolution. These challenges limit the widespread use of CoEAs.

There is a growing interest in optimisation problems that involve one or more adversaries. These problems include robust optimisation or designing game-playing strategies. We focus on a special case of adversarial problems called test-based optimisation problems [13]. Test-based optimisation is an important class of optimisation problems where individuals in a population of designs are evaluated against test cases, which co-evolve with the designs [14]. For example, in supervised learning, we consider model parameters as solutions and training data as test cases [8]. Researchers also apply reinforcement learning methods in board games, including Go [27] and Stratego [21], which consider the opponents as test cases in their self-plays. Gradient-based methods have been used to tackle these problems when the "payoff" function is differentiable [26]. However, in many real-world scenarios, the payoff function is not differentiable, for example, when the strategy space is discrete. In these cases, CoEAs have been suggested to be a promising approach [22]. A notable early success in this field was the work by Hillis [10], who used competitive CoEAs to optimise sorting networks and their corresponding test cases. Other early examples of the successful application of CoEAs on test-based problems can be found in works by [2] and [18]. De Jong et al. explored test-based problems in the context of either coevolution [14] or multi-objective optimisation [15] from empirical studies. There is a gap in the theoretical understanding of CoEAs on test-based problems.

Hillis [10] showed empirically that there is a significant improvement in sorting networks via competitive CoEAs compared with normal EAs. But it is still unclear why competitive CoEAs lead to a better design than traditional EAs [22,24]. We would like to understand how competitive CoEAs work on test-based optimisation problems from the simplest example. We formalise a general problem class, which includes Hillis' co-evolutionary approach on sorting networks as follows: consider a function $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, where $\mathcal{X}$ is a set of designs and $\mathcal{Y}$ is a set of test cases. We define $g(x,y) = 1$ if and only if design x passes test case y. Our optimisation problem can be defined as follows: $arg\max_{x\in\mathcal{X}} \min_{y\in\mathcal{Y}} g(x,y)$. In other words, the Maximin Optimisation is to find $(x^*,y^*) \in \mathcal{X} \times \mathcal{Y}$ such that

$$\text{for all } (x,y) \in \mathcal{X} \times \mathcal{Y}, g(x,y^*) \leq g(x^*,y^*) \leq g(x^*,y).$$

So here come the following research questions:

1. Under what circumstances can a competitive CoEA obtain an optimal solution in polynomial expected time?
2. How does the runtime depend on the problem (g) and on the algorithm?

To understand the questions above, we proceed by using runtime analysis. Runtime analysis of traditional evolutionary algorithms considers the time complexity of a given randomised algorithm. It provides either lower or upper bounds for the number of fitness function evaluations (called runtime) to understand the performance of given algorithms [5]. Runtime analysis can identify relationships between algorithmic parameters and problem characteristics that determine the efficiency of evolutionary algorithms. There is limited literature about runtime analysis of CoEAs apart from [6,12,16]. We want to develop more runtime analysis for CoEAs and expect the insights from runtime analysis of CoEAs will improve the design of CoEAs [22].

Our Contributions. We first introduce a formulation of a binary test-based adversarial optimisation problem, the Diagonal problem. We prove that the traditional $(1, \lambda)$-EA cannot solve the Diagonal Game in expected polynomial runtime. However, we rigorously show for the first time that, with the help of coevolution, (1,λ)-CoEA can solve Diagonal problems in expected polynomial runtime under certain settings by using a two-phase analysis and order statistics tools. This suggests the promising potential of coevolution for solving binary adversarial optimisation problems.

2 Preliminaries

For a filtration $\mathcal{F}_t$, we write $\mathrm{E}_t(\cdot) := \mathrm{E}(\cdot|\mathcal{F}_t)$ and $\Pr_t(\cdot) := \Pr(\cdot|\mathcal{F}_t)$. Denote the 1-norm as $|z|_1 = \sum_{i=1}^{n} z_i$ for $z \in \{0,1\}^n$. Denote $X_t = |x_t|_1 \in [n] \cup \{0\}$ for $x_t \in \{0,1\}^n$ and $Y_t = |y_t|_1 \in [n] \cup \{0\}$ for $y_t \in \{0,1\}^n$ for any $n \in \mathbb{N}$. We focus on the search space $\mathcal{X} \times \mathcal{Y} = \{0,1\}^n \times \{0,1\}^n$ for any $n \in \mathbb{N}$. We consider the filtration $(\mathcal{F}_t)_{t \geq 0}$ including the information of $(X_0, Y_0), \ldots, (X_t, Y_t)$ in this paper. For any $\varepsilon \in [0,1]$ and any problem with a unique optimum (x^*, y^*), we say that algorithm A finds an ε-approximation to the optimum (x^*, y^*) in iteration $T \in \mathbb{N}$ if $||x^*|_1 - |x_T|_1| + ||y^*|_1 - |y_T|_1| < \varepsilon n$ where (x_T, y_T) is the search point of A at iteration T. "With high probability" is abbreviated to "w.h.p.".

2.1 Diagonal Games

In order to model the binary test-based optimisation problem inspired by Hillis's method of sorting networks [10], a payoff function with $\mathcal{X} \times \mathcal{Y}$ as input and $\{0,1\}$ as output is introduced as follows. Here $\mathcal{X} = \{0,1\}^n$ is the solution space of a set of designs for sorting networks and $\mathcal{Y} = \{0,1\}^n$ is the solution space of a set of test cases. We continue to consider a function $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. We define

$g(x, y) = 1$ if and only if design x passes test case y. Our optimisation problem can be defined in terms of MAXIMIN optimisation: *find* $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ *such that*

$$\text{for all } (x, y) \in \mathcal{X} \times \mathcal{Y}, g(x, y^*) \leq g(x^*, y^*) \leq g(x^*, y).$$

We want to start our analysis with simple problems with clear structures so we introduce a constraint function c as follows, which splits the search space into several parts.

Definition 1. *(Generalised boundary test-based problem) Given a constraint function* $c(z) : \mathbb{R} \to \mathbb{R}$*, a generalised boundary function is called* GENERALBOUNDARY $g_c : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$*, where* $\mathcal{X} = \{0, 1\}^n$ *and* $\mathcal{Y} = \{0, 1\}^n$*, if*

$$g_c(x, y) = \begin{cases} 1 & |y|_1 \leq c(|x|_1) \\ 0 & \textit{otherwise.} \end{cases}$$

In our case, we start with a linear constraint function $c(|x|_1) = |x|_1$*.*

Definition 2. *For* $\mathcal{X} = \{0, 1\}^n$ *and* $\mathcal{Y} = \{0, 1\}^n$*, the payoff function* DIAGONAL $: \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ *is*

$$\textsc{Diagonal}(x, y) := \begin{cases} 1 & |y|_1 \leq |x|_1 \\ 0 & \textit{otherwise} \end{cases}.$$

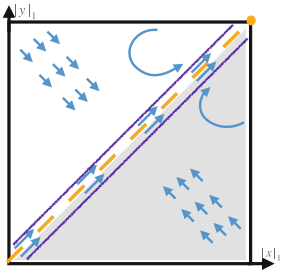


Fig. 1. Example of DIAGONAL problem. The horizontal axis represents the number of 1-bits in the designs x, and the vertical axis represents the number of 1-bits in the test cases y. The grey area represents search points of payoff 1, and the rest represents search points with payoff 0. (Color figure online)

We also use $g(x, y)$ to denote DIAGONAL. Notice that 1 means the design x passes the test cases $y \in \mathcal{Y}$. In this simple DIAGONAL game, y_n with $|y_n|_1 = n$ represents the most difficult test case, and x_n with $|x_n|_1 = n$ is the only solution that can pass y_n (i.e. $g(x_n, y_n) = 1$). Thus, $(1^n, 1^n)$ is the MAXIMIN optimum in this case. In this optimum, neither the design nor the test case is willing to deviate from affecting their payoff $g(x, y)$ anymore. This exactly coincides with the definition of Nash equilibrium. This paper aims to explore whether the CoEAs can find such an optimal solution efficiently.

2.2 Drift Analysis Toolbox

Before our analysis, we introduce the Negative Drift Theorem [20,25], which will be used to prove the inefficiency of algorithms.

Theorem 1 (Negative Drift Theorem [4,20,25]). *For constants a, b, δ, η, $r > 0$, with $a < b$, there exist $c > 0$, $n_0 \in \mathbb{N}$ such that the following holds for all $n \geq n_0$. Suppose $(X_t)_{t\geq 0}$ is a sequence of random variables with a finite state space $S \subset \mathbb{R}_0^+$ and with associated filtration $\mathcal{F}_t$. Assume $X_0 \geq bn$, and let $T_a := \min\{t \geq 0 \mid X_t \leq an\}$ be the hitting time of $\mathcal{S} \cap [0, an]$. Assume further that for all $s \in \mathcal{S}$ with $s > an$, for all $j \in \mathbb{N}_0$, and for all $t \geq 0$, the following conditions hold:*

(1) $\mathrm{E}(X_t - X_{t+1} \mid \mathcal{F}_t, X_t = s) \leq -\delta$
(2) $\Pr[|X_t - X_{t+1}| \geq j \mid \mathcal{F}_t, X_t = s] \leq \frac{r}{(1+\eta)^j}$

Then, $\Pr[T_a \leq e^{cn}] \leq e^{-cn}$.

Before our analysis, we introduce the Additive Drift Theorem [4,9], which will be used to provide the bounds for the runtime of algorithms.

Theorem 2 (Additive Drift Theorem [4,9]). *Let $(X_t)_{t\geq 0}$ be a sequence of non-negative random variables with a finite state space $\mathcal{S} \subseteq \mathbb{R}_0^+$ such that $0 \in \mathcal{S}$. Let $T := \inf\{t \geq 0 \mid X_t = 0\}$.*

(1) If there exists $\delta > 0$ such that for all $s \in \mathcal{S} \setminus \{0\}$ and for all $t \geq 0$, $\mathrm{E}(X_t - X_{t+1} \mid X_t = s) \geq \delta$, then $\mathrm{E}(T) \leq \mathrm{E}(X_0)/\delta$.
(2) If there exists $\delta > 0$ such that for all $s \in \mathcal{S} \setminus \{0\}$ and for all $t \geq 0$, $\mathrm{E}(X_t - X_{t+1} \mid X_t = s) \leq \delta$, then $\mathrm{E}(T) \geq \mathrm{E}(X_0)/\delta$.

3 Traditional Evolutionary Algorithm Cannot Solve Diagonal Efficiently

In this section, we would like to explore whether traditional $(1, \lambda)$-EA can efficiently solve problems with only binary fitness. For a fair comparison between traditional evolutionary and coevolutionary algorithms, we chose $(1, \lambda)$-EA as the closest traditional evolutionary algorithm to the coevolutionary algorithm studied in this paper.

Algorithm 1 samples x uniformly at random. We define the same mutation operator $\mathcal{D}_{\mathrm{mut}}^t$ for x. Ω is the sample space and $\omega_t \in \Omega$ means that the algorithm performs bit-wise mutation for each bit in the bit-string with probability χ/n where $\chi \in (0, 1]$ in iteration t where x is of length n[1]. Next, we evaluate each individual by taking the fitness of i-th offspring. Then, until the termination criteria are met, only the bit-wise mutation operator mutates x. After that, Algorithm 1 selects the individual with the best fitness. If there is a tie in line 5 of Algorithm 1, then we consider choosing among all the individuals of the highest fitness uniformly at random. Next, we prove the following theorem.

[1] We consider $\chi \in (0, 1]$ including the default choice $\chi = 1$ used in [25].

Algorithm 1. $(1, \lambda)$-EA [11]

Require: Search spaces $\mathcal{X}$.
Require: Mutation $\mathcal{D}^t_{\text{mut}} : \Omega \times \{0,1\}^n \to \{0,1\}^n$.
Require: Payoff function $f : \mathcal{X} \to \mathbb{R}$.
1: Set $t := 1$ and choose $x_t \in \mathcal{X}$ uniformly at random.
2: **loop** until the termination criteria met
3: Set $t := t + 1$
4: Let $y_{t,1} := \mathcal{D}(\omega_t, x_{t-1}), \ldots, y_{t,\lambda} := \mathcal{D}(\omega_t, x_{t-1})$.
5: Choose $y_t \in \{y_{t,1}, \ldots, y_{t,\lambda}\}$ among all elements with the largest f-value.
6: $(1, \lambda)$-EA: Set $x_t := y_t$.
7: Go to 2.

Theorem 3. *Given any $\varepsilon \in (0, 1/4)$, $n \in \mathbb{N}$ and the function $f : \{0,1\}^{2n} \to \{0,1\}$ s.t. $z = (x, y)$ where $x, y \in \{0,1\}^n$ and $f(z) = \textsc{Diagonal}(x, y)$, the runtime of $(1, \lambda)$-EA with $\lambda = poly(n)$ and constant $\chi \in (0, 1]$ on finding an ε-approximation to the optimum (i.e. Nash Equilibrium) of f is at least $e^{\Omega(n)}$ with probability $1 - e^{-\Omega(n)}$.*

From our analysis of $(1, \lambda)$-EA on Diagonal, Theorem 3 shows that for any constant $\varepsilon \in (0, 1/4)$, standard $(1, \lambda)$-EA cannot find any ε-approximation of the Maximin optimum of Diagonal efficiently. Moreover, for expected runtime T_ε, if we apply Markov's inequality:

$$E[T_\varepsilon] \geq e^{cn} \Pr(T_\varepsilon \geq e^{cn}) = e^{cn}(1 - e^{-cn}) = e^{\Omega(n)}.$$

Diagonal only consists of binary values, resulting in a very hard and flat fitness landscape in the search space. It leads to a random walk of the search point on the search space, which consists of fitness 1 (i.e. in a grey area of Fig. 1). A further insight is that traditional EAs (e.g. $(1, \lambda)$-EA, $(1 + 1)$ EA) cannot cope well with the interaction between x and y since these algorithms might only favour the highest (or lowest) fitness other than Nash equilibrium in Diagonal[2]. The approach of considering a pair of individuals from two distinct populations ($x \in \mathcal{X}$ and $y \in \mathcal{Y}$) as a single entity within one population (represented as $z := (x, y)$) fails to capture the complexity of interactions between these two populations. So is there any alternative evolutionary approach that could help us to resolve this issue?

4 Competitive Coevolution Solves Diagonal Efficiently

After our analysis, we know that $(1, \lambda)$-EA cannot solve the Diagonal Games efficiently. We are wondering whether a competitive CoEA exists that can solve this problem by changing the selection mechanism or increasing the size of offspring. The key is to properly capture the complex interaction between populations so the algorithm can find Nash equilibrium efficiently. So, we extend

[2] The proof of Theorem 3 can be generalised to a broader class of traditional EAs.

the traditional evolutionary algorithm in the context of competitive CoEAs and consider the $(1, \lambda)$-variant of CoEAs. Then, we prove $(1, \lambda)$-CoEA solves the DIAGONAL problem in expected polynomial runtime.

Algorithm 2. $(1, \lambda)$-CoEA (Alternating Update)

Require: Search spaces $\mathcal{X}, \mathcal{Y}$.
Require: Mutation $\mathcal{D}^t_{\text{mut}}(x) : \Omega \to \{0,1\}^n$ for all $x \in \{0,1\}^n$
Require: Payoff function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
Sample $x \sim \text{Unif}(\mathcal{X})$; Sample $y \sim \text{Unif}(\mathcal{Y})$.
for $t \in \{1, 2, \dots\}$ **do**
  if $t \bmod 2 = 0$ **then**
    for $i = 1$ to λ **do**
      $x^i \sim \mathcal{D}^t_{\text{mut}}(x)$;
    $x := \arg\max_{i \in [\lambda]} g(x^i, y)$;
  else
    for $i = 1$ to λ **do**
      $y^i \sim \mathcal{D}^t_{\text{mut}}(y)$;
    $y := \arg\max_{i \in [\lambda]} -g(x, y^i)$;

Algorithm 2 samples both design x and test case y uniformly at random. We define the same mutation operator $\mathcal{D}^t_{\text{mut}}$ for both x and y. Ω is the sample space and $\mathcal{D}^t_{\text{mut}}(x)$ means that for any $x \in \{0,1\}^n$, the algorithm performs bit-wise mutation for each bit in the bit-string with probability χ/n for $\chi \in (0, n)$ in iteration t. Then, instead of using pairwise dominance, we evaluate each design by taking the payoff of i-th offspring against the parent opponent and evaluate each test case by taking the payoff of it against the parent design for λ offspring. Then, until the termination criteria are met, only the bit-wise mutation operator mutates either x or y in an alternating manner. After that, Algorithm 2 selects the pair of the design and the test case of the best fitness. In the following analysis, we define $f(x^i) := g(x^i, y)$ and $h(y^i) := -g(x, y^i)$ where x, y, x^i, y^i are defined in Algorithm 2. In this paper, in order to archive polynomial runtime for Algorithm 2, we restrict $\lambda \in \text{poly}(n)$.

4.1 Characteristic Lemma for Alternating Update

In this section, without loss of generality, we write the λ offspring at generation $t \in \mathbb{N}$ in descending order in their 1-norms: $|x_t^{(1)}|_1 \geq |x_t^{(2)}|_1 \geq \cdots \geq |x_t^{(\lambda)}|_1$ and $|y_t^{(1)}|_1 \geq |y_t^{(2)}|_1 \geq \cdots \geq |y_t^{(\lambda)}|_1$. We also use $X_t^{(i)}$ and $Y_t^{(i)}$ to denote $|x_t^{(i)}|_1$ and $|y_t^{(i)}|_1$ respectively. $X_t := |x_t|_1$ and $Y_t = |y_t|_1$ where $x_t, y_t \in \{0,1\}^n$ are current search point at iteration $t \in \mathbb{N}$.

Lemma 1. *Consider the fitness of x-bitstring denoted by f and the fitness of y-bitstring denoted by h in Algorithm 2,*

(1) If $|x^{(1)}|_1 \geq |x^{(2)}|_1$*, then* $f(x^{(1)}) \geq f(x^{(2)})$.
(2) If $|y^{(1)}|_1 \geq |y^{(2)}|_1$*, then* $h(y^{(1)}) \geq h(y^{(2)})$.

By Lemma 1, if λ offspring all have the same 1-norms, then they have the same fitness. The algorithm will conduct a random walk around the search space. The algorithm makes actual progress based on argmax selection mechanism when "crossing the diagonal". Next, we rigorously define it.

Definition 3. *(Cross the diagonal) Given Algorithm 2 applied to* DIAGONAL*, and for current search point* $(x_t, y_t) \in \{0,1\}^n \times \{0,1\}^n$*, assume that we have the* λ *offspring at iteration* $t \in \mathbb{N}$ *in descending order in their 1-norms. We say that* λ *offspring cross the diagonal* ***horizontally*** *at iteration* t*, if there exist some* k *such that* $|x_t^k|_1 \geq |y_t|_1$ *with* $|y_t|_1 > |x_t|_1$*; We say that* λ *offspring cross the diagonal* ***vertically*** *at iteration* t *if there exist some* ℓ *such that* $|y_t^\ell|_1 > |x_t|_1$ *with* $|x_t|_1 \geq |y_t|_1$*. We say* λ *offspring cross the diagonal if either occurs.*

Definition 3 means that when crossing the diagonal, the fitness of either x-bitstring or y-bitstring strictly improves. Next, we introduce the concept of a c-tube (the purple strip as presented in Fig. 1).

Definition 4. *(c-tube) We call* $C = \{(x,y) \in \mathcal{X} \times \mathcal{Y} \mid ||x|_1 - |y|_1| < c\}$ *c-tube. We say a current search point* $(x,y) \in \mathcal{X} \times \mathcal{Y}$ *lies outside the c-tube if* $(x,y) \notin C$.

4.2 Phase 1

Algorithm 2 updates the search point in an alternating manner. From the characteristic lemma and definition of crossing the diagonal, we know when $Y_t - X_t > 0$, Algorithm 2 makes progress on searching the optimum by updating X_t to let it cross the diagonal and vice versa. In the analysis, we want to avoid the case when $Y_t - X_t > 0$, but Algorithm 2 updates Y_t instead of X_t in c-tube. This inspires the following definition. We define a successful cycle formally.

Definition 5. *Given* $c > 0$*, then for all* $t \in \mathbb{N}$*, we have a successful cycle in iteration* $2t$ *with respect to* c *which consists of two consecutive steps if* $X_{2t} + c > Y_{2t} > X_{2t}$*,* $Y_{2t+1} + c > X_{2t+1} \geq Y_{2t+1}$ *and* $X_{2t+2} + c > Y_{2t+2} > X_{2t+2}$*. We have a successful cycle in iteration* $2t+1$ *with respect to* c *which consists of two consecutive steps if* $Y_{2t+1} + c > X_{2t+1} \geq Y_{2t+1}$*,* $X_{2t+2} + c > Y_{2t+1} \geq X_{2t+2}$ *and* $X_{2t+3} + c > Y_{2t+3} \geq Y_{2t+3}$.

A successful cycle implies that the algorithm crosses the diagonal twice without leaving the c-tube. We show that the search point will move along the diagonal towards the optimum. We use $H_t := 2n - (X_t + Y_t)$ as the potential function to show there is a positive drift towards the optimum $(X_t, Y_t) = (n, n)$ although with a small probability of escaping from the c-tube.

In the following analysis, we consider a deterministic initialisation to simplify the analysis. The following analysis sufficiently covers the core principle of how Algorithm 2 works via competitive coevolution. We consider an initialisation at $(0^n, 0^n)$ which is the farthest search point with respect to the optimum (n, n) in terms of Hamming distance. First, we present our main lemma in this subsection.

Lemma 2. *Assume that Algorithm 2 is initialised with* $(X_0, Y_0) = (0, 0)$. *For all* $t \in \mathbb{N}$, *we define* $D_t := |X_t - Y_t|$ *and* $T := \inf\{t > 0 \mid H_t := 2n - X_t - Y_t < \varepsilon n\}$ *and for all* $\tau \in \mathbb{N}$, *let event* E_τ *denote that the algorithm has* τ *consecutive successful cycles with respect to* c *or* $\min_{t\in[2\tau]} H_t < \varepsilon n$. *If* $\exists c > 0, p_c, p_e \in [0, 1]$ *s.t. for all* $t \in [1, T/2)$,

(1) $\Pr(X_{2t+1} \geq Y_{2t+1} \mid X_{2t} < Y_{2t}) \geq 1 - p_c$;
(2) $\Pr(X_{2t+2} < Y_{2t+2} \mid X_{2t+1} \geq Y_{2t+1}) \geq 1 - p_c$;
(3a) $\Pr(D_{2t+1} > c \mid D_{2t} < c) \leq p_e$;
(3b) $\Pr(D_{2t+2} > c \mid D_{2t+1} < c) \leq p_e$,

then $\Pr(E_\tau) \geq 1 - 2\tau(p_c + p_e)$ *for any constant* $\varepsilon \in (0, 1)$.

The conditions (1), (2) means that the search point crosses the diagonal at iteration either $2t$ or $2t + 1$ with probability at least $1 - p_c$. The condition (3) means that the search point escapes from the c-tube at iteration $t + 1$ with probability at most p_e for all $t \in [1, T)$. So Lemma 2 gives a lower bound for the probability the algorithm has τ consecutive successful cycles.

We will use Lemma 2 to wrap up all the arguments with $\tau = \Omega(n)$ and $\lambda = n^{\Omega(1)}$ later. We proceed with Phase 2 by assuming a successful cycle always exists in the analysis. In other words, when $Y_t > X_t$, a successful cycle guarantees that we update X_t and when $Y_t \leq X_t$, a successful cycle guarantees that we update Y_t. We will compute p_c and p_e in Phase 2.

4.3 Phase 2

Let us define $D_t = |X_t - Y_t|$ to be the distance away from the diagonal. After the search point crosses the diagonal, we start Phase 2. We divide Phase 2 into three sections. Firstly, we show that given a search point stays within some c-tube, the λ offspring cross the diagonal with probability $1 - 2\left(\frac{1}{\lambda}\right)^{\frac{1}{2e^X}}$. Meanwhile, the search point escapes from the c-tube with prob. bounded by $\frac{1}{\lambda^{O(1)}}$. Secondly, we show that we always have a positive drift when the search point crosses the diagonal. Finally, we wrap up everything using a restart argument, which accounts for the failed generations.

Phase 2.1. Firstly, we need some lemmas to formulate the most likely scenarios when λ offspring are produced.

Lemma 3. *[3] Given a binomial random variable* $Z \sim Bin(n, p)$, $\Pr(Z \, is \, even) = \frac{1}{2} + \frac{1}{2} \cdot (1 - 2p)^n$.

We will use Lemma 3 to show the following Lemma 4. So we can see from Lemma 4, that we need sufficiently large offspring size to avoid the case that $X_t^{(1)}$ coincides with $X_t^{(\lambda)}$ and guarantee that there is some offspring identical to the parents pair with high probability.

Lemma 4. *Given problem size, offspring size and iteration* $n, \lambda, t \in \mathbb{N}$ *and mutation rate* $\chi = O(1)$*, if* t *is even, then*

$$\Pr\left(\exists k \in [\lambda], x_t^k = x_t\right) \geq 1 - e^{-\Omega(\lambda)}, \Pr\left(\max_{i\in[\lambda]} X_t^i > \min_{i\in[\lambda]} X_t^i\right) \geq 1 - e^{-\Omega(\lambda)}$$

If t *is odd, then*

$$\Pr\left(\exists \ell \in [\lambda], y_t^\ell = y_t\right) \geq 1 - e^{-\Omega(\lambda)}, \Pr\left(\max_{i\in[\lambda]} Y_t^i > \min_{i\in[\lambda]} Y_t^i\right) \geq 1 - e^{-\Omega(\lambda)}.$$

Next, we provide some useful concentration inequality, which can be used to show how much deviation is made by each i-th offspring in the number of 1-bits. To obtain the concentration, we proceed by using Moment Generating Functions (MGFs).

Lemma 5. *For* $n \in \mathbb{N}$ *and any* $s \in [n] \cup \{0\}$*, we define* $U = V_1 - V_2$ *where* $V_1 \sim Bin(n - s, \chi/n)$ *and* $V_2 \sim Bin(s, \chi/n)$ *are independent, with* $\chi < n$*. The MGF (moment generating function)* $M_U(\eta) \leq \exp\left(\chi(e^\eta - 1)\right)$ *for any* $\eta > 0$*.*

Lemma 6. *With the same setting as Lemma 5, for any* $s \geq 0$ *and* $\lambda \geq 1$*,* $\Pr(U \geq s) \leq e^{-\chi}\lambda^\chi e^{-s\ln\ln\lambda}$*. Furthermore, for any* $s \geq e^2\chi$ *and any* $\lambda \geq 1$*,* $\Pr(U \geq s) \leq e^{-\chi}e^{-s}$*.*

Given Algorithm 2 with constant $\chi > 0$, we define the number of 1-bits in each offspring in t iteration as $X_t^{(i)}$ where $i \in [\lambda]$. Given the current number of 1-bits $s \in [n]$ for the parent solution, we define the change of the number of 1 in X_t for each offspring by $\Delta X_t^{(i)} \sim V_1 - V_2$ where $V_1 \sim Bin(n-s, \frac{\chi}{n})$ and $V_2 \sim Bin(s, \frac{\chi}{n})$. So $\Delta X_t^{(i)}$ has the same MGFs from Lemma 5 for each $i \in [n]$. From Lemma 6, for any $i \in [\lambda]$ and $s > 0$, we have $\Pr\left(\Delta X_t^{(i)} > s\right) \leq e^{-\chi}\lambda^\chi e^{-s\ln\ln\lambda}$.

Phase 2.2. Next, we show in a certain c-tube that the search point crosses the diagonal with high probability, resulting in the positive drift towards the optimum. First, we show the search point induced by Algorithm 2 crosses the diagonal w.h.p.

Lemma 7. *Given problem size, offspring size and iteration* $n, \lambda, t \in \mathbb{N}$*, let* $c := \kappa \ln\lambda / \ln\ln\lambda$ *for any constant* $\kappa \in (0, 1)$*, we denote* $E_t = \{$*The algorithm crosses the diagonal as defined in Definition 3 at iteration* $t+1\}$*. Assume any constants* $\chi > 0$ *and* $\varepsilon \in (0, 1)$*. If* $D_t < c$*,* $n - X_t \geq \varepsilon n$ *and* $n - Y_t \geq \varepsilon n$*, and if either of the two conditions holds* (1) t *is even and* $X_t < Y_t$*;* (2) t *is odd and* $X_t \geq Y_t$*, then* $\Pr(E_t) \geq 1 - 2\left(1/\lambda^{1/2e^\chi}\right)$

Lemma 7 means that if the search point lies in a given tube of length c, then before reaching an ε-approximation, the search point keeps crossing the diagonal with high probability.

Next, we show that the search point deviates from some c-tube with a small probability. The idea is to show the algorithm is more likely to select the samples/offspring which lie inside the c-tube. We first prove some lemma to proceed.

Lemma 8. *Let problem size, offspring size and iteration* $n, \lambda, t \in \mathbb{N}$, $n - X_t \geq \varepsilon n$ *and* $n - Y_t \geq \varepsilon n$ *for any constant* $\varepsilon \in (0,1)$ *and constant mutation rate* $\chi \in (0,1)$. *For any* $\kappa \in (\chi, (1+\chi)/2)$, *let* $c := \kappa \ln \lambda / \ln \ln \lambda$,

(1) if t *is even and* $X_t < Y_t$, *then* c *satisfies the following conditions*
  (A) $\Pr\left(\max_{i\in[\lambda]} \Delta X_t^i \geq D_t \mid D_t < c\right) \geq 1 - 2\left(1/\lambda^{1/2e^\chi}\right)$
  (B) $\Pr\left(K/M \leq 2(1+\delta)\left(1/\lambda^{\kappa-\chi}\right)\right) \geq 1 - e^{-\Omega(\lambda)}$ *for any constant* $\delta \in (0,1)$ *where* $\Delta X_t^i := X_t^i - X_t$, *and* $K = |\{i \mid \Delta X_t^i \geq D_t + c\}|$ *and* $M = |\{i \mid \Delta X_t^i \geq D_t\}|$.
(2) If t *is odd and* $X_t \geq Y_t$, *then* c *satisfies*
  (C) $\Pr\left(\max_{i\in[\lambda]} \Delta Y_t^i \geq D_t \mid D_t < c\right) \geq 1 - 2\left(1/\lambda^{1/2e^\chi}\right)$
  (D) $\Pr\left(K'/M' \leq 2(1+\delta)\left(1/\lambda^{\kappa-\chi}\right)\right) \geq 1 - e^{-\Omega(\lambda)}$ *for any constant* $\delta \in (0,1)$ *where* $\Delta Y_t^i := Y_t^i - Y_t$, *and* $K' = |\{i \mid \Delta Y_t^i \geq D_t + c\}|$ *and* $M' = |\{i \mid \Delta Y_t^i \geq D_t\}|$.

Notice that in Lemma 8, ΔX_t^i denotes the change of number of 1-bits in i-th offspring of x_t, and $K = |\{i \mid \Delta X_t^i \geq D_t + c\}|$ means the number of samples/offspring s.t. $\Delta X_t^i \geq D_t + c$ and $M = |\{i \mid \Delta X_t^i \geq D_t\}|$ means the number of samples/offspring s.t. $\Delta X_t^i \geq D_t$ and similar for y_t. (A) and (C) conditions in Lemma 8 means that for sufficiently large λ, the next search point produced by Algorithm 2 can cross the diagonal with high probability. (B) and (D) conditions in Lemma 8 means that in those offspring which cross the diagonal, the portion of offspring which make a large jump to cross outside the c-tube is rare with overwhelmingly high probability.

Lemma 9. *(Escape from the c-tube with small prob.) Assume the conditions of Lemma 8 hold. Consider* $(1,\lambda)$*-CoEA on* Diagonal. *We define* $H_t = 2n - X_t - Y_t$. *Let* $T := \inf\{t > 0 \mid H_t \leq \varepsilon n\}$ *for any constant* $\varepsilon \in (0,1)$. *For any* $t \in [1,T]$, *any constants* $\chi \in (0,1)$ *and* $\gamma \in (0, (1-\chi)/2)$, *we have*

$$\Pr(D_{t+1} > c \mid D_t < c) \leq 9\left(\frac{1}{\lambda}\right)^\gamma$$

where c *is defined in Lemma 8.*

Lemma 9 means that if the search point lies in a given tube of length c, then before reaching ε-approximation, the search point stays in the given tube with probability $1 - O\left(1/\lambda^\gamma\right)$ for constant $\gamma > 0$.

We use Lemma 8 to show that firstly within c-tube, the search point can cross the diagonal with high probability. Then, we consider two extreme cases when the search point lies in the tube. Assume $Y_t > X_t$, the first one is that the search point is close to the upper boundary (with Hamming distance 1), then we need to show it can still cross the diagonal but not cross too much and escape from the lower boundary with high probability. The second extreme case is if the search point stays very close to the diagonal (with Hamming distance 1). It is a challenge to show that the next search point will not escape from the lower boundary since we have already shown within the tube, that there is a

high probability that the offspring jump at least Hamming distance D_t to cross the diagonal. So, it is the case that a few offspring may escape from the lower boundary. The key is to observe that more offspring cross the diagonal lie inside the tube compared with those outside the tube. All the offspring in Algorithm 2 which cross the diagonal have fitness 1. Then Algorithm 2 selects the next search point uniformly at random from these offspring crossing the diagonal and with an overwhelming higher probability to select those inside the tube.

Lemma 10. *Let problem size, offspring size and iteration $n, \lambda, t \in \mathbb{N}$, $n - X_t \geq \varepsilon n$ and $n - Y_t \geq \varepsilon n$ for any constant $\varepsilon \in (0,1)$. For any $\chi \in (0,1)$, let $D_t \in [0, c]$ where $c = (\frac{1+3\chi}{4}) \ln \lambda / \ln \ln \lambda$. If t is even and $Y_t > X_t$, then $X_{t+1} - X_t \geq 1$ with probability at least $1 - O(1/\lambda^{(1-\chi)/4})$. If t is odd and $Y_t \leq X_t$, then $Y_{t+1} - Y_t \geq 1$ with probability at least $1 - O(1/\lambda^{(1-\chi)/4})$. Moreover, let $H_t = 2n - X_t - Y_t$, then $\mathrm{E}_t(H_t - H_{t+1}) \geq 1/2$.*

From Lemma 10, before reaching ε-approximation, if the search point stays within the c-tube, there exists positive constant drift towards the optimum when considering H_t as the Hamming distance to (n, n). Since we show the existence of positive drift, next we come to the main theorem of this paper.

Phase 2.3. Now, we wrap up everything and use a restart argument to compute the overall runtime by using Lemma 2 and Lemma 10. Moreover, we will restart the algorithm every $2T_c$ generation with $x_0 = y_0 = 0^n$. Given the problem size $n \in \mathbb{N}$, we have the main result:

Theorem 4. *Consider the $(1, \lambda)$-CoEA with constant mutation rate $\chi \in (0,1)$, and offspring size $\lambda \geq (264(2-\delta)n)^{4/(1-\chi)}$ for any constant $\delta \in (0,1)$. Assume that the algorithm is initialised at $x = y = 0^n$, and restarted at $x = y = 0^n$ every $2T_c$ generations, with $T_c := 4(2-\delta)n$. Then the expected time to find a δ-approximation to the* Maximin *optimum of* Diagonal *is at most $24(2-\delta)\lambda n$.*

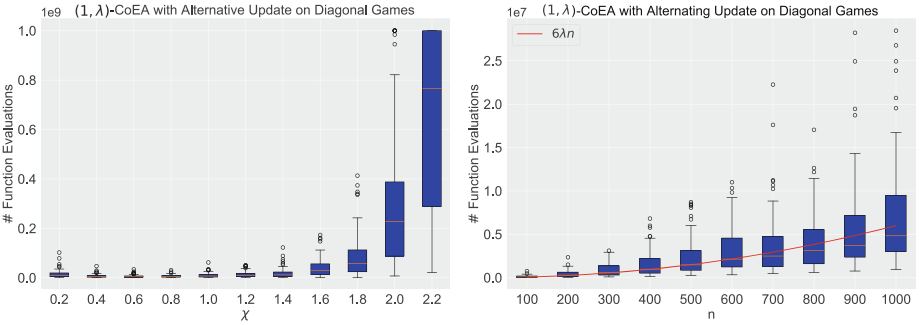
Theorem 4 shows that a large offspring population size helps the CoEA to cross the diagonal consistently with high probability, which is thus favoured during selection and finally leads to the positive drift towards the optimum. Lemma 8 and Lemma 9 show the necessity of large offspring size. Otherwise, a small number of offspring will let CoEA fall to the flat fitness landscape apart from the tube along the diagonal and then get lost in the random walk around the search space. Above all, with proper design of coevolution and large offspring population size, $(1, \lambda)$-CoEA can find an ε-approximation to the optimum of Diagonal efficiently.

5 Experiments

To complement our asymptotic results with data for concrete problem sizes, we conduct the following experiments.

5.1 Settings

We conduct experiments with the $(1, \lambda)$-CoEA on DIAGONAL problem with $n = \lambda = 1000$ and the initialisation of the algorithm is set up as uniformly at random. We also set different mutation rates in the range of 0.2 to 2.2 in increments of 0.2 and conduct 100 independent runs for each configuration to explore the suitable range of mutation rate for the DIAGONAL problem. The budget for each run is set to be 10^9 function evaluations.



(a) Runtime from $\chi = 0.2$ to 2.2. (b) Runtime from $n = 100$ to $n = 1000$.

Fig. 2. Runtime of $(1, \lambda)$-CoEA on DIAGONAL. Figure 2a (**left**): Runtime against different mutation rates under $n = \lambda = 1000$. Figure 2b (**right**): Runtime against specific $\chi = 0.6$. The red curve is $f(n, \lambda) = 6\lambda n$ where in this case we set $n = \lambda$. (Color figure online)

Then, as Fig. 2a shows, $\chi = 0.6$, the expected performance of $(1, \lambda)$-CoEA is the best. Then, we fix such a mutation rate, and run the experiments in the range of $n = 100$ to $n = 1000$ in increments of 100. We conduct 100 independent runs for each configuration. The budget for each run is 10^9 function evaluations.

5.2 Results

Figure 2a displays the runtime of $(1, \lambda)$-CoEA on DIAGONAL for different mutation rate from 0.2 to 2.2. This data confirms that for the suitable low mutation rates and sufficiently large offspring size, Algorithm 2 finds the optimum efficiently. The higher mutation rate eventually leads to the inefficiency of the algorithm. As we observe in Fig. 2b, under suitable mutation rate $\chi = 0.6$, the empirical average of the runtime is bounded above by $O(\lambda n)$. Notice that in our theoretical analysis, we need to require $\lambda = \Omega(n^{4/(1-\chi)})$, while it seems that $\lambda = O(n)$ in the experiments is already sufficient to guarantee a polynomial runtime for $(1, \lambda)$-CoEA on DIAGONAL. This suggests that the current bound may not be tight and our theoretical analysis still has room to improve.

6 Discussion and Conclusion

CoEAs exhibit complex dynamics on Maximin optimisation. To the best of our knowledge, this paper is the first runtime analysis of CoEAs on binary test-based adversarial optimisation problems. As a starting point, we propose a binary test-based problem called Diagonal. We showed that for Diagonal, $(1, \lambda)$-EA get trapped in binary fitness landscape. Thus, traditional $(1, \lambda)$-EA failed to find any approximation to optimum in polynomial runtime due to negative drift induced by flat fitness landscape. However, for $(1,\lambda)$-CoEA with the alternating update method, if the offspring population is sufficiently large $\lambda = \Omega(n^{4/(1-\chi)})$ with a reasonable constant mutation rate $\chi \in (0, 1)$, it can find an approximation to optimum efficiently in expected runtime $O(\lambda n)$. We want to highlight the necessity of coevolution and large offspring size in solving these binary problems in which the fitness landscape is very flat and hard to search from these analyses.

On the technical side, this paper shows that mathematical runtime analyses are also feasible for the $(1, \lambda)$-CoEA. We are optimistic that our tools will widen the toolbox for future analyses of competitive CoEAs. On the practical side, it brings insight for practitioners that traditional EAs may not be well suited for Diagonal-like problems. We suggest using CoEAs with large samples and relatively low mutation rates, which can help to search Nash Equilibria on binary problems with similar hard and flat fitness landscapes more efficiently.

For future work, it is interesting to provide a more precise upper bound for the runtime of $(1, \lambda)$-CoEAs on Diagonal and a more general analysis by relaxing the deterministic initialisation since the empirical results suggest our theoretical bound might not be tight enough. Using more advanced theoretical tools like [17], we can derive a better tail bound of the current runtime for $(1, \lambda)$-CoEA. It is also worth exploring whether there are more efficient ways to capture the interaction between two populations well, for example, any combination of coevolution and self-adaptation or multi-objective optimisation [23]. Furthermore, it is exciting to explore the behaviour of CoEAs on a more general class of payoff functions like generalised boundary test-based problems.

Acknowledgments. This work was supported by a Turing AI Fellowship (EPSRC grant ref EP/V025562/1). The computations were performed using the University of Birmingham's BlueBEAR high performance computing (HPC) service.

Disclosure of Interest. No, I declare no competing interests as defined by Springer Nature, or other interests that might be perceived to influence results and/or discussion reported in this manuscript.

References

1. Al-Dujaili, A., Schmiedlechner, T., Hemberg, A.E., O'Reilly, U.M.: Towards distributed coevolutionary GANs, August 2018. http://arxiv.org/abs/1807.08194, arXiv:1807.08194 [cs]
2. Axelrod, R., et al.: The evolution of strategies in the iterated prisoner's dilemma. Dyn. Norms **1**, 1–16 (1987)
3. Cameron, P.J.: Notes on combinatorics (2007)
4. Doerr, B., Neumann, F.: Theory of evolutionary computation: recent developments in discrete optimization (2019)
5. 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
6. Fajardo, M.A.H., Lehre, P.K., Lin, S.: Runtime analysis of a co-evolutionary algorithm: overcoming negative drift in maximin-optimisation. In: Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA 2023, pp. 73–83. Association for Computing Machinery, New York, NY, USA (2023)
7. Goodfellow, I., et al.: Generative adversarial nets. Commun. ACM **63**(11), 139–144 (2020)
8. Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction., vol. 2. Springer, Cham (2009)
9. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artif. Intell. **127**(1), 57–85 (2001)
10. Hillis, W.: Co-evolving parasites improve simulated evolution as an optimization procedure. Physica D **42**(1–3), 228–234 (1990)
11. Jagerskupper, J., Storch, T.: When the plus strategy outperforms the comma strategy and when not. In: 2007 IEEE Symposium on Foundations of Computational Intelligence, pp. 25–32. IEEE (2007)
12. Jansen, T., Wiegand, R.P.: The cooperative coevolutionary (1+1) EA. Evol. Comput. **12**(4), 405–434 (2004)
13. Jaśkowski, W.: Algorithms for test-based problems. Adviser: Krzysztof Krawiec. Ph.D. thesis. Poznan, Poland: Institute of Computing Science, Poznan University of Technology (2011)
14. Jong, E.D.D., Pollack, J.B.: Ideal evaluation from coevolution. Evol. Comput. **12**(2), 159–192 (2004)
15. Knowles, J., Corne, D., Deb, K.: Multiobjective Problem Solving from Nature: From Concepts to Applications. Natural Computing Series, 1st edn. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72964-8
16. Lehre, P.K.: Runtime analysis of competitive co-evolutionary algorithms for maximin optimisation of a bilinear function. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1408–1416, GECCO 2022 (2022)
17. Lehre, P.K., Lin, S.: Concentration tail-bound analysis of coevolutionary and bandit learning algorithms. arXiv preprint arXiv:2405.04480 (2024)
18. Lindgren, K.: Evolutionary phenomena in simple dynamics. In: Artificial Life II, pp. 295–312 (1992)
19. Mitchell, M.: Coevolutionary learning with spatially distributed populations. Comput. Intell. Principles Pract. **400** (2006)
20. Oliveto, P.S., Witt, C.: Erratum: Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation, November 2012. http://arxiv.org/abs/1211.7184, arXiv:1211.7184 [cs]

21. Perolat, J., et al.: Mastering the game of Stratego with model-free multiagent reinforcement learning. Science **378**(6623), 990–996 (2022)
22. Popovici, E., Bucci, A., Wiegand, R.P., De Jong, E.D.: Coevolutionary principles. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) Handbook of Natural Computing, pp. 987–1033. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-92910-9_31
23. Qin, X., Lehre, P.K.: Self-adaptation via multi-objectivisation: an empirical study. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) International Conference on Parallel Problem Solving from Nature, pp. 308–323. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_22
24. Rosin, C.D.: Coevolutionary search among adversaries. University of California, San Diego (1997)
25. Rowe, J.E., Sudholt, D.: The choice of the offspring population size in the (1,λ) EA. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, GECCO 2012, pp. 1349–1356. Association for Computing Machinery, New York, NY, USA (2012)
26. Ruder, S.: An overview of gradient descent optimization algorithms, June 2017. http://arxiv.org/abs/1609.04747, arXiv:1609.04747 [cs]
27. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
28. Toutouh, J., Hemberg, E., O'Reilly, U.M.: Spatial evolutionary generative adversarial networks. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 472–480. Association for Computing Machinery, New York, NY, USA (2019)
29. Wiegand, R.P.: An analysis of cooperative coevolutionary algorithms. George Mason University (2004)

Evolving Populations of Solved Subgraphs with Crossover and Constraint Repair

Jiwon Lee and Andrew M. Sutton(✉)

Algorithmic Evolution Lab, University of Minnesota Duluth, Duluth, USA
{lee02761,amsutton}@d.umn.edu

Abstract. We introduce a population-based approach to solving parameterized graph problems for which the goal is to identify a small set of vertices subject to a feasibility criterion. The idea is to evolve a population of individuals where each individual corresponds to an optimal solution to a subgraph of the original problem. The crossover operation then combines both solutions and subgraphs with the hope to generate an optimal solution for a slightly larger graph. In order to correctly combine solutions and subgraphs, we propose a new crossover operator called *generalized allelic crossover* which generalizes uniform crossover by associating a probability at each locus depending on the combined alleles of the parents. We prove for graphs with n vertices and m edges, the approach solves the k-vertex cover problem in expected time $O\big(4^k m + m^4 \log n\big)$ using a simple RLS-style mutation. This bound can be improved to $O\big(4^k m + m^2 nk \log n\big)$ by using standard mutation constrained to the vertices of the graph.

1 Introduction

Many combinatorial problems involving graphs require finding a small set of components (e.g., vertices or edges) subject to some feasibility criterion. Examples include the k-vertex cover problem, where a solution is a set of vertices of size at most k that includes at least one endpoint of every edge in the graph, and the k-edge dominating set problem in which a solution is a set of edges of size at most k such that each edge not in the set is adjacent to at least one edge in the set.

When applying methods from evolutionary computation to solve such problems, a popular approach to handle infeasibility is to add a penalty term into the fitness function to ensure feasible solutions are preferred over infeasible solutions. A somewhat orthogonal approach is to employ some kind of *constraint repair* operation that repairs infeasible solutions that may have been produced by mutation or crossover. Recently, Branson and Sutton [1] introduced a focused jump-and-repair operation that tries to repair infeasible solutions by taking a focused jump in the fitness landscape and subsequently invokes a domain-specific repair operator to transform offspring rendered infeasible by mutation into feasible individuals.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 133–148, 2024.
https://doi.org/10.1007/978-3-031-70071-2_9

An open question from [1] is whether populations and crossover might be used together with constraint repair to efficiently solve parameterized problems. We address that question in this paper by developing a new population-based approach in which the population consists of feasible solutions to subgraphs of the original graph. We introduce a generalization of uniform crossover, called generalized allelic crossover, and show that this crossover together with mutation and constraint repair results in fixed-parameter tractable runtime for the k-vertex cover problem. In particular, on graphs with n vertices and m edges, our approach finds a vertex cover of size at most k (if one exists) within $O\big(4^k m + m^4 \log n\big)$ generations using RLS mutation. This bound can be improved to $O\big(4^k m + m^2 nk \log n\big)$ using standard mutation that only affects the segment of the bitstring corresponding to the vertices. Ignoring polynomial factors, these bounds are not as tight as the FPT bounds for (1+1) EA on the same problem [1], but are interesting for a number of reasons. First, the population-based approach can be easily parallelized to incur runtime speedups. Furthermore, even without parallelization we found the approach to be significantly more efficient than the (1+1) EA variant on empirical studies of randomly generated graphs (reported in Sect. 5).

1.1 Background

Constraint repair is one of the main techniques employed to address constrained optimization by randomized search heuristics [2]. Repair-based crossover operators were initially developed in the context of permutation-based encodings [7,13,15]. A repair mechanism based on local search was applied to vertex cover problems and examined empirically by Pelikan, Kalapala and Hartmann [19] for hierarchical Bayesian optimization (hBOA) and the simple genetic algorithm (SGA). The authors showed that these approaches (along with simulated annealing) produced optimal vertex covers on Erdős-Rényi random graphs significantly faster than branch-and-bound.

Finding minimal vertex covers has been an intense subject of research for evolutionary algorithms. Khuri and Bäck [10] investigated handling infeasible solutions by adding a penalty term to the fitness function that strongly discounts candidate solutions that are not vertex covers. They demonstrated that a genetic algorithm significantly outperforms the well-known 2-approximation based on maximal matching on random graphs and structured graphs introduced by Papadimitriou and Steiglitz [18]. This promising empirical performance of evolutionary methods on vertex cover influenced the development of theoretical work on the problem [5,8,17].

From a parameterized complexity perspective, the vertex cover problem was the first problem for which a fixed-parameter tractable evolutionary algorithm was developed in the work of Kratsch and Neumann [11]. They proved that Global SEMO using a tailored mutation operator has an expected optimization time bounded by $O(OPT \cdot n^4 + n \cdot 2^{OPT^2+OPT})$ on any graph G where OPT is the size of a minimum vertex cover of G. When the objective function also uses

cost of an optimal fractional cover (obtained by linear programming) the bound is improved to $O(n^2 \log n + OPT \cdot n^2 + 4^{OPT} n)$.

Branson and Sutton [1] recently showed that a focused jump-and-repair operation can probabilistically simulate *iterative compression*, which is an algorithm design technique for efficiently compressing a feasible solution into a slightly smaller solution. They proved that the (1+1) EA employing focused jump-and-repair and using a restarting framework results in a fixed-parameter tractable $O(2^{OPT} n^2 \log n)$ runtime bound on k-vertex cover problems. They also give fixed-parameter tractable bounds for the FeedbackVertexSet and OddCycleTransversal problems.

In this paper, we show how a carefully designed crossover operator can also leverage populations based on subgraphs to probabilistically simulate iterative compression. The effect of crossover on parameterized complexity has also been studied on the closest string problem [22].

2 Preliminaries

We consider undirected graphs $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is a set of n vertices and $E = \{e_1, \ldots, e_m\} \subseteq \binom{V}{2}$ is a family of m 2-element sets of V called edges. Given a vertex set $S_v \subseteq V$ and an edge set $S_E \subseteq E$, we take the intersection to be the set of vertices that appear in both sets (i.e., the edge set is "flattened"): $S_V \cap S_E = S_V \cap \left(\bigcup_{\{u,v\} \in S_E} \{u, v\}\right)$. The set difference $S_V \setminus S_E$ is defined analogously.

We construct subgraphs of $G = (V, E)$ using subsets of E. Specifically, for any edge set $S_E \subseteq E$, the unique subgraph of G corresponding to S_E is the graph $(V \cap S_E, S_E)$. The neighborhood of $v \in V$ is the set $N(v) := \{u \in V \mid \{u, v\} \in E\}$.

A *vertex cover* of $G = (V, E)$ is a set $S \subseteq V$ such that for every $e \in E$, $e \cap S \neq \emptyset$. The problem of deciding if a graph has a vertex cover of a given size is NP-complete [9], and thus finding a minimum size vertex cover is NP-hard.

Large instances of NP-hard problems can often be solved in practice because real-world instances usually exhibit some kind of structure that can be leveraged by solvers. In these settings, worst-case complexity as a function of problem size alone is not as useful. Parameterized complexity theory [3,4] refines classical complexity theory by factoring the running time of an algorithm into more than one parameter of the input. The aim is to extract the hardness of a problem class by isolating the superpolynomial contribution to the running time to a parameter independent of the problem size.

Formally, a parameterized problem is expressed as a language $L \subseteq \Sigma^* \times \mathbb{N}$ for a finite alphabet Σ. A problem L is *fixed-parameter tractable* if $(x, k) \in L$ can be decided in time $g(k) \cdot |x|^{O(1)}$ for some function g that depends only on k. The complexity class of fixed-parameter tractable problems is FPT. An algorithm is a *Monte Carlo FPT algorithm* for a parameterized problem L if it accepts $(x, k) \in L$ with probability at least 1/2 in time $g(k) \cdot |x|^{O(1)}$ and accepts $x \notin L$ with probability zero.

Parameterized complexity theory is especially relevant to the analysis of evolutionary algorithms in the context of understanding the influence of problem structure on the running time of NP-hard optimization problems [16,22]. Assume $(x, k) \in L$ and let T be the optimization time of a randomized search heuristic (measured, e.g., by the number of calls to the fitness function) until it certifies $(x, k) \in L$. Any randomized search heuristic with a bound $\mathbb{E}[T] \leq g(k) \cdot |x|^{O(1)}$ on L can be transformed into a Monte Carlo FPT algorithm by stopping its execution after $2g(k) \cdot |x|^{O(1)}$ fitness function evaluations. We thus say a randomized search heuristic runs in *randomized FPT time* on a parameterized problem of size n when $\mathbb{E}[T] \leq g(k) \cdot n^{O(1)}$. The parameterized k-vertex cover problem is, given a graph G and an integer k, decide whether there is a vertex cover of size at most k.

2.1 Generalized Allelic Crossover

Let $x, y \in \{0,1\}^n$. With each locus $i \in [n]$, we associate three probabilities $p_i^{(0)}$, $p_i^{(1)}$ and $p_i^{(2)}$. The *generalized allelic crossover* operator (GAC) produces an offspring z from two parents x and y by setting for each $i \in [n]$

$$z_i = \begin{cases} 1 & \text{with probability } p_i^{(x_i+y_i)}, \\ 0 & \text{otherwise.} \end{cases}$$

Standard uniform crossover [23] can be considered a special case of GAC when $p_i^{(0)} = 0$, $p_i^{(1)} = 1/2$, and $p_i^{(2)} = 1$ for all $i \in [n]$. Moreover, GAC can also implement deterministic set operations such as union ($p_i^{(0)} = 0$, $p_i^{(1)} = p_i^{(2)} = 1$) and intersection ($p_i^{(0)} = p_i^{(1)} = 0$, $p_i^{(2)} = 1$). Keeping $p_i^{(0)} = 0$ and $p_i^{(2)} = 1$ but varying $p_i^{(1)}$ recovers the "parameterized uniform crossover" of Spears and De Jong [21] (if $p_i^{(1)} = p_j^{(1)}$ for all $i, j \in [n]$).

We note that GAC is allowed to deviate from common crossover design philosophies. Specifically, when $p_i^{(2)} < 1$ (or $p_i^{(0)} > 0$), it becomes possible for GAC to produce offspring that lie outside the smallest hyperplane containing both parents. Thus, with such settings, GAC is not a *forma-respecting* operator (also called *inheritance-respectful* by Friedrich et al. [6]), nor does it always *strictly transmit* in the sense of Radcliffe [20]. Similarly, it is not necessarily a *geometric* crossover operator in the sense of Moraglio [14], as it can, with certain settings, produce offspring that do not lie in the convex hull described by the parents.

One may argue that such allowances in some sense perverts the original philosophy of crossover, which is meant to share information possessed by all parents. Nevertheless, we show in this paper how it can be leveraged to achieve good results, at least on the vertex cover problem.

3 A Population-Based Subgraph GA for k-Vertex Cover

The philosophy of our approach is to start with a large population of feasible solutions to small subgraphs of G, and allow these subgraphs to produce offspring

that correspond to feasible solutions of slightly larger subgraphs. This process continues until a feasible solution is found for the entire graph G.

In the context of the k-vertex cover problem, a population of feasible solutions to subgraphs corresponds to a set of subgraphs, each with a valid vertex cover of size at most k. We represent each individual as a bit string of length $n + m$ in which the first n elements encode a candidate vertex cover and the last m elements encode a candidate subgraph. Given a bit string $x \in \{0,1\}^{n+m}$, we define the operators $S(x)$ and $E(x)$ that extract the candidate cover and the candidate edge set, respectively. In particular,

$$S(x) \coloneqq \{v_i \in V : i \in [n] \text{ and } x_i = 1\}$$
$$E(x) \coloneqq \{e_i \in E : i \in [m] \text{ and } x_{n+i} = 1\}$$

Using these operators, we can define the concept of feasibility as follows.

Definition 1. *Let $G = (V, E)$ be a graph. An individual $x \in \{0,1\}^{n+m}$ is* feasible *when the vertex set $S(x) \cap E(x)$ corresponds to a feasible k-vertex cover in the subgraph of G selected by $E(x)$, i.e., when*

1. *for each $e \in E(x)$, $e \cap S(x) \neq \emptyset$, and*
2. *$|S(x) \cap E(x)| \leq k$.*

Throughout the paper, we will assume that we are given a graph G that is guaranteed to have a vertex cover of size k. This is not strictly a limitation, since the runtime bounds provided can be used to design a search for the smallest k with probabilistic guarantees on the success of each run (cf the restart framework in [1]).

In each generation, an offspring is created via crossover or mutation. Survival selection proceeds similar to the Global SEMO algorithm from evolutionary multiobjective optimization: if the offspring is not dominated or tied by any individuals in the current population, it is included in the next population. Moreover, any individual in the current population that is dominated by the offspring does not survive.

Definition 2. *A solution x is* dominated *by a solution y, written as $x \prec y$, if and only if at least one of the following conditions holds.*

1. *x is infeasible but y is feasible,*
2. *$E(x) \subset E(y)$, or*
3. *$E(x) = E(y)$ and $|S(x)| > |S(y)|$.*

If $E(x) = E(y)$ and $|S(x)| = |S(y)|$, then x and y are tied. *If y dominates x or x and y are tied, we write $x \preceq y$. If x and y are not tied and x does not dominate y (and vice versa), then x and y are* incomparable.

With probability p_c, in line 6 an offspring is created from two parents by applying a crossover operator, and in line 7, the constraint-repair operation is called that attempts to repair an offspring that was possibly made infeasible by

Algorithm 1: Population-based subgraph GA

Input: A graph $G = (V, E)$ and a crossover probability p_c

1 Initialize P_0;
2 $t \leftarrow 0$;
3 **while** P_t *does not contain an optimal solution* **do**
4 Choose parents $x, y \in P_t$ uniformly at random;
5 **with probability** p_c **do**
6 $z \leftarrow$ Crossover(x,y);
7 $z \leftarrow$ ConstraintRepair(z,x,y,G);
8 **else**
9 $z \leftarrow$ Copy(x);
10 $z \leftarrow$ Mutate(z);
11 **if** $\nexists w \in P_t$ *s.t.* $z \preceq w$ **then**
12 $P_{t+1} = \{w \in P_t \mid w \not\prec z\} \cup \{z\}$;
13 $t \leftarrow t + 1$;

crossover. Otherwise, with probability $1-p_c$, in line 9, no crossover occurs and an arbitrary parent is copied to the offspring and is varied according to a mutation operator in line 10. Finally, in line 12 , the offspring is added to the population if it is not dominated or tied by any element of the population, and all elements of the population dominated by the offspring are removed.

3.1 Variation Operators and Controlling Population Growth

The crossover operation in Algorithm 1 is implemented as generalized allelic crossover (defined in Sect. 2.1) on bitstrings of length $n + m$ with the following probabilities. For all $1 \leq i \leq n + m$, we set $p_i^{(0)} = 0$. Otherwise,

$$p_i^{(1)} = p_i^{(2)} = \begin{cases} 1/2 & \text{for } 1 \leq i \leq n; \\ 1 & \text{for } n+1 \leq i \leq n+m. \end{cases}$$

We may thus think of bitstrings as separated into a length-n *vertex segment* consisting of the first n elements, and a length-m *edge segment* consisting of elements $n+1$ to $n+m$. The crossover probabilities are designed so that vertex segments are combined probabilistically and edge segments are combined deterministically. In particular, given the offspring z of two parents x and y, it always holds that $S(z)$ is a uniform random subset of $S(x) \cup S(y)$ and $E(z) = E(x) \cup E(y)$.

We consider two separate approaches to mutation. For RLS-Mutation, an index i is chosen uniformly at random in $\{1, \ldots, n+m\}$ and the single bit x_i is flipped. For Vertex-Mutation, we flip each bit in x_i with probability $1/n$, but only for indexes in $\{1, \ldots, n\}$ that correspond to the vertices of G.

We do not consider standard bit mutation on the entire bit string for the following reasons. We will require that crossover and mutation always create

offspring that dominates its parent(s), or is always dominated by a parent. Enforcing this constraint ensures that the population size cannot increase during the execution of the algorithm (captured in Lemma 1 below). The Vertex-Mutation operator clearly creates an offspring with a subgraph equal to the one of its parent. The RLS-Mutation operator may add or delete exactly one edge (or none), in which case one subgraph contains the other. Using standard bit mutation on the entire bitstring allows for the chance to create an offspring with an edge set that is incomparable with respect to subset inclusion to the edge set of its parent. This offspring would also be incomparable to its parent in the sense of Definition 2. Allowing such incomparable offspring could cause uncontrolled population growth during the execution of the algorithm. While there may be specific techniques to mitigate this growth, in this paper we will restrict ourselves to the above defined mutation operators that guarantee a population size that does not grow.

Lemma 1. *Consider the execution of Algorithm 1 using either* RLS-Mutation *or* Vertex-Mutation *for its mutation operation. Let P_t denote the population in generation t. If all individuals in P_t are feasible, it holds that (1) all individuals in P_{t+1} are feasible, and (2) $|P_{t+1}| \leq |P_t|$.*

Proof. In line 12 of Algorithm 1, an offspring z is only added to P_{t+1} if it is not dominated nor tied by any individual P_t. The first condition trivially holds, as all infeasible solutions are automatically dominated by feasible solutions.

Let $x, y \in P_t$ be the parents selected in line 4 of Algorithm 1. We argue that either $x \prec z$, $z \prec x$, or x and z are tied. If z is not feasible, then the $z \prec x$ clearly holds, since x is feasible. Thus, assume that z is also feasible. It suffices to show that either $E(x) \subseteq E(z)$ or $E(z) \subseteq E(x)$. If one graph is a proper subset of the other, then we have dominance (of the superset graph). Otherwise, when $E(x) = E(z)$, then x and z are either tied (if $|S(x)| = |S(z)|$) or one dominates the other.

If z was created by crossover, then since $p_i^{(1)} = p_i^{(2)} = 1$ for all indexes $i \in \{n+1, \ldots, n+m\}$, it follows that $E(z) = E(x) \cup E(y)$, and therefore $E(x) \subseteq E(z)$. If z was created by RLS-Mutation, then either $E(x) = E(z)$ (when the flipped bit index is at most n), otherwise $E(z)$ has gained (respectively, lost) exactly one edge compared to $E(x)$. This means that $E(z) \subset E(x)$ (respectively, $E(x) \subset E(z)$). Finally, since Vertex-Mutation does not affect the indexes corresponding to the edges, we would have $E(x) = E(z)$.

Since z is added to the population only if it is not dominated or tied, it follows that $z \in P_{t+1} \iff x \notin P_{t+1}$ and thus we have $|P_{t+1}| \leq |P_t|$. □

3.2 Constraint Repair Operator

Crossover can produce infeasible solutions. We employ a constraint repair operation inspired by the iterative compression procedure that attempts to repair any crossover offspring that does not correspond to a vertex cover.

In particular, if crossover removes a vertex from $S(x) \cup S(y)$, then any uncovered edge can be repaired by adding the neighbors of that vertex back into the

offspring's vertex set. The resulting offspring is guaranteed to be a vertex cover. However, it is not necessarily feasible in the sense of the k-vertex cover problem, as it may return a cover of size greater than k. We formalize the vertex cover repair operator in Algorithm 2

Algorithm 2: ConstraintRepairVC (z, x, y, G)

Input: An offspring z, parents x, y and a graph $G = (V, E)$

1 **if** $S(z)$ *is a feasible cover for* $E(z)$ **then return** z;
`// Store vertices that were removed from` $S(x) \cup S(y)$
2 $A \leftarrow (S(x) \cup S(y)) \setminus S(z)$;
3 **foreach** $v \in A$ **do**
4 $\quad S(z) \leftarrow S(z) \cup N(v)$;
5 **return** z;

The intuition behind this procedure is that since $S(x)$ is a feasible cover for $E(x)$ and $S(y)$ is a feasible cover for $E(y)$, if v belongs to one of them, but not $S(z)$, then the neighbors of v in G may need to be added to potentially cover the edges in $E(z) = E(x) \cup E(y)$.

4 Runtime Analysis

Given an individual $x \in \{0,1\}^{n+m}$, since $S(x)$ is interpreted to be a candidate solution in the subgraph defined by $E(x)$, any $v \in S(x) \setminus E(x)$ does not contribute to the solution. This motivates the following definition.

Definition 3. *We say an individual* $x \in \{0,1\}^{n+m}$ *is* efficient *when* $S(x) \setminus E(x) = \emptyset$. *We call a population efficient when all of its individuals are efficient.*

We are interested in bounding the total expected number of generations that Algorithm 1 spends on populations that are not efficient. The mutation operator together with the fitness domination criteria listed above ensures that there is always selective pressure toward efficient populations, but in some cases the "inefficiency" of a population can increase. However, we show in the following theorem that efficiency is lost only in cases where we are in some sense making overall progress, and thus the number of times this occurs can be bounded.

Theorem 1. *Let* $G = (V, E)$ *be a graph. We assume that* G *is connected. Let* P_0 *be any set of feasible solutions with* $|P_0| = \text{poly}(n)$ *and* $\bigcup_{x \in P_0} E(x) = E$. *Let* $p_c \in (0,1)$ *be a constant. The expected number of generations of Algorithm 1 in which the population is not efficient is bounded above by* $O\big(|P_0|^2 m^2 \log n\big)$ *using* RLS-Mutation *and* $O\big(|P_0|^2 kn \log n\big)$ *using* Vertex-Mutation.

Proof. We design a nonnegative drift potential φ over populations as follows.

$$\varphi(P_t) = \sum_{x \in P_t} |S(x) \setminus E(x)|$$

. Clearly, $\varphi(P_t) = 0$ if and only if P_t is efficient.

This drift function can fluctuate during the course of the execution of Algorithm 1. However, we will later show that the number of times it can increase is strictly bounded. Thus we are able to bound the total time (in expectation) that the algorithm spends waiting for the potential to decrease to zero. Since the remainder of the time the potential would be at zero, the population must be efficient during those generations.

Let $A = \{\varphi(P_{t+1}) > \varphi(P_t)\}$ be the event that the offspring created from population P_t survives into P_{t+1} and results in a strict increase in potential. We first bound the conditional drift of φ on the complementary event $\overline{A}$, namely, the potential of P_{t+1} is not strictly greater than the potential of P_t. A sufficient event to *decrease* the potential is to (1) perform mutation on a single parent with probability $1 - p_c$, (2) select $x \in P_t$ with probability $1/|P_t|$, and (3) flip exactly one bit in $S(x) \setminus E(x)$. Summing the probability of these disjoint events over all possible individuals $x \in P_t$, for RLS-Mutation the conditional drift can be bounded as

$$\mathbb{E}\left[\varphi(P_t) - \varphi(P_{t+1}) \;\middle|\; P_t, \overline{A}\right] \geq \sum_{x \in P_t} \frac{(1-p_c)|S(x) \setminus E(x)|}{|P_t|(n+m)} \geq \frac{(1-p_c)}{|P_0|(n+m)} \cdot \varphi(P_t)$$

since $|P_t| \leq |P_0|$, by Lemma 1. Similarly, for Vertex-Mutation, the probability of flipping exactly one particular bit is $(1/n)(1-1/n)^{n-1} \geq 1/e$, so we have

$$\mathbb{E}\left[\varphi(P_t) - \varphi(P_{t+1}) \;\middle|\; P_t, \overline{A}\right] \geq \frac{(1-p_c)}{|P_0|en} \cdot \varphi(P_t)$$

The expected time until φ hits zero (or increases, if sooner) can be bounded above using multiplicative drift [12] as $O(|P_0|(n+m)\log(|P_0|n)) = O(|P_0|m \log n)$ for RLS-Mutation and $O(|P_0|n \log n)$ for Vertex-Mutation.

We now argue that the total number of times that this potential can increase is strictly bounded for the entire run. We consider two further events in the offspring creation process. Let B denote the event that z is created by mutation from parent $x \in P_t$ and survives into P_{t+1}, necessarily replacing x in the population. Let C denote the event that z is created by a successful crossover between parents x and y, again necessarily replacing x and y in the population.

Note that event C must reduce the population size by at least one because a new feasible subgraph is created from two parents, and those two parents would be dominated by the offspring. Thus the event $A \cap C$ can happen at most $|P_0| - 1$ times during the entire run.

Under RLS-Mutation, a necessary condition for the event $A \cap B$ is that the mutation occurs in the edge segment of the bitstring (indexes larger than n) and particularly, when a bit is flipped from zero to one. If the mutation producing

z had occurred in the vertex segment (indexes at most n) then the potential cannot increase, as this would imply $S(z) > S(x)$ and $E(z) = E(x)$, thereby z would be dominated by x. Similarly, if the mutation occurs in the edge segment and changes a one to a zero, then $E(z) \subset E(x)$ and again z would not survive to be included in P_{t+1}. Since edges can be added to a subgraph in the population at most $|P_0|m$ times, the event $A \cap B$ occurs at most $|P_0|m$ times during the entire run.

Under Vertex-Mutation, event $A \cap B$ only occurs when x is replaced by offspring z where $E(x) = E(z)$, and to compensate for the fact $|S(z) \setminus E(z)| > |S(x) \setminus E(x)|$, it must be true that $|S(z) \cap E(z)| < |S(x) \cap E(x)|$. However, both x and z are necessarily feasible, so $|S(x) \cap E(x)| \leq k$ and thus for any given subgraph in the population, event $A \cap B$ can occur at most k times. Since $B \cup C$ is necessary for A, it follows that $A \cap B$ and $A \cap C$ partition A, and thus the potential can only increase during these events.

Therefore, in the case of RLS-Mutation, the potential can reset at most $|P_0|(m+1)-1$ times, and for Vertex-Mutation, the potential can reset at most $|P_0|(k+1)-1$ times. The claimed bounds thus follow from the multiplicative drift arguments above, and by pessimistically assuming the potential always resets to the highest possible value and all possible resets occur. □

To bound the runtime of Algorithm 1, it remains only to estimate the total time spent on efficient populations. The proof of the following theorem establishes this bound by determining the probability that crossover successfully produces a dominating offspring from any parents in an efficient population. Such an event strictly reduces the population size by combining two solved subgraphs into a larger solved subgraph. The total waiting time for these events together with the time spent on inefficient populations yields the claimed bounds.

Theorem 2. *Let $G = (V, E)$ be a connected graph and let P_0 be any polynomial-size set of feasible individuals such that $E = \bigcup_{x \in P_0} E(x)$.*

Setting $p_c \in (0, 1)$ to be a constant, Algorithm 1 finds a k vertex cover of G (if one exists) in $O\big(4^k|P_0| + |P_0|^2 m^2 \log n\big)$ generations using RLS-Mutation *and in $O\big(4^k|P_0| + |P_0|^2 nk \log n\big)$ generations using* Vertex-Mutation.

Proof. By Theorem 1, the expected number of generations the algorithm spends on populations that are not efficient is at most $O\big(|P_0|^2 m^2 \log n\big)$ using RLS-Mutation and at most $O\big(|P_0|^2 kn \log n\big)$ using Vertex-Mutation.

We thus seek to bound the number of generations spent on efficient populations until an optimal solution is found. Suppose that P_t is efficient and let $x, y \in P_t$ with $E(x) \neq E(y)$. Since both x and y must be feasible, $S(x)$ is a cover of the subgraph $E(x)$ and $S(y)$ is a cover of the subgraph $E(y)$. Moreover, the feasibility of x and y together with the efficiency of P_t guarantees that $|S(x)|, |S(y)| \leq k$. Thus $S(x) \cup S(y)$ is a valid cover of the subgraph $E(x) \cup E(y)$ with $|S(x) \cup S(y)| \leq 2k$. We have also assumed there is a k-cover of the entire graph G, namely $S^* \subseteq V$ where $|S^*| \leq k$. Then $S^* \cap (E(x) \cup E(y))$ is also a cover of the subgraph $E(x) \cup E(y)$. Let $R = (S(x) \cup S(y)) \setminus S^*$ denote the

set of vertices that belong to $S(x) \cup S(y)$, but not to the optimal cover. Let $T = (S(x) \cup S(y)) \cap S^*$ be the set of vertices that belong to both covers.

We consider the application of generalized allelic crossover using x and y as parents to produce an offspring z. Note that $E(z) = E(x) \cup E(y)$ since $p_i^{(1)} = p_i^{(2)} = 1$ for all $n+1 \leq i \leq n+m$ in the edge segment of the bitstring. Similarly, for all $1 \leq i \leq n$ in the vertex segment of the bitstring, we have $p_i^{(1)} = p_i^{(2)} = 1/2$, so every vertex $v \in S(x) \cup S(y)$ belongs to $S(z)$ with probability $1/2$. Since $p_i^{(0)} = 0$ for all i, any vertex (respectively, edge) not in $S(x) \cup S(y)$ (respectively, $E(x) \cup E(y)$) will not belong to $S(z)$ (respectively, $E(z)$).

Note that since $T \subseteq (S(x) \cup S(y))$, we have $S(z) = T$ with probability exactly $2^{-|S(x) \cup S(y)|} \geq 2^{-2k}$. We condition on this event for the remainder of the proof. Every edge in the subgraph $E(x) \cup E(y) = E(z)$ that is not covered by $S(z)$ must have one endpoint in S^* and one endpoint in R because both S^* and $(S(x) \cup S(y))$ are valid vertex covers of $E(x) \cup E(y)$.

After crossover, the repair operation listed in Algorithm 2 identifies the set of vertices removed from $S(x) \cup S(y)$, which in this csea corresponds exactly to the set R, and then add the neighbor set $N(R)$. This results in a repaired offspring z' with $S(z') = T \cup N(R) \subseteq S^*$ which must cover $E(x) \cup E(y) = E(z')$.

The fact $S(z') \subseteq S^*$ implies $|S(z')| \leq k$ and so it follows that $x, y \prec z'$, and since z' would not be dominated by any other element of the population, z' replaces x and y in P_{t+1}. This event, which occurs with probability at least 4^{-k}, results in a strictly smaller population $|P_{t+1}| < |P_t|$.

A feasible, efficient population containing more than one individual can always shrink with probability $\Omega(4^{-k})$ under the above sequence of events. It follows that the waiting time until an efficient population shrinks in this way is bounded above by $O(4^k)$. The population can shrink at most $|P_0| - 1$ times before it consists of a single feasible individual x^*. As GAC always composes subgraphs by union, it holds that $E(x^*) = E$, and since feasibility is maintained $S(x^*)$ is a vertex cover of size at most k for G.

Before generating x^*, the algorithm can spend at most $O(4^k|P_0|)$ generations on efficient populations and the total time spent on inefficient populations is bounded by Theorem 1, which yields the claimed result. □

Theorem 2 requires only an initial population of feasible subgraphs that compose into G. For specific graphs, this could be constructed by including promising subgraphs that are hoped to be "close" to an optimal cover. However, every graph at least has a natural initial population of size m in which each subgraph consists of a single unique edge from G (together with a cover that contains at least one vertex incident on that edge). This yields the following general bound.

Corollary 1. *Let $G = (V, E)$ be a graph on n vertices and m edges. Algorithm 1 finds a vertex cover of size at most k of G (if one exists) in $O\big(4^k m + m^4 \log n\big)$ generations using* RLS-Mutation *and $O\big(4^k m + m^2 nk \log n\big)$ generations using* Vertex-Mutation.

Proof. Construct P_0 from G as follows. For each edge $\{u, v\} \in E$, let x be any string in $\{0,1\}^{n+m}$ such that $x_u = 1$ and $E(x) = \{\{uv\}\}$. Then P_0 satisfies the conditions for Theorem 2. □

5 Experiments

To investigate the concrete running time of Algorithm 1, and to compare it with the similar repair-based (1+1) $\text{EA}_k^{\text{J+R}}$ introduced in [1], we performed a number of experiments on the planted vertex cover instances from [1]. Each of these instances were generated by randomly selecting a subset of k vertices and including an edge with probability p subject to having at least one end point in the subset.

The number of vertices n varies from 20 to 100 by 10, planted cover size k varies from 3 to 10, and edge probabilities are $p \in \{\frac{1}{10}, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$. On each graph we ran each algorithm for 50 trials and measured the run time as the number of calls to the fitness function until a vertex cover of size at most k is found. For Algorithm 1, we set $p_c = 0.8$, and experimented with both RLS-Mutation and Vertex-Mutation. The median run times for $k = 10$ as a function of n are reported in Fig. 1. We omit results for other k-values due to space constraints, but mention that the trend is identical. Note that after generating a random graph for a given n, we remove isolated vertices and the figures report the true n after removal. This explains the variability in n at low edge densities. In Fig. 2, we plot the median run times as a function of k fixing $n = 100$. The bottom right plot shows the median as a function of k taken over all p and n. We also provide box plots of the running time of all three algorithms on graphs with $n = 100, k = 10$ over all edge densities in Fig. 3.

Despite the fact that runtime bound of the (1+1) $\text{EA}_k^{\text{J+R}}$ from [1] is exponentially smaller in k than the one derived in Corollary 1, we see that Algorithm 1 with Vertex-Mutation scales better with both n and k on this class of graphs, and has smaller variability as measured by interquartile range. Not surprisingly, the variant using RLS-Mutation is strongly affected by the number of edges, and performs poorly on denser graphs, as can be seen in Fig. 1.

Khuri and Bäck [10] conducted experiments on hard vertex cover instances using a GA with two-point crossover and proportional selection. In addition to (nonplanted) random graphs, they investigated two structured graph instances originally defined by Papadimitriou and Steiglitz [18] to demonstrate that greedy degree-heuristics fail to approximate minimum vertex covers. These instances (PS100 and PS202) have vertex counts $n = 100, 202$, edge counts $m = 1122, 4556$, and minimum vertex covers of size $k = 34, 68$. We also report the success rates of the population subgraph algorithm on these instances for different runtime budgets in Table 1 along with the success rates reported in [10]. Note that the minimum covers for these graphs are comparatively large. Nevertheless, we still observe surprisingly high success rates, even at runtime budgets much smaller than 4^k.

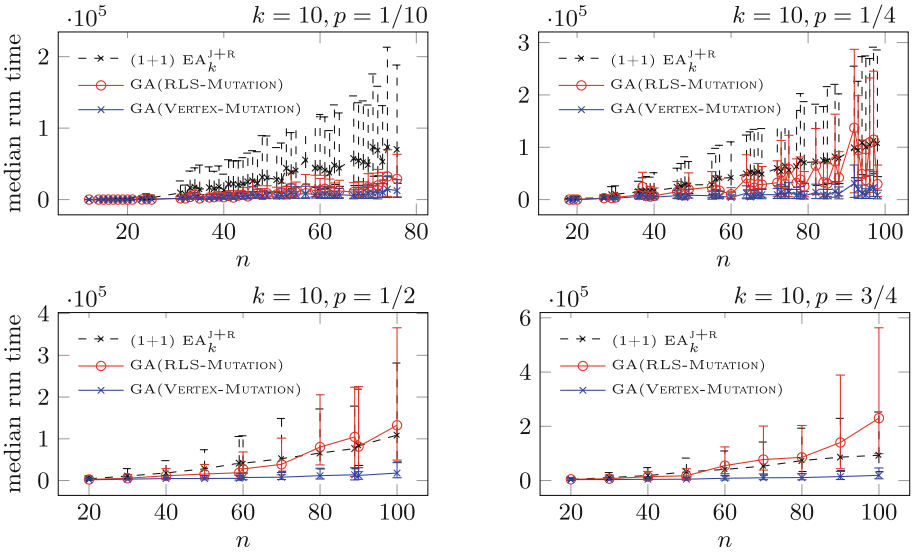


Fig. 1. Median run times of Algorithm 1 and the (1+1) EA_k^{J+R} on random planted $k = 10$ vertex cover instances of varying edge density p as a function of n. Error bars denote interquartile range.

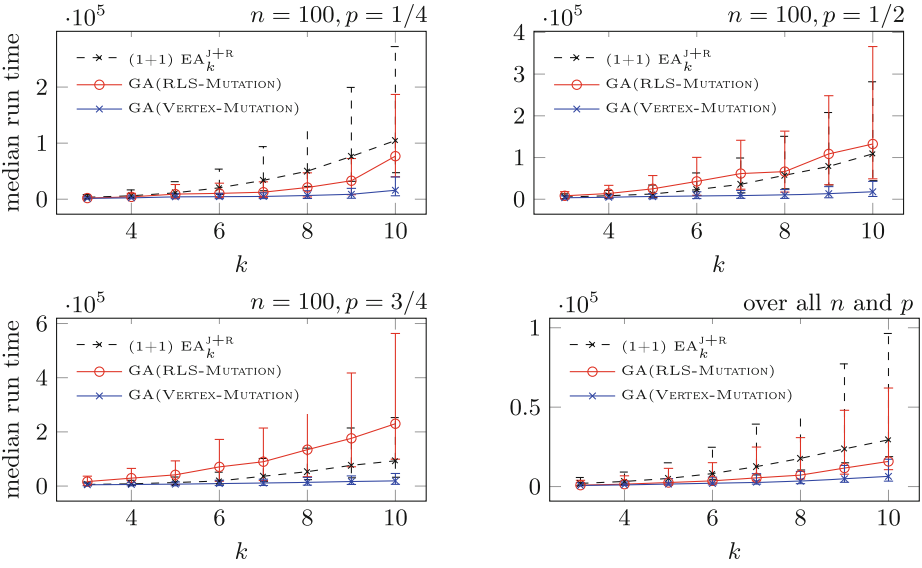


Fig. 2. Median run times of Algorithm 1 and the (1+1) EA_k^{J+R} on random planted vertex cover instances of varying edge density p as a function of k. Error bars denoteinterquartile range.

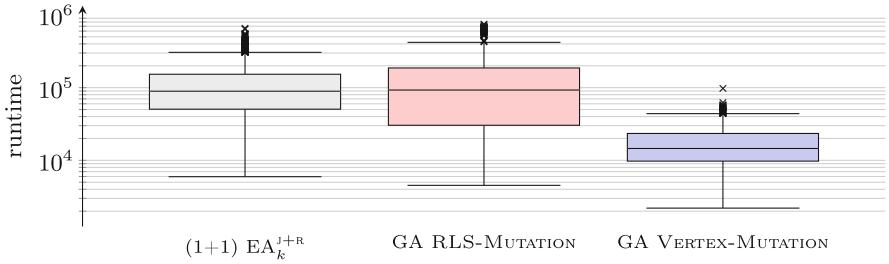


Fig. 3. Runtime statistics for $n = 100$ and $k = 10$ over all edge densities.

Table 1. Success rates on Papadimitriou-Steiglitz instances PS100 and PS202 from [10] for different runtime budgets.

budget	Khuri & Bäck GA [10]		GA RLS-Mutation		GA Vertex-Mutation	
	PS100	PS202	PS100	PS202	PS100	PS202
$2 \cdot 10^4$	65%	—	18%	0%	97%	0%
$4 \cdot 10^4$	—	60%	63%	0%	98%	29%
10^6	—	—	100%	96%	100%	100%

6 Conclusion

We have introduced a population-based technique designed to solve feasible component-selection problems in graphs. In this technique, one begins with a large population of solutions to small subgraphs, e.g., single edges or vertices. We showed that if a suitable constraint repair operation is used, the approach can achieve fixed-parameter tractable running time bounds on the NP-hard k vertex cover problem. Our results give insight into how crossover can be leveraged to exploit structure in hard combinatorial optimization problems. Moreover, experimental results suggest that the population-based approach can be more efficient than the (1+1) EA on certain classes of graphs.

There are a number of potential directions for future work. As yet, no lower bounds exist for FPT evolutionary algorithms on the k-vertex cover problem. This is rather difficult, as the structure of different kinds of graphs have varying and unpredictable degrees of influence on the assorted modules of evolutionary algorithms. Nevertheless, it would be interesting to obtain lower bounds in terms of k for certain graph categories. Moreover, the proposed subgraph approach leverages only a suitable constraint repair operations, so it could be easily extended to similar problems in which a small set of vertices or edges need to be selected subject to some feasibility criterion.

Acknowledgments. This research was funded by NSF grant 2144080.

Disclosure of Interests. The authors have no competing interests.

References

1. Branson, L., Sutton, A.M.: Focused jump-and-repair constraint handling for fixed-parameter tractable graph problems. In: Proceedings of the Sixteenth ACM/SIGEVO Conference on Foundations of Genetic Algorithms. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3450218.3477304
2. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput. Methods Appl. Mech. Eng. **191**(11–12), 1245–1287 (2002). https://doi.org/10.1016/S0045-7825(01)00323-1
3. Downey, R.G., Fellows, M.R.: Parameterized Complexiy. Springer, New York (1999). https://doi.org/10.1007/978-1-4612-0515-9
4. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Cham (2006). https://doi.org/10.1007/3-540-29953-X
5. Friedrich, T., Hebbinghaus, N., Neumann, F., He, J., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. In: Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO), London, UK, pp. 797–804. ACM (2007).https://doi.org/10.1145/1276958.1277118
6. Friedrich, T., et al.: Crossover for cardinality constrained optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). ACM, July 2022. https://doi.org/10.1145/3512290.3528713
7. Goldberg Jr., D.E., Lingle, R.: Alleles, loci, and the traveling salesman problem. In: Grefenstette, J.J. (ed.) Proceedings of the First International Conference on Genetic Algorithms and their Applications (ICGA), Pittsburgh, PA, USA, vol. 154, pp. 154–159. Lawrence Erlbaum, Hillsdale, NJ (1985)
8. Jansen, T., Oliveto, P.S., Zarges, C.: Approximating vertex cover using edge-based representations. In: Neumann, F., Jong, K.A.D. (eds.) Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms (FOGA XII), Adelaide, SA, Australia, 16–20 January 2013, pp. 87–96. ACM (2013). https://doi.org/10.1145/2460239.2460248
9. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a Symposium on the Complexity of Computer Computations, Held 20–22 March 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA, pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
10. Khuri, S., Bäck, T.: An evolutionary heuristic for the minimum vertex cover problem. In: Kunze, J., Stoyan, H. (eds.) Workshops of the Eighteenth Annual German Conference on Artificial Intelligence (KI-1994), pp. 86–90, Saarbrücken, Germany (1994)
11. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica **65**(4), 754–771 (2012). https://doi.org/10.1007/s00453-012-9660-4
12. Lengler, J.: Drift analysis. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation - Recent Developments in Discrete Optimization, pp. 89–131. Natural Computing Series, Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_2

13. Mitchell, G.G., O'Donoghue, D., Barnes, D., McCarville, M.: GeneRepair - a repair operator for genetic algorithms. In: Late-Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO), Chicago, IL, USA, pp. 235–239 (2003). http://mural.maynoothuniversity.ie/10351/
14. Moraglio, A.: Abstract convex evolutionary search. In: Proceedings of the Eleventh Workshop on Foundations of Genetic Algorithms (FOGA XI). ACM, January 2011. https://doi.org/10.1145/1967654.1967668
15. Mühlenbein, H.: Parallel genetic algorithms in combinatorial optimization. In: Balci, O., Sharda, R., Zenios, S.A. (eds.) Computer Science and Operations Research: New Developments in their Interfaces, pp. 441–453. Pergamon Press, Amsterdam (1992). https://doi.org/10.1016/b978-0-08-040806-4.50034-4
16. Neumann, F., Sutton, A.M.: Parameterized complexity analysis of randomized search heuristics. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 213–248. Natural Computing Series. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_4
17. Oliveto, P.S., He, J., Yao, X.: Analysis of the (1+1)-EA for finding approximate solutions to vertex cover problems. IEEE Trans. Evol. Comput. **13**(5), 1006–1029 (2009). https://doi.org/10.1109/tevc.2009.2014362
18. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Upper Saddle River (1982)
19. Pelikan, M., Kalapala, R., Hartmann, A.K.: Hybrid evolutionary algorithms on minimum vertex cover for random graphs. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), London, UK, pp. 547–554. ACM (2007). https://doi.org/10.1145/1276958.1277073
20. Radcliffe, N.J.: Forma analysis and random respectful recombination. In: Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA) (1991)
21. Spears, W.M., Jong, K.A.D.: On the virtues of parameterized crossover. In: Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA), pp. 230–236 (1991)
22. Sutton, A.M.: Fixed-parameter tractability of crossover: steady-state GAs on the closest string problem. Algorithmica **83**(4), 1138–1163 (2021). https://doi.org/10.1007/s00453-021-00809-8
23. Syswerda, G.: Uniform crossover in genetic algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms (ICGA), vol. 3, pp. 2–9 (1989)

Analysis of Evolutionary Diversity Optimisation for the Maximum Matching Problem

Jonathan Gadea Harder[1(✉)], Aneta Neumann[2], and Frank Neumann[2]

[1] Hasso Plattner Institut, Potsdam, Germany
jonandrop.ja@gmail.com
[2] University of Adelaide, Adelaide, South Australia

Abstract. This paper delves into the enhancement of solution diversity in evolutionary algorithms (EAs) for the maximum matching problem, with a particular focus on complete bipartite graphs and paths. We utilize binary string encoding for matchings and employ Hamming distance as the metric for measuring diversity, aiming to maximize it. Central to our research is the $(\mu + 1)$-EA$_D$ and 2P-EA$_D$, applied for diversity optimization, which we rigorously analyze both theoretically and empirically.

For complete bipartite graphs, our runtime analysis demonstrates that, for reasonably small μ, the $(\mu + 1)$-EA$_D$ achieves maximal diversity with an expected runtime of $O(\mu^2 m^4 \log(m))$ for the big gap case (where the population size μ is less than the difference in the sizes of the bipartite partitions) and $O(\mu^2 m^2 \log(m))$ otherwise. For paths we give an upper bound of $O(\mu^3 m^3)$. Additionally, for the 2P-EA$_D$ we give stronger performance bounds of $O(\mu^2 m^2 \log(m))$ for the big gap case, $O(\mu^2 n^2 \log(n))$ otherwise, and $O(\mu^3 m^2)$ for paths. Here n is the total number of vertices and m the number of edges. Our empirical studies, examining the scaling behavior with respect to m and μ, complement these theoretical insights and suggest potential for further refinement of the runtime bounds.

1 Introduction

Evolutionary algorithms (EAs) stand as a robust class of heuristics that navigate the intricate landscapes of various domains, from combinatorial optimization to bioinformatics, and have proven especially valuable in addressing problems within graph theory [25]. Central to the discussion in the field is the concept of diversity within EAs, which has been pivotal in enhancing the search process and preventing premature convergence on suboptimal solutions [11].

1.1 Related Work

Recent research in evolutionary computation investigates various connections between quality and diversity. Quality Diversity (QD) has gained recognition as a widely adopted search paradigm, particularly in the fields of robotics and games [1,4,5,15,29,30]. The goal of QD is to illuminate the space of solution behaviours by exploring various niches

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 149–165, 2024.
https://doi.org/10.1007/978-3-031-70071-2_10

in the feature space and maximizing quality within each specific niche. In particular, the popular MAP-elites algorithm divides the search space into cells to identify the solution with the highest possible quality for each cell [1,19,33,34]

The area of Evolutionary diversity optimization (EDO) aims to find a maximal diverse set of solutions that all meet a given quality criterion. EDO approaches have been applied in a wide range of settings. Diversity, while typically a means to avoid stagnation in the search for a single optimal solution, here is leveraged to yield a set of diverse, high-quality solutions. This is advantageous for decision-makers who value a variety of options from which to select the most fitting solution, accounting for different practical considerations and trade-offs [31,32]. For example the use of different diversity measures has been explored for evolving diverse set of TSP instances that exhibit the difference in performance of algorithms for the traveling salesperson problem as well as differences in terms of features of variation of a given image [6]. In the classical context of combinatorial optimization, EDO algorithms have been designed for problems such as the knapsack problem [2], the computation of minimum spanning trees [3], communication networks [14,24], to compute sets of problem instances [12,22,23], as well as the computation of diverse sets of solutions for monotone submodular functions under given constraints [8,21]. Furthermore, Pareto Diversity Optimization (PDO) has been developed in [20] which is a coevolutionary approach optimizing the quality of the best possible solution as well as computing a diverse set of solutions meeting a given threshold value. EDO approaches have been analyzed with respect to their theoretical behavior for simple single- and multi-objective pseudo-Boolean functions [10] as well as simple scenarios of the traveling salesperson problem [6,26,27], the minimum spanning tree problem [3], the traveling thief problem [28], the permutation problems [7] and the optimization of submodular functions [21].

1.2 Our Contribution

This paper builds upon the methodology of [13] applying the theoretical runtime analysis framework to the maximum matching problem, specifically in bipartite graphs and paths. We aim to provide a deeper understanding of how diversity mechanisms influence the efficiency of population-based EAs in converging to a diverse set of high-quality maximum matchings.

To achieve this, we adopt a binary string representation for matchings and use Hamming distance as a measure of diversity. We then delve into the theoretical underpinnings of evolutionary diversity optimization for the maximum matching problem, examining structural properties that impact the performance of diversity-enhancing mechanisms within EAs. We provide runtime analysis for evolutionary algorithms, shedding light on their scalability for different problem instances. Finally, we present our experimental investigations to assess how close the bounds on the theoretical runtimes match the experimental runtimes.

In summary, our research provides theoretical insights and empirical evidence to understand how diversity can be effectively maximized for the maximum matching problem. Our findings contribute to a deeper understanding of the interplay between diversity and optimization in EAs and pave the way for further research in this direction.

The paper is organized as follows. In Sect. 2, we introduce the maximum matching problem and the evolutionary diversity optimization approaches analyzed in this study. We then explore structural properties and present runtime analyses for diversity optimization in the context of complete bipartite graphs and paths (Sect. 3). Experimental investigations are detailed for both unconstrained and constrained scenarios (Sects. 4 and 5), followed by concluding remarks and suggestions for future research directions (Sect. 6).

2 Preliminaries

In this part of the paper, we present the core concepts related to diversity optimization for matchings in bipartite graphs. We start by establishing the definitions and measures of diversity that will be used throughout our discussion.

2.1 Maximum Matching Problem and Diversity Optimization

Our study is concerned with the matching problem in bipartite graphs, described by a graph $G = (V, E)$. The aim is to find a maximum matching M, which is a collection of edges that do not share common vertices. It is presumed that each individual in the starting population represents a valid maximum matching. Our analysis is directed at determining how long it takes evolutionary algorithms to cultivate a population that is not only diverse but also meets a specified quality benchmark.

Let $x \in \{0,1\}^{|E|}$ represent a bitstring where each bit corresponds to an edge in E, indicating whether the edge is included in the matching.

The divergence between individuals is gauged using the Hamming distance, which is appropriate given our binary string representation of solutions. This distance measures how many bits differ between two strings.

2.2 Diversity Measure

The diversity of a multiset (duplicates allowed) of search points P (called population in the following) is defined as the cumulative Hamming distance across all pairs of unique individual within P. This is mathematically expressed as

$$D(P) = \sum_{(x,y)\in \tilde{P}\times\tilde{P}} H(x,y),$$

where $\tilde{P}$ is the set (no duplicates) containing all solutions in P, and $H(x,y)$ is the Hamming distance between any two solutions x and y. The notion of contribution for a solution x within a population is quantified as the difference in diversity if x were to be excluded and defined as

$$c(x) = D(P) - D(P \setminus \{x\}).$$

It is important to note that $c(x) = 0$ if x is duplicated in P.

2.3 Initial Population

For our analysis, we assume that the initial population consists of maximum matchings. These can be efficiently obtained using deterministic maximum matching algorithms such as the Hopcroft-Karp algorithm for bipartite graphs [18].

Algorithm 1:.$(\mu + 1)$-EA$_D$

Input: A population P of μ maximum matchings, individual length m, mutation probability $1/m$

; **while** *termination criterion not met* **do**
  Choose $s \in P$ uniformly at random
  Produce s' by flipping each bit of s with probability $1/m$ independently
  if s' *meets the quality criteria* **then**
    Add s' to P
    Choose a solution $z \in P$ where $c(z) = \min_{x \in P} c(x)$ u.a.r.
    Set $P := P \setminus \{z\}$
  end
end

2.4 Algorithms

The $(\mu+1)$-EA$_D$ (see Algorithm 1) operates on a principle of maintaining and enhancing diversity within a population. It starts with a population of solutions, iteratively evolving them through mutation. In each iteration, it selects a solution uniformly at random, applies mutation, and if the new solution meets quality criteria, it is added to the population. To maintain population size, the least diverse individual (or one of them, if there are several) is removed. This process continues until the termination criterion is met. In our case this would be achieving maximal diversity and the quality criterion being a valid maximum matching.

The Two-Phase Matching EA$_D$ (see Algorithm 2) is also designed to generate diverse solutions in the population. The first phase involves *unmatching* a random subset of vertices in a solution, while the second phase focuses on *rematching* these vertices to other unmatched vertices in the graph. The algorithm keeps adding these newly formed solutions to the population if they fulfill the quality criteria and, similar to the $(\mu + 1)$-EA$_D$, removes the least diverse solutions to maintain population size. The algorithm continues this process until the set criteria are met, aiming to achieve a diverse set of high-quality matchings.

2.5 Drift Theorems

We analyse the considered algorithms with respect to their runtime behaviour. The expected runtime refers to the expected number of generated offspring until a given

Algorithm 2:. Two-Phase Matching EA_D (2P-EA_D)

Input: A population P of μ maximum matchings, individual length m

while *termination criterion not met* **do**
  Choose $s \in P$ uniformly at random (u.a.r);
  Create s' as a duplicate of s;
  Select a subset of vertices $S \subseteq V$, where each vertex is included with probability $\frac{1}{|V|}$;
  foreach *vertex* $v \in S$ **do**
    Unmatch v in s' (sets only one corresponding bit to 0 if s' is a valid matching)
  end
  foreach *vertex* $v \in S$ **do**
    if *unmatched neighbors of* v *exist* **then**
      Match v in s' u.a.r with an unmatched neighbor
    end
  end
  if s' *meets quality criteria* **then**
    Add s' to P
    Choose a solution $z \in P$ where $c(z) = \min_{xinP} c(x)$ u.a.r.
  end
end

goal has been achieved (usually until a valid population of maximal diversity has been computed). For our analysis, we make use of the additive and multiplicate drift theorems which we state in the following.

Theorem 1 (Additive Drift Theorem [17]**).** *Let* $S \subseteq \mathbb{R}^+$, $(X^t)_{t\in\mathbb{N}}$ *over* $S \cup \{0\}$, *and* $T = \min\{t \mid X^t \leq 0\}$. *For* $\delta > 0$ *with*

$$E[X^t - X^{t+1} \mid T > t] \geq \delta \implies E[T \mid X^0] \leq \frac{X^0}{\delta}$$

Theorem 2 (Multiplicative Drift Theorem [9]**).** *Let* $(X_t)_{t\in\mathbb{N}}$ *be random variables over* $\mathbb{R}$, $x_{\min} > 0$, *and let* $T = \min\{t \mid X_t < x_{\min}\}$. *Furthermore, suppose that* $X_0 \geq x_{\min}$ *and, for all* $t \leq T$, *it holds that* $X_t \geq 0$, *there is some value* $\delta > 0$ *such that, for all* $t < T$, *it holds that* $X_t - E[X_{t+1} \mid X_0, \ldots, X_t] \geq \delta X_t$.
Then $E[T \mid X_0] \leq \frac{1+\ln\left(\frac{X_0}{x_{\min}}\right)}{\delta}$.

3 Runtime Analysis for Complete Bipartite Graphs

This section introduces key theoretical results on complete bipartite graphs. We commence with a lemma that characterizes the conditions for maximal diversity within a population. Subsequently, we present a series of theorems that delineate the expected runtime to achieve this optimal diversity. These theorems compare the performance of the $(\mu + 1)$-EA_D and 2P-EA_D algorithms, providing a quantitative basis for assessing their efficacy.

Lemma 1 (Diversity of a Population). *Let $G = ((L, R), E)$ be a complete bipartite graph with $|R| \leq |L|$. For a population P of size $\mu < \min\{\frac{|R|}{2}, |L| - |R|\}$, maximal diversity $D(P)$ is attained if and only if all matchings in P are pairwise edge-disjoint.*

Proof. See the detailed proof in [16]. □

In the following theorem we show that there is always a local improvement, needing 2 bit flips, to reach a population with maximum diversity if the difference in size between both partitions is larger than the population size.

Theorem 3. *Let $G = ((L, R), E)$ be a complete bipartite graph with $\mu < \frac{|R|}{2}$, $\mu < |L| - |R|$ and $|R| < |L|$. In the $(\mu + 1)$-EA$_D$ applied to G, the expected time until the diversity is maximized is $O(\mu^2 m^2 \log(m))$.*

Proof. We define the potential function X_t as the difference between the optimal diversity div_{opt} and the current diversity $\text{div}(t)$ at time t:

$$X_t := \text{div}_{\text{opt}} - \text{div}(t).$$

In each solution, exactly $|R|$ vertices from L are adjacent to a matching edge, leaving $|L| - |R|$ vertices in L unadjacent in every solution. Additionally, each vertex in R can be matched to at most $\mu < |L| - |R|$ different vertices across all solutions, ensuring that, for each vertex in R, there exists a vertex in L that is not matched with it in any solution.

To show that there is always a 2-bit flip which improves diversity by at least $\frac{X_t}{\mu}$, we focus on a sequence of improving 2-bit flips. Each 2-bit flip corresponds to changing a match for a vertex in R, which entails deactivating one edge (currently part of a matching) and activating another edge (currently not part of the matching). This process is akin to reassigning a vertex in R to a different, unmatched vertex in L.

Consider an edge e used in i solutions. When this edge is deactivated (removed from the matching), the diversity change is $-(\mu - i)$, since $\mu - i$ solutions lose a unique edge, reducing diversity. Conversely, when a new edge is activated (added to the matching) that is unused across all other solutions in the current population, it contributes $\mu - 1$ to the diversity.

Thus, for each such 2-bit flip involving edge e, the total change in diversity is:

$$-(\mu - i) + (\mu - 1) = -\mu + i + \mu - 1 = i - 1.$$

This calculation demonstrates that the diversity improve achieved by applying the 2-bit flip for an edge in the sequence either decreases or remains unchanged if it is flipped later in the sequence. Note that in each step of the sequence the new maximum matching contains an edge unused by any other matching, so the offspring is always valid and the diversity improvement is at least 1, since this would be achieved by replacing the parent. Additionally, when all edges are unique across all solutions, the population becomes optimal. As a result, the total change across these unique edges equals the difference from the optimum, X_t.

Let $\overline{e}$ represent the count of such "imperfect" edges (edges used in more than one solution). Applying the 2-bit flip to one edge of the sequence gives at-least the diversity

increase it achieves in the sequence, since the value of i can only decrease or remain unchanged, and it is at most μ. Thus $\overline{e}\mu \geq X_t$, which implies $\overline{e} \geq \frac{X_t}{\mu}$. The expected drift then is:

$$E[X_t - X_{t+1} \mid X_t] \geq \frac{\overline{e}}{\mu m^2}\left(1 - \frac{1}{m}\right)^{m-2} \geq \frac{X_t}{\mu^2 m^2 e}.$$

Given that $\binom{\mu}{2}2|R|$ is the maximum diversity, when all edges are pairwise distinct, it holds that $X_0 \leq \binom{\mu}{2}2|R| \leq \mu^2|R| \leq |R|^3 \leq |L| \cdot |R| \cdot |R| \leq m^{1.5}$, the application of the multiplicative drift theorem yields the expected runtime of $O(\mu^2 m^2 \log(m))$ to achieve maximum diversity. □

We now show that the Two-Phase Matching Algorithm achieves significant speedup since no longer two edges have to be flipped to change where one vertex is matched to.

Theorem 4. *Let $G = ((L, R), E)$ be a complete bipartite graph with $\mu < \frac{|R|}{2}$, $\mu < |L| - |R|$ and $|R| < |L|$. In the Two-Phase Matching Evolutionary Algorithm applied to G, the expected time until the diversity is maximized is $O(\mu^2 n^2 \log(n))$, where $n = |L| + |R|$.*

Proof. We define the potential function X_t as the difference between the optimal diversity div_{opt} and the current diversity $\text{div}(t)$ at time t:

$$X_t := \text{div}_{\text{opt}} - \text{div}(t).$$

The maximal diversity is achieved when all matchings in the population are pairwise edge-disjoint. The drift in the potential function X_t at each step of the algorithm is analyzed as follows:

In each step, the algorithm first selects a solution and a subset of vertices, which it rematches with unmatched vertices in L. Let $\overline{e}$ represent the count of such "imperfect" edges (edges used in more than one solution). As shown in Theorem 3 it holds that $\overline{e} \geq \frac{X_t}{\mu}$. The expected drift then is obtained by selecting the corresponding solution to any of the $\overline{e}$ edges, unmatching the adjacent vertex in R and rematching it to include an edge unused by any solution. The probability to unmatch any and no other particular vertex in R is $\frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en}$, and the probability of matching it with an appropriate unmatched vertex in L is at-least $\frac{1}{n}$.

The expected decrease in the potential function X_t per step, or the expected drift, is then given by:

$$E[X_t - X_{t+1} \mid X_t] \geq \frac{\overline{e}}{\mu n^2 e} \geq \frac{X_t}{\mu^2 n^2 e},$$

where the factor $\frac{1}{\mu n^2}$ accounts for the probability of selecting the right vertex and making a beneficial rematch.

Given that $\binom{\mu}{2}2|R|$ is the maximum diversity, when all edges are pairwise distinct, it holds that $X_0 \leq \binom{\mu}{2}2|R| \leq \mu^2|R| \leq |R|^3 \leq m^{1.5}$, the application of the multiplicative drift theorem yields the expected runtime of $O(\mu^2 n^2 \log(n))$ to achieve maximum diversity. □

Theorem 6 covers the case $\mu \geq |L| - |R|$ missing in the previous theorems, which gives a much larger runtime bound. Intuitively this happens because as μ gets greater than the gap between $|L| - |R|$ it is no longer guaranteed that we can always find a new rematch, such that this matching edge is not used by any other solution, thus making more than two bit flips necessary. Theorem 5 includes such a situation with a theoretical lower bound.

Theorem 5. *Let $G = ((L, R), E)$ be a complete bipartite graph with $|R| < |L|$. Consider a population size μ, satisfying $\mu < \frac{|R|}{2}$ and $\mu \geq |L| - |R|$. There exists a starting population P_w such that when the $(\mu + 1)$-EA$_D$ is applied to G, the expected time to reach a population with maximal diversity is $\Omega(m^{3.5})$.*

Proof. See the detailed proof in [16]. □

In the following theorem we generalize that while there is not always an improving 2-bit flip, a 4-bit flip can always be found.

Theorem 6. *For a complete bipartite graph $G = ((L, R), E)$ where $|R| < |L|$, let the population size μ satisfy $\mu < \frac{|R|}{2}$ and $\mu \geq |L| - |R|$. When the $(\mu + 1)$-EA$_D$ is applied to G, the expected time to achieve maximal diversity is bounded by $O(\mu^2\, m^4 \log(m))$.*

Proof. We investigate the expected time for the $(\mu + 1)$-EA$_D$ to maximize diversity in a complete bipartite graph with the given conditions. Initially, we note that for any maximum matching there exist $|L| - |R|$ unmatched vertices from the left partition.

Let M be a maximum matching in G. Consider that full diversity is not achieved yet and thus an edge $e_{rl} \in M$ is part of multiple maximum matchings. We define $\overline{L}(r) \subseteq L$ to be the set of vertices in L that are not matched to a vertex $r \in R$ in any of the solutions of the population. We denote by $M(r)$ the vertex in L to which a vertex $r \in R$ is matched under M.

Consider a matching M, where we want to switch $M(r)$ with an element $\bar{l} \in \overline{L}(r)$ to resolve a possible conflict. If there is some $r' \in R \backslash \{r\}$so that for any of the $\mu - 1$ other matchings M', $M(r) \neq M'(r')$ and $M(r') \in \overline{L}(r)$, we can swap $M(r)$ and $M(r')$ and we are done. Otherwise, we have that for some $r' \in R \backslash \{r\}$, $M(r') \in \overline{L}(r)$ and there exist some matching M' so that $M'(r') = M(r)$ such that no swapping between $M(r)$ and $M(r')$ is possible, but given the number of matchings μ, at most $\mu - 1$ distinct element of $\overline{L}(r)$ fall into this case. However

$$|\overline{L}(r)| - (\mu - 1) > |\overline{L}(r)| - \mu > |L| - 2\mu > 0,$$

where we used $\mu < |L|/2$, there must be some r', so that we are in the first case, i.e. there exists must exist some $r' \in R \backslash \{r\}$, such that $M(r') \in \overline{L}(r)$ and we can switch $M(r)$ and $M(r')$ without conflict.

Just as in Theorem 3 each of those 4-bit flips only decreases or does not change the multiplicities of other edges, since they are both unique edges over all solutions. Also successively applying these 4-bit flips at most $\overline{e}$ times will result in optimal diversity, so $\overline{e}\mu \geq X_t$ holds.

Define X_t to be the difference between the optimal diversity and the current diversity at time t. Then, we observe a positive drift in the expected diversity increase per time step, similarly as Theorem 3 which can be bounded below by:

$$E[X_t - X_{t+1} \mid X_t] \geq \frac{\overline{e}}{\mu m^4}\left(1-\frac{1}{m}\right)^{m-4} \geq \frac{X_t}{\mu^2 m^4 e}.$$

Here, $\frac{1}{\mu}$ represents the probability of selecting the correct individual for reassignment, and the term $\frac{1}{m^4}\left(1-\frac{1}{m}\right)^{m-4}$ accounts for the probability of selecting the appropriate edges for activation and deactivation.

Since the maximum initial diversity gap X_0 can be at most $m\mu^2$ (each pair of solutions can have a hamming distance of at most m), the Multiplicative Drift Theorem provides us with a runtime bound of $O(\mu^2 m^4 \log(m))$ to achieve maximum diversity.□

A similar speedup as for $\mu < |L| - |R|$ can be shown by applying the 2P-EA$_D$.

Theorem 7. *Given a complete bipartite graph $G = ((L, R), E)$ with $|R| < |L|$, consider a population size μ that fulfills $\mu < \frac{|R|}{2}$ and $\mu \geq |L| - |R|$. For the 2P-EA$_D$, the expected time to reach maximal diversity is $O(\mu^2 m^2 \log(m))$.*

Proof. Consider the $(\mu + 1)$-EA$_D$ applied to a complete bipartite graph $G = ((L, R), E)$ under the condition $\mu \geq |L| - |R|$. Define the potential function X_t as in the previous theorem:

$$X_t := \text{div}_{\text{opt}} - \text{div}(t).$$

Let $\overline{e}$ be the number of edges that are shared across different matchings. The expected drift in X_t per step, considering the efficient selection and rematching process of only two vertices, is given by:

$$E[X_t - X_{t+1} \mid X_t] \geq \frac{\overline{e}}{\mu n^2 n^2}\left(1-\frac{1}{n}\right)^{n-2} \geq \frac{X_t}{\mu^2 n^4 e},$$

where the factor $\frac{1}{\mu n^2}$ accounts for the probability of selecting the right solution and pair of vertices and $\frac{1}{n^2}$ of making a beneficial rematch. The term $\left(1-\frac{1}{n}\right)^{n-2}$ considers the probability of unmatching and rematching exactly two vertices without affecting the others.

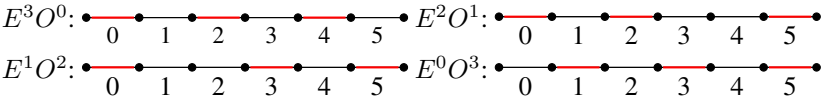
Given the initial diversity gap $X_0 \leq m\mu^2$, applying the Multiplicative Drift Theorem yields an expected runtime of $O(\mu^2 n^4 \log(n))$ to achieve maximum diversity. Now since $|L| - |R| \leq \mu < \frac{|R|}{2}$ it holds that $|R| < |L| < 1.5|R|$, which implies $O(|L|) = O(|R|)$. Also by definition $n = |L| + |R|$, so $O(n^2) = O(|L||R|) = O(m)$ and we get a bound of $O(\mu^2 m^2 \log(m))$. □

4 Runtime Analysis for Paths

This section introduces key theoretical results on paths. We commence with an introduction of useful notation to simplify the following proofs. Subsequently, we present a series of theorems that delineate the expected runtime to achieve this optimal diversity. These theorems compare the performance of the $(\mu+1)$-EA$_D$ and 2P-EA$_D$ algorithms, providing a quantitative basis for assessing their efficacy.

In a path with an even number of edges, such as when $m = 6$, there are multiple ways to form a maximum matching. Each maximum matching includes exactly three edges, ensuring that no two edges in the matching share a vertex. The notation E^iO^j is used to represent these matchings, where i and j denote the number of edges with even and odd indices in the matching, respectively. A detailed proof of this result is provided in [16].

With an even number of edges, such as $m = 6$, there are the following maximum matching configurations, represented as (matching edges in red):



The following Lemma characterizes the conditions for maximal diversity within a population using this notation.

Lemma 2 (Diversity of a Population). *For an m-path with m even and population size $\mu \leq \frac{m}{2} + 1$, the population with optimum diversity contains:*

- *For each j from 0 to $\lfloor \frac{\mu}{2} \rfloor - 1$, the individuals $E_jO_{\frac{m}{2}-j}$ and $E_{\frac{m}{2}-j}O_j$.*
- *For odd μ and $\lfloor \frac{\mu}{2} \rfloor \leq k \leq \frac{m}{2} - \lfloor \frac{\mu}{2} \rfloor$, it further contains any one individual of the form $E_kO_{\frac{m}{2}-k}$.*

Proof. See the detailed proof in [16]. □

Building up on this, in the following theorem we show that there is always a local improvement, needing 2 bit flips, to improve diversity.

Theorem 8. *Let G be a path with m edges, where m is even, and let the population size μ satisfy $2 \leq \mu \leq \frac{m}{2} + 1$. In the $(\mu+1)$-EA$_D$ applied to G, the expected time until the diversity is maximized is $O(\mu^3 m^3)$.*

Proof. We consider a path graph with an even number of edges m, where multiple maximum matchings are possible. The maximum matching is unique when m is odd, hence the maximum diversity is trivially obtained in that case. Therefore, our analysis focuses on when m is even.

Within a population, suppose there is duplication. By Lemma 2 it follows that there exists at least one individual for which the first $i \geq 0$ matched edges have even indices without another individual having the first $i + 1$ matched edges with even indices, or an individual where the last $i \geq 0$ matched edges have odd indices without another individual having the last $i + 1$ matched edges with odd indices.

Considering that the total number of distinct maximum matchings for a path with m edges exceeds μ, the likelihood of choosing an individual from the current population and correctly flipping two edges to enhance diversity is at least $\frac{1}{\mu}\frac{1}{m^2}(1 - \frac{1}{m})^{m-2}$. This lower bound on the probability yields a diversity improvement of at least 1.

If the population has not reached maximal diversity but consists of pairwise distinct maximum matchings, then there must exist a maximal $0 \leq j \leq \lfloor \frac{\mu}{2} - 1 \rfloor$ such that $E^jO^{\frac{m}{2}-j}$ or $E^{\frac{m}{2}-j}O^j$ is not present in the population. W.l.og. let this be $E^jO^{\frac{m}{2}-j}$. We focus on the individual $E^kO^{\frac{m}{2}-k}$ with minimal k s.t. $k < j$ (i.e., with the most odd edges). By applying a 2-bit flip we get $E^{k-1}O^{\frac{m}{2}-k+1}$. The diversity change, by

replacing the parent, would be only determined by this edge change. This new odd edge is already used by j matchings, since j is maximal, and only those since else $E^k O^{\frac{m}{2}-k}$ would not have the most odd edges of the remaining population. By symmetry the deactivated even edge is used in $\mu - j$ solutions (excluding the parent). Thus the change in diversity by replacing the parent would be $\mu - j - j = \mu - 2j$. By choice of j this is strictly positive. Since replacing the parent is possible, the diversity increase is at least of this size. Let X_t denote the difference between the optimal diversity and the current diversity at time t. The possibility of enhancing diversity via a two-bit flip provides us with a drift given by

$$E[X_t - X_{t+1} \mid X_t] \geq \frac{1}{\mu m^2}\left(1 - \frac{1}{m}\right)^{m-2} \geq \frac{1}{\mu m^2 e}.$$

Since the initial diversity deficit X_0 is at most $m\mu^2$ (each pair of solutions can have a hamming distance of at most m), applying the additive drift theorem results in a runtime estimation of $O(\mu^3 m^3)$. □

Theorem 9. *In the 2P-EA$_D$ applied to a path with m edges, the expected time until the diversity is maximized is $O(\mu^3 m^2)$.*

Proof. Since the proof follows closely the arguments presented in Theorem 8, we will focus only on the different bounds on drift, which is the main differing element.

Any maximum matching $E^j O^{\frac{m}{2}-j}, j > 0$ can be chosen with probability $\frac{1}{\mu}$ and be mutated to $E^{j-1} O^{\frac{m}{2}-j+1}$ by unmatching the jth vertex and rematching him with probability $\frac{1}{2}$ to his unmatched left neighbour. Since all previous edges have to be of even index this neighbour must be unmatched. Analogously it holds for $E^j O^{\frac{m}{2}-j}, j < m-1$ to $E^{j+1} O^{\frac{m}{2}-j-1}$. For both the case of having duplicates or not being optimal in Theorem 8 we make use of such a local edge swap. The drift is therefore given by

$$E[X_t - X_{t+1} \mid X_t] \geq \frac{1}{\mu n 2}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{\mu n 2 e}.$$

where $\left(1 - \frac{1}{n}\right)^{n-1}$ is the probability of not rematching any other vertex. Given that the initial diversity deficit X_0 is at most $m\mu^2$ (each pair of solutions can have a hamming distance of at most m), the additive drift theorem provides an upper bound on the expected run time of $O(\mu^3 m^2)$, since $m = n - 1$. □

The upper bounds provided in Theorems 8 and 9 are general worst-case bounds. However, it's worth noting that populations with low diversity may offer more opportunities for improvement, potentially leading to faster convergence in practice.

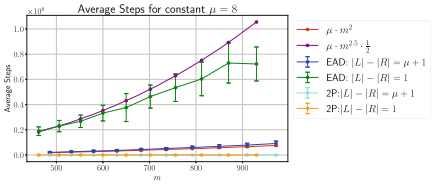
For the $(\mu+1)$-EA$_D$, when diversity is low, there's a higher probability of selecting individuals with shared edges, making beneficial 2-bit flips more likely. Similarly, for the 2P-EA$_D$, low diversity increases the chances of unmatching and rematching vertices in ways that create new, unique matchings.

This observation suggests that the algorithms might exhibit a form of adaptive behavior, where progress is initially rapid when diversity is low, and then slows as the population approaches maximal diversity.

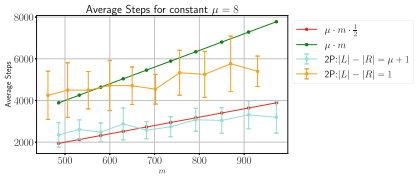
Future work could explore this adaptive nature more rigorously, potentially yielding more nuanced runtime analyses that better reflect the algorithms' performance across different stages of the optimization process.

5 Empirical Analysis

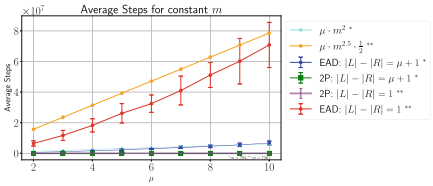
In this section, we present our empirical findings on the performance of the evolutionary diversity algorithms on complete bipartite graphs and paths. Our experiments were designed to test the theoretical predictions made in previous sections, particularly focusing on the efficiency of the algorithm in terms of the number of iterations (steps) required to achieve optimal diversity.



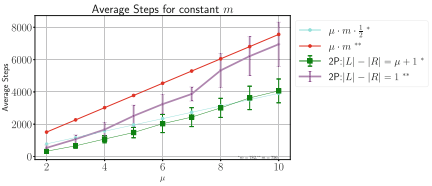
(a) EA_D and 2P with $\mu = 8$ in Comp. Bip. Graphs



(b) 2P with fixed $\mu = 8$ in Comp. Bip. Graphs



(c) AED and 2P with fixed m in Comp. Bip. Graphs



(d) 2P with fixed m in Comp. Bip. Graphs

Fig. 1. Experimental results on complete bipartite graphs

5.1 Experimental Setup

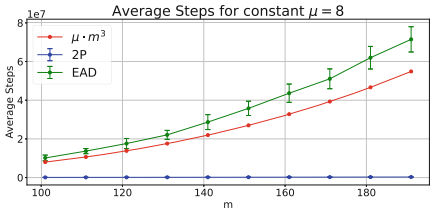
Our experiments were designed to explore the performance dynamics of the algorithms under two specific conditions: when the population size μ is held constant and when the number of edges m remains fixed.

Complete Bipartite Graphs. The starting condition for complete bipartite graphs involves a maximum matching where for each $0 \leq i \leq |R| - 1$,$r_i \in R$ is matched to $l_i \in L$, forming a homogeneous initial population. In the constant μ scenario, we increase the size of both L and R by one unit per iteration to maintain a steady $|L| - |R|$ difference, allowing a controlled analysis of the algorithms' scalability. In the constant m scenario we simply increase μ by one per iteration.

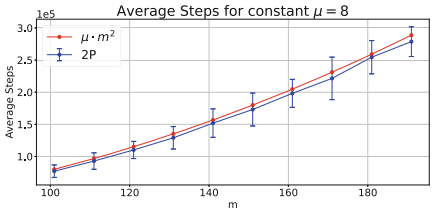
Paths. For paths, the initial population comprises maximum matchings including all even-indexed edges. With a fixed μ, the number of edges is incrementally increased by ten in each iteration, in order to cover a wider set of problem sizes, while staying experimentally feasible. In the constant m case, out of feasibility, we simply increase μ by one per iteration.

5.2 Methodology

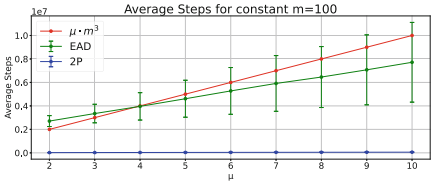
Each experiment was conducted 30 times to determine the average number of iterations and the standard deviation, estimating the algorithms' asymptotic runtime for both fixed population size (μ) and a fixed number of edges (m). For complete bipartite graphs and fixed m we chose $|L| = 24$ and $|R| = 23$ for the small gap case and $|L| = 34$ and $|R| = 23$ for the big gap case, such that the number of edges $m = 782$ for the small gap case and $m = 756$ for the big gap case are comparable in size.



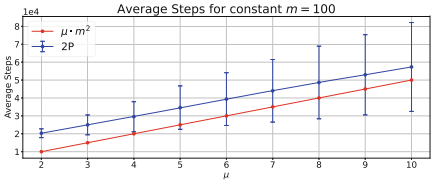
(a) EA_D and 2P with fixed $\mu = 8$ in paths



(b) 2P with fixed $\mu = 8$ in paths



(c) EA_D and 2P with fixed $m = 100$ in paths



(d) 2P with fixed $m = 100$ in paths

Fig. 2. Experimental results on paths

5.3 Complete Bipartite Graphs

This subsection focuses on the performance of evolutionary diversity algorithms on complete bipartite graphs, specifically examining the $(\mu + 1)$-EA_D and 2P-EA_D algorithms.

In Fig. 1a, we show the average number of iterations for a fixed population size of $\mu = 8$ and different values of $|L| - |R|$. Specifically, we examine cases where the difference $|L| - |R|$ is either 1, referred to as the 'small gap' scenario or $\mu + 1$, the 'big gap' scenario. The $(\mu + 1)$-EA_D algorithm presented a quadratic growth in m for the big gap case in iterations, empirically estimated as μm^2, suggesting an out-performance by a factor of approximately $\mu \log(m)$ over the theoretical bound. For the small gap case we empirically estimate the run time as $\mu m^{2.5}$, an even stronger suggested out-performance by a factor of $\mu m^{1.5} \log(m)$ when compared against the theoretical bound of $O(\mu^2 m^4 \log(m))$.

In Fig. 1c, we display the average iteration counts for a constant edge count m, considering the same values of $|L| - |R|$. These findings echo the trends observed in

Fig. 1a, showcasing how the algorithm's behavior remains consistent across different graph sizes and population disparities.

In Fig. 1b for μ fixed and Fig. 1d for m fixed, we zoom in on the results for the 2P-EA$_D$ algorithm. For both the small and big gap case the 2P-EA$_D$ algorithm exhibited a linear increase in the number of iterations with respect to m when μ was held constant and vice versa. Empirically, the run time for 2P-EA$_D$ was observed to be close to μm, a notable deviation from the predicted $O(\mu^2 m \log(m))$. The results summarized in Table 1 provide a summary of these observations. It is evident that the performance of the 2P-EA$_D$ algorithm is not only superior in practice but also suggests that our theoretical bounds may be refined to more closely predict the empirical outcomes.

Table 1. Summary of results for complete bipartite graphs

Algo.	$\lvert L\rvert - \lvert R\rvert > \mu$		$\lvert L\rvert - \lvert R\rvert \leq \mu$	
	Empirical	Theor. UB	Empirical	Theor. UB
EA$_D$	$\sim \mu m^2$	$O(\mu^2 m^2 \log(m))$	$\sim \mu m^{2.5}$	$O(\mu^2 m^4 \log(m))$
2P	$\sim \mu m$	$O(\mu^2 n^2 \log(n))$	$\sim \mu m$	$O(\mu^2 m^2 \log(m))$

Table 2. Summary of results for paths

Algorithm	Empirical	Theor. UB
EA$_D$	$\sim \mu m^3$	$O(\mu^3 m^3)$
2P	$\sim \mu m^2$	$O(\mu^3 m^2)$

5.4 Paths

This subsection focuses on the performance of evolutionary diversity algorithms on paths, specifically examining the $(\mu+1)$-EA$_D$ and 2P-EA$_D$ algorithms.

In Fig. 2a, we present the average number of iterations when the population size μ is fixed at 8. The graph illustrates how the number of iterations required for convergence changes as the number of edges m in the path increases. Figure 2c shows the average number of iterations for a fixed number of edges $m = 100$ and varying population size μ. For the $(\mu+1)$-EA$_D$ algorithm, a trend of polynomial growth in the number of iterations is observed as a function of the problem size. When μ is fixed at 8, the empirical runtime grows in line with μm^3, which could indicate a performance better than the theoretical upper bound of $O(\mu^3 m^3)$ by a factor of μ^2.

When we examine the 2P-EA$_D$ algorithm in Fig. 2b for a fixed μ, and in Fig. 2d for a fixed m, we notice a similar pattern. The empirical runtime for the 2P-EA$_D$ is consistently around μm^2, also possibly deviating by a factor of μ^2 from the theoretical $O(\mu^3 m^2)$ bound. The results in Table 2 provide a summary of these observations. It is evident that the performance of the 2P-EA$_D$ algorithm is not only superior in practice but also suggests that our theoretical bounds may be refined to more closely predict the empirical outcomes.

6 Conclusions

In this study, we explored the application of evolutionary algorithms (EAs) for maximizing diversity in solving the maximum matching problem in complete bipartite graphs and paths. Our methodology was structured into two distinct phases: a rigorous theoretical analysis followed by comprehensive empirical evaluations. We specifically looked at the $(\mu + 1)$-EA$_D$ and the Two-Phase Matching Evolutionary Algorithm (2P-EA$_D$), finding that both could achieve maximal diversity in expected polynomial time, with 2P-EA$_D$ showing a speed advantage in all scenarios. Our findings not only underscore the utility of EAs in combinatorial diversity problems but also open up avenues for further research. A significant future direction would be to refine the theoretical upper bounds of these algorithms' runtime. Additionally, applying these insights to other graph problems and exploring real-world applications, could provide practical benefits.

Acknowledgements. This work has been supported by the Australian Research Council through grant DP190103894.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alvarez, A., Dahlskog, S., Font, J.M., Togelius, J.: Empowering quality diversity in dungeon design with interactive constrained map-elites. In: IEEE Conference on Games, CoG 2019, pp. 1–8. IEEE (2019). https://doi.org/10.1109/CIG.2019.8848022
2. Bossek, J., Neumann, A., Neumann, F.: Breeding diverse packings for the knapsack problem by means of diversity-tailored evolutionary algorithms. In: Chicano, F., Krawiec, K. (eds.) GECCO 2021: Genetic and Evolutionary Computation Conference, Lille, France, 10–14 July 2021, pp. 556–564. ACM (2021). https://doi.org/10.1145/3449639.3459364
3. Bossek, J., Neumann, F.: Evolutionary diversity optimization and the minimum spanning tree problem. In: Chicano, F., Krawiec, K. (eds.) GECCO 2021: Genetic and Evolutionary Computation Conference, Lille, France, 10–14 July 2021, pp. 198–206. ACM (2021). https://doi.org/10.1145/3449639.3459363
4. Bossens, D.M., Tarapore, D.: QED: using quality-environment-diversity to evolve resilient robot swarms. IEEE Trans. Evol. Comput. **25**(2), 346–357 (2021). https://doi.org/10.1109/TEVC.2020.3036578
5. Cully, A., Demiris, Y.: Quality and diversity optimization: a unifying modular framework. IEEE Trans. Evol. Comput. **22**(2), 245–259 (2018). https://doi.org/10.1109/TEVC.2017.2704781
6. Do, A.V., Bossek, J., Neumann, A., Neumann, F.: Evolving diverse sets of tours for the travelling salesperson problem. In: Coello, C.A.C. (ed.) GECCO 2020: Genetic and Evolutionary Computation Conference, Cancún Mexico, 8–12 July 2020, pp. 681–689. ACM (2020). https://doi.org/10.1145/3377930.3389844
7. Do, A.V., Guo, M., Neumann, A., Neumann, F.: Analysis of evolutionary diversity optimization for permutation problems. ACM Trans. Evol. Learn. Optim. **2**(3), 11:1–11:27 (2022). https://doi.org/10.1145/3561974

8. Do, A.V., Guo, M., Neumann, A., Neumann, F.: Diverse approximations for monotone submodular maximization problems with a matroid constraint. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5558–5566. ijcai.org (2023). https://doi.org/10.24963/IJCAI.2023/617
9. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**(4), 673–697 (2012). https://doi.org/10.1007/S00453-012-9622-X
10. Friedrich, T., Horoba, C., Neumann, F.: Illustration of fairness in evolutionary multi-objective optimization. Theor. Comput. Sci. **412**(17), 1546–1556 (2011). https://doi.org/10.1016/J.TCS.2010.09.023
11. Friedrich, T., Oliveto, P.S., Sudholt, D., Witt, C.: Analysis of diversity-preserving mechanisms for global exploration. Evol. Comput. **17**(4), 455–476 (2009). https://doi.org/10.1162/EVCO.2009.17.4.17401
12. Gao, W., Nallaperuma, S., Neumann, F.: Feature-based diversity optimization for problem instance classification. Evol. Comput. **29**(1), 107–128 (2021). https://doi.org/10.1162/evco_a_00274
13. Gao, W., Pourhassan, M., Neumann, F.: Runtime analysis of evolutionary diversity optimization and the vertex cover problem. In: Silva, S., Esparcia-Alcázar, A.I. (eds.) Genetic and Evolutionary Computation Conference, GECCO 2015, Companion Material Proceedings, pp. 1395–1396. ACM (2015). https://doi.org/10.1145/2739482.2764668
14. Gounder, S., Neumann, F., Neumann, A.: Evolutionary diversity optimisation for sparse directed communication networks. In: Genetic and Evolutionary Computation Conference, GECCO 2024. ACM (2024, to appear)
15. Gravina, D., Khalifa, A., Liapis, A., Togelius, J., Yannakakis, G.N.: Procedural content generation through quality diversity. In: IEEE Conference on Games, CoG 2019, London, United Kingdom, 20–23 August 2019, pp. 1–8. IEEE (2019). https://doi.org/10.1109/CIG.2019.8848053
16. Harder, J.G., Neumann, A., Neumann, F.: Analysis of evolutionary diversity optimisation for the maximum matching problem (2024). https://arxiv.org/abs/2404.11784
17. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Nat. Comput. **3**(1), 21–35 (2004). https://doi.org/10.1023/B:NACO.0000023417.31393.C7
18. Hopcroft, J.E., Karp, R.M.: An N5/2 algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**(4), 225–231 (1973). https://doi.org/10.1137/0202019
19. Mouret, J.B., Clune, J.: Illuminating search spaces by mapping elites. arXiv preprint arXiv:1504.04909 (2015)
20. Neumann, A., Antipov, D., Neumann, F.: Coevolutionary pareto diversity optimization. In: GECCO 2022: Genetic and Evolutionary Computation Conference, pp. 832–839. ACM (2022). https://doi.org/10.1145/3512290.3528755
21. Neumann, A., Bossek, J., Neumann, F.: Diversifying greedy sampling and evolutionary diversity optimisation for constrained monotone submodular functions. In: GECCO 2021: Genetic and Evolutionary Computation Conference, pp. 261–269. ACM (2021). https://doi.org/10.1145/3449639.3459385
22. Neumann, A., Gao, W., Doerr, C., Neumann, F., Wagner, M.: Discrepancy-based evolutionary diversity optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 991–998. ACM (2018). https://doi.org/10.1145/3205455.3205532
23. Neumann, A., Gao, W., Wagner, M., Neumann, F.: Evolutionary diversity optimization using multi-objective indicators. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 837–845. ACM (2019). https://doi.org/10.1145/3321707.3321796

24. Neumann, A., et al.: Diversity optimization for the detection and concealment of spatially defined communication networks. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 1436–1444. ACM (2023). https://doi.org/10.1145/3583131.3590405
25. Neumann, F., Witt, C.: Bioinspired computation in combinatorial optimization: algorithms and their computational complexity. In: Blum, C., Alba, E. (eds.) Genetic and Evolutionary Computation Conference, GECCO 2013, pp. 567–590. ACM (2013). https://doi.org/10.1145/2464576.2466738
26. Nikfarjam, A., Bossek, J., Neumann, A., Neumann, F.: Computing diverse sets of high quality TSP tours by eax-based evolutionary diversity optimisation. In: FOGA 2021: Foundations of Genetic Algorithms XVI, pp. 9:1–9:11. ACM (2021). https://doi.org/10.1145/3450218.3477310
27. Nikfarjam, A., Bossek, J., Neumann, A., Neumann, F.: Entropy-based evolutionary diversity optimisation for the traveling salesperson problem. In: GECCO 2021: Genetic and Evolutionary Computation Conference, pp. 600–608. ACM (2021). https://doi.org/10.1145/3449639.3459384
28. Nikfarjam, A., Neumann, A., Neumann, F.: Evolutionary diversity optimisation for the traveling thief problem. In: GECCO 2022: Genetic and Evolutionary Computation Conference, pp. 749–756. ACM (2022). https://doi.org/10.1145/3512290.3528862
29. Nikfarjam, A., Neumann, A., Neumann, F.: On the use of quality diversity algorithms for the traveling thief problem. In: GECCO 2022: Genetic and Evolutionary Computation Conference, pp. 260–268. ACM (2022). https://doi.org/10.1145/3512290.3528752
30. Pugh, J.K., Soros, L.B., Stanley, K.O.: Quality diversity: a new frontier for evolutionary computation. Front. Robot. AI **3**, 40 (2016). https://doi.org/10.3389/FROBT.2016.00040
31. Ulrich, T., Bader, J., Thiele, L.: Defining and optimizing indicator-based diversity measures in multiobjective search. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 707–717. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_71
32. Ulrich, T., Thiele, L.: Maximizing population diversity in single-objective optimization. In: Krasnogor, N., Lanzi, P.L. (eds.) 13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, 12–16 July 2011, pp. 641–648. ACM (2011). https://doi.org/10.1145/2001576.2001665
33. Vassiliades, V., Chatzilygeroudis, K., Mouret, J.B.: Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. IEEE Trans. Evol. Comput. **22**(4), 623–630 (2017)
34. Zhang, H., Chen, Q., Xue, B., Banzhaf, W., Zhang, M.: Map-elites for genetic programming-based ensemble learning: an interactive approach [AI-explained]. IEEE Comput. Intell. Mag. **18**(4), 62–63 (2023). https://doi.org/10.1109/MCI.2023.3304085

Archive-Based Single-Objective Evolutionary Algorithms for Submodular Optimization

Frank Neumann[1(✉)] and Günter Rudolph[2]

[1] Optimisation and Logistics, School of Computer and Mathematical Sciences, The University of Adelaide, Adelaide, Australia
frank.neumann@adelaide.edu.au

[2] Computational Intelligence, Department of Computer Science, TU Dortmund University, Dortmund, Germany

Abstract. Constrained submodular optimization problems play a key role in the area of combinatorial optimization as they capture many NP-hard optimization problems. So far, Pareto optimization approaches using multi-objective formulations have been shown to be successful to tackle these problems while single-objective formulations lead to difficulties for algorithms such as the $(1 + 1)$-EA due to the presence of local optima. We introduce for the first time single-objective algorithms that are provably successful for different classes of constrained submodular maximization problems. Our algorithms are variants of the $(1 + \lambda)$-EA and $(1+1)$-EA and increase the feasible region of the search space incrementally in order to deal with the considered submodular problems.

Keywords: evolutionary algorithms · submodular optimization · runtime analysis · theory

1 Introduction

Many combinatorial optimization problems that face diminishing returns can be stated in terms of a submodular function under given set of constraints [7]. The maximization of a non-monotone submodular function even without constraints includes the classical maximum cut problem in graphs and is therefore an NP-hard combinatorial optimization problem that cannot be solved in polynomial time unless $P = NP$ but different types of approximation algorithms are available [2]. Monotone submodular functions play a special role in the area of optimization as they capture import coverage and influence maximization problems in networks. The maximization of monotone submodular functions is NP-hard even for the case of simple constraint that limits the number of elements that can be chosen, but greedy algorithms have shown to obtain best possible approximation guarantees for different types of constraints [7,8]. At best, one can hope to develop a method that can provide an α-approximation in polynomial time, i.e., a solution with a value of at least $\alpha\, f(x^*)$ where $\alpha \in (0, 1)$ and x^* is an optimal solution of the submodular function $f(\cdot)$. Such an algorithm was proposed in [8]

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 166–180, 2024.
https://doi.org/10.1007/978-3-031-70071-2_11

where it was proved that a greedy method can find an $(1 - 1/e)$-approximation of the maximum of a submodular function in polynomial time.

Although the (1+1)-EA shares many characteristics with a greedy algorithm, it was proven in [4, Thm. 1], that it can get trapped in local optima even for monotone submodular problems with a uniform constraint requiring exponential time to achieve an approximation better than $1/2 + \varepsilon$ for any given $\varepsilon > 0$.

Due to this disappointing result, the focus shifted to other types of evolutionary algorithms. Since multiobjective EAs have proven successful in the treatment of combinatorial problems in the past [3,6,10], the variant GSEMO [5] has been applied to the maximiziation of a submodular function with cardinality constraint. Guided by the proof in [8] it was proven in [4] that the GSEMO can find a $(1 - 1/e)$-approximation in polynomial time with small failure probability. In the sequel there have been several publications in this direction considering different variants of GSEMO together with appropriate muli-objective formulations treating the considered constraint as an additional objective [1,9,12–15].

Recently, the *sliding window* GSEMO (SW-GSEMO) has been introduced in [11] which outscores the performance of the original GSEMO significantly of large problem instances. The improvement here comes from a sliding window selection method that selects parent individuals dependent on the anticipated progress in time during the optimization run. Motivated by the insights gained through the development of SW-GSEMO, we show that singleobjective EAs with only few small algorithmic changes to the standard versions are able to achieve the same theoretical and competitive practical performance as their multiobjective counterpart, i.e., it is not necessary to apply *multiobjective* EAs. This result potentially opens a new area in the development of evolutionary algorithms for submodular problems which has so far relied on the use of multi-objective problem formulations and algorithms.

The outline of the paper is as follows. In Sect. 2 we introduce terminology and basic results regarding submodular functions. Section 3 provides theoretical results for single-objective EAs without archive, whereas Sect. 4 presents the proof that an (1+1)-EA can successfully solve the submodular problem if it is equipped with a specific kind of archive. The theoretical findings are supported by experimental results on graph cover problems in Sect. 5. Finally, Sect. 6 reports on our conclusions.

2 Preliminaries

Definition 1. *Let U be a finite ground set and $f : 2^U \to \mathbb{R}_0^+$. If for all $A, B \subseteq U$ holds*

a) $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$ then f is termed submodular*;*
b) $f(A) \leq f(B)$ then f is called monotone*.*

Many functions arising in combinatorial optimization are submodular. For example, let $A_1, \ldots, A_n$ be subsets of a finite universe U. Then the *coverage function* $f(S) = |\cup_{i \in S} A_i|$ with $S \subseteq \{1, \ldots, n\}$ is submodular. Submodular

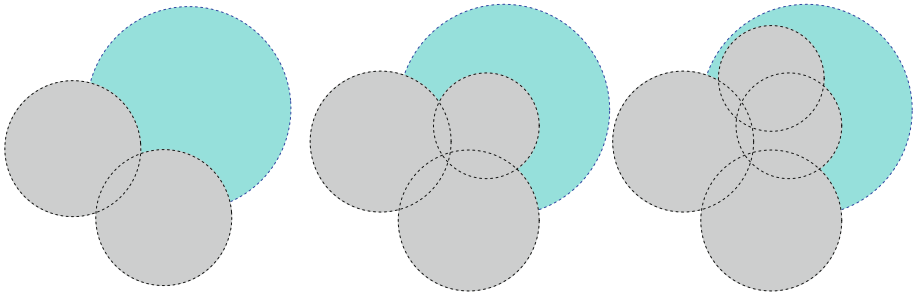


Fig. 1. Gray set A in the left figure is a subset of the gray set B in the middle figure which in turn is a subset of the gray set C in the right figure. Adding the blue set leads to a lower gain of area the larger the gray set is. (Color figure online)

functions are also called functions of diminishing returns, as demonstrated in Fig. 1: The later we add the blue set to the increasing gray set, the smaller is the gain of area.

Theorem 1 (see [8], Proposition 2.1). *The following conditions are equivalent to the definition of submodular set functions:*

a) for all $A \subseteq B \subseteq U$ *and* $x \notin B$

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$$

b) for all $A \subseteq B \subseteq U$

$$f(B) \leq f(A) + \sum_{x \in B \setminus A} (f(A \cup \{x\}) - f(A))$$

For later purpose we present an auxiliary result which can be established from well-known properties of monotone submodular functions [7,8]. We present it here together with the proof as it is crucial for optimizing monotone submodular functions with a uniform constraint.

Lemma 1. *Let* f *be a monotone submodular set function,* X^* *an optimal solution and* X *some feasible solution. Then*

$$f(X^*) \leq f(X) + r\,\delta \tag{1}$$

where $\delta = \max_{x \in X^* \setminus X} (f(X \cup \{x\}) - f(X))$ *and* $r = |X^*|$.

Proof. Note that we have $f(X^*) \leq f(X \cup X^*)$ since f is monotone. Let

$$\delta = \max_{x \in X^* \setminus X} (f(X \cup \{x\}) - f(X))$$

be the largest marginal gain among all elements in $X^* \setminus X$. We have

$$\begin{aligned} f(X^*) &\leq f(X \cup X^*) \\ &\leq f(X) + \sum_{x \in X^* \setminus X} (f(X \cup \{x\}) - f(X)) \qquad (2) \\ &\leq f(X) + |X^*| \cdot \delta \\ &= f(X) + r \cdot \delta. \end{aligned}$$

where inequality (2) follows from Theorem 1(b). □

Definition 2. *Let $c : 2^U \to \mathbb{R}_+$ with $|U| = n < \infty$ and budget $B > 0$. The constraint $c(X) \leq B$ is termed a* cardinality *or* uniform constraint *if $c(X) = |X|$ for $X \in 2^U$ and $B \leq n$. Otherwise it is called a* general constraint*.*

Definition 3. *The maximization of a monotone submodular function under a given constraint is termed the* monotone submodular maximization problem *(MSMP).*

Definition 4. *The submodularity ratio α_f of a non-negative set function f is defined as $\alpha_f = \min_{X \subseteq Y \subseteq U, v \notin Y} \frac{f(X \cup v) - f(X)}{f(Y \cup v) - f(Y)}$.*

A function f is submodular iff $\alpha_f = 1$ holds. In Sect. 4, we will consider general monotone objective and monotone cost functions and investigate approximations dependent of α_f.

When considering evolutionary algorithms for the optimization of submodular function, we work with the search space $\{0,1\}^n$, i.e. search points are binary strings of length n. We identify each element $u_i \in U$ with a bit x_i, $1 \leq i \leq n$, and define the set $X \subseteq U$ as $X = \{u_i \in U \mid x_i = 1\}$. To ease the presentation we use the search point x and its set of chosen elements X in an interchangeable way.

3 $(1 + \lambda)$-EA Without Archive

We first consider the case of the optimization of a monotone submodular functions with a uniform constraint, i.e. $|x|_1 = \sum_{i=1}^{n} x_i \leq B$ holds for any feasible solution $x \in \{0,1\}^n$, before we consider the case of general constraints.

3.1 Algorithm

The $(1 + \lambda)$-EA always starts at the zero string $x_j = 0^n$ where $j = 0, 1, \ldots B$ denotes the jth epoch on the way to reach the bound B. The current bound $\hat{B}$ is set to zero initially. It is clear that x_0 is feasible.

In each epoch $j \geq 0$ the $(1 + \lambda)$-EA samples λ offspring independently by mutation. For our theoretical investigations, we consider standard-bit-mutation which flip each bit independently of the others with probability $1/n$. For our

experimental investigations, we consider standard-bit-mutation-plus as done in [11] which repeats standard-bit-mutation until at least one bit has been flipped. We are seeking the best solution for the incremented bound $\hat{B}$. If an offspring y is feasible and not worse than its parent x_j then it is accepted as a candidate for selection. After all λ offspring have been evaluated the best candidate becomes the new best individual of epoch $j+1$.

This process repeats until the current bound $\hat{B}$ exceeds the maximum bound B. The $(1+\lambda)$-EA is given in Algorithm 1 in case of uniform constraint.

Algorithm 1 $(1+\lambda)$ EA, input: f, c, B, λ

```
Set j := 0, x_j := 0^n, B̂ := 0
while B̂ < B do
   B̂ = B̂ + 1
   x̂ = x_j
   for k := 1 to λ do
      y := mutation(x_j)
      if c(y) ≤ B̂ and f(y) ≥ f(x̂) then
         x̂ := y
      end if
   end for
   x_{j+1} := x̂
   j := j + 1
end while
return x_j
```

3.2 Uniform Constraint

Theorem 2. *The $(1+\lambda)$-EA finds a $(1-\frac{1}{e})$-approximation of a monotone submodular maximization problem with uniform constraint in at most $t_{\max} = 2\,e\,r\,n\log(n)$ function evaluations with probability $1-o(1)$, where $\lambda = 2\,e\,n\log(n)$, r is equal to the maximum budget in the constraint and n is the dimension of the problem.*

Proof. The proof is oriented at the proofs of Theorem 1 in [11] and Theorem 2 in [4] with adaptation to the context of the $(1+\lambda)$-EA.

Let x^* be the optimal solution and $f(x^*)$ denote the global maximum. Assume that at each epoch $j = 0, 1, \ldots, r$ the EA has found a solution x_j with at most j elements such that

$$f(x_j) \geq \left[1 - \left(1 - \frac{1}{r}\right)^j\right] \cdot f(x^*). \tag{3}$$

If the assumption is true, then x_r has the desired approximation ratio as can be seen from

$$f(x_r) \geq \left[1 - \left(1 - \frac{1}{r}\right)^r\right] \cdot f(x^*) \geq \left(1 - \frac{1}{e}\right) \cdot f(x^*).$$

Therefore we have to establish the validity of inequality (3) for all $r = 0, 1 \ldots, r$, which is done by induction.

We begin with the feasible solution $x_j = (0, \ldots, 0)$ at epoch $j = 0$. Evidently, inequality (3) is fulfilled since $f(x_0) \geq 0$. Now assume that x_j is the current solution at time $\tau_j = j \cdot \lambda$ for $j = 0, 1, \ldots, r-1$. Now we can make λ trials to find the best feasible improvement by mutation. For the best feasible improvement only a single specific bit mutation from 0 to 1 is necessary. As a consequence, the probability to transition from x_j to x_{j+1} in a single trial is lower bounded via

$$\mathsf{P}\{x_j \to x_{j+1} \text{ in single trial}\} = \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{e\,n}. \tag{4}$$

As a consequence, the probability that the transition to x_{j+1} does not happen in λ trials is upper bounded by

$$\begin{aligned}\mathsf{P}\{x_j \not\to x_{j+1} \text{ in } \lambda \text{ trials}\} &\leq \left(1 - \frac{1}{e\,n}\right)^{\lambda} = \left(1 - \frac{1}{e\,n}\right)^{2\,e\,n\,\log(n)} \\ &= \left[\left(1 - \frac{1}{e\,n}\right)^{e\,n}\right]^{2\,\log(n)} \leq e^{-2\,\log(n)} = \frac{1}{n^2}.\end{aligned}$$

Owing to the above bound and Boole's inequality we finally obtain

$$\mathsf{P}\left\{\bigcup_{j=1}^{r}\{x_j \text{ not generated at } \tau_j\}\right\} \leq \sum_{j=1}^{r}\mathsf{P}\{x_j \text{ not generated at } \tau_j\} \leq \frac{r}{n^2} \leq \frac{1}{n}$$

since $r \leq n$. This bound on the failure probability proves the success probability $1 - o(1)$ in the statement of the theorem.

It remains to prove the induction step. According to (1) in Lemma 1 we have $f(x^*) \leq f(x_i) + r\,\delta_{i+1} \Leftrightarrow \delta_{i+1} \geq \frac{1}{r}\,(f(x^*) - f(x_i))$. It follows that

$$\begin{aligned}f(x_{j+1}) &\geq f(x_j) + \frac{1}{r}\,(f(x^*) - f(x_j)) \\ &= f(x_j)\left(1 - \frac{1}{r}\right) + \frac{1}{r}\,f(x^*) \\ &\geq \left(1 - \left(1 - \frac{1}{r}\right)^{j}\right) \cdot f(x^*) \cdot \left(1 - \frac{1}{r}\right) + \frac{1}{r}\,f(x^*) \qquad (5) \\ &= \left(1 - \frac{1}{r}\right) f(x^*) - \left(1 - \frac{1}{r}\right)^{j+1} f(x^*) + \frac{1}{r}\,f(x^*) \\ &= f(x^*)\left(1 - \left(1 - \frac{1}{r}\right)^{j+1}\right)\end{aligned}$$

where (5) results from inserting the induction hypothesis. □

3.3 General Constraint

The $(1 + \lambda)$-EA does not work in the more general case where the constraint is given by a linear function or the general cost function considered in [11,12] as can be observed by the following example instance of the classical knapsack problem. Consider the knapsack problem where each item i has weight w_i and profit p_i and the sum of the weight of the chosen items in any feasible solution is at most B. Assume for items i, $1 \leq i \leq n-1$, we have $w_i = 1$ and $p_i = 1$ and for item n we have $w_n = n - 1$ and $p_n = L$ where L is a large value, e.g. $L = 2^n$. We set $B = n - 1$. The optimal solution consists of the item n only and any solution not chosen item n has profit at most $n - 1$. Note that choosing item n plus any other item leads to an infeasible solutions. Hence, any feasible solution that is not optimal has approximation ratio at most n/L which is $n/2^n$ for $L = 2^n$. The $(1 + \lambda)$-EA starts with the solution 0^n and increases the bound iteratively. Only once $\hat{B} = n - 1$ holds, item n may be introduced. However, once $\hat{B} = n-1$ holds, a large number of the first $n-1$ items has been introduced which prevents element n from being inserted. Inserting element n is then only possible of all other elements are removed in the same mutation step. This leads with high probability to an exponential runtime for obtaining the optimal solution in the case of $\lambda = 2en \log n$ as chosen in Theorem 2. Even for $\lambda = 1$, the algorithm would have included a constant fraction of the first $n-1$ elements before $\hat{B} = n-1$ holds which again implies an exponential optimization time with high probability for $\lambda = 1$. The arguments can be generalized to show an exponential optimization time with high probability for the $(1 + \lambda)$-EA as defined in Algorithm 1 for any $\lambda \geq 1$.

4 (1 + 1)-EA with Archive

We now consider the case of general (monotone) objective function f and cost function c as already investigated in [11,12] for variants of GSEMO using multi-objective formulations. Recall that the submodularity ratio α_f of a given function f measures how close the function is of being submodular (see Definition 4).

4.1 Algorithm

For the general setting, we consider a variant of the classical $(1 + 1)$-EA. The algorithm is shown in Algorithm 2. It starts with the solution $x_0 = 0^n$ and a constraint bound $\hat{B} = 0$. As for $(1+\lambda)$-EA, we use standard-bit-mutation for our theoretical investigations and standard-bit-mutation-plus in the experiments. For t_{epoch} iterations, the single solution is improved under the current bound and solutions that are currently infeasible but still meet the bound B of the given problem are added to an archive A. After the current epoch is finished the bound $\hat{B}$ is increased by 1 and it is checked whether the archive contains a solution feasible for the updated bound that is better than the current solution $\hat{x}$ of the algorithm. If so, the current solution $\hat{x}$ is updated with the best (now)

Algorithm 2 $(1+1)$-EA with archive, input: $f, c, B, t_{\max}$

Set $j := 0, x_j := 0^n$, $\hat{B} := 0$, $A := \emptyset$
$t_{\text{epoch}} = \lfloor (t_{\max}/(\lceil B \rceil)) \rfloor$
while $(\hat{B} \leq B) \wedge (t < t_{\max})$ **do**
  for $(k := 1, (k \leq t_{\text{epoch}}) \wedge (t < t_{\max}), k := k + 1)$ **do**
    $y :=$ mutation(x_j)
    $t := t + 1$
    if $(c(y) > \hat{B}) \wedge (c(y) \leq B)$ **then**
      if $\nexists z \in A : (c(z) \leq c(y) \wedge f(z) > f(y))$ **then**
        $A := A \cup \{y\}$
      end if
    end if
    if $(c(y) \leq \hat{B}) \wedge (f(y) \geq f(x_j))$ **then**
      $x_j := y$
    end if
  end for
  $A := A \setminus \{y \in A : c(y) \leq \hat{B}\}$
  $\hat{B} := \min\{\hat{B} + 1, B\}$
  $\hat{x} := x_j$
  $A^* := \{y \in A : c(y) \leq \hat{B}\}$
  if $|A^*| > 0$ **then**
    $y^* := \arg\max\{f(y) : y \in A^*\}$
    if $f(y^*) \geq f(\hat{x})$ **then**
      $\hat{x} := y^*$
    end if
  end if
  $j := j + 1$
  $x_j := \hat{x}$
end while
return x_j

feasible solution that can be found in the archive. The algorithm then proceeds with the next epoch consisting of t_{epoch} steps for the updated bound and does so until t_{epoch} steps have finally been carried out for the bound B of the given problem. For our theoretical investigations, t_{epoch} is the crucial parameter for the success probability and we assume that B is a positive integer. Hence, we will mainly concentrate on t_{epoch} as parameter and the total number of iterations $t_{\max}$ can be obtained by considering that in total B epochs of length t_{epoch} are carried out. For our experiments the algorithm works with $t_{\max}$ as an input and divides it (roughly) equally among the epochs. During the run, the algorithm stores at $x_{\hat{B}}$ the best feasible solution that it obtains for budget $\hat{B}$, $0 \leq \hat{B} \leq B$, and finally returns x_B as the solution to the given problem with budget constraint B.

4.2 Analysis

Let $x^*_{\hat{B}}$ be an optimal solution for a reduced budget $R(B, c)$ dependent on the characteristics of the monotone cost function c and given budget $\hat{B}$, $0 \leq \hat{B} \leq B$.

For details on the budget reduction, we refer the reader to Eq. 4 in [12]. As done in [11], we assume that $c\colon \{0,1\}^n \to \mathbb{N}$ takes on non-negative integer values for our analysis. We show that for each $\hat{B} \in \{0,\ldots,B\}$, $(1+1)$-EA with archive computes a solution $x_{\hat{B}}$ with

$$c(x_{\hat{B}}) \leq \hat{B} \text{ and } f(x_{\hat{B}}) \geq \frac{\alpha_f}{2}(1-e^{-\alpha_f}) \cdot f(x^*_{\hat{B}})$$

where $x^*_{\hat{B}}$ is an optimal solution for budget $R(B,c)$. Note that this matches the results given in Theorem 5 in [15] where it is shown that GSEMO computes for any possible budget up to the given budget B a solution of the stated quality.

We now show that a $(1+1)$-EA using an archive for the solutions that exceed the current bound $\hat{B}$ (but have cost at most B) is able to obtain the same approximation ratio as the multi-objective approaches presented in [11,12].

Let $\delta_c = \min_{V' \subseteq V} \min_{v \notin V'}(c(V' \cup \{v\}) - c(V') \geq 1$ the minimal possible cost increase when adding one element to any set not containing the element.

We use $t_{\text{epoch}} = en\ln(nB^2)$ which implies $t_{\max} \leq (B+1) \cdot t_{\text{epoch}} = en(B+1)\ln(nB^2) = O(nB(2\ln B + \ln n))$.

We denote by $x^*_{\hat{B}}$ an optimal solution for the reduced constraint bound $R(B,c)$ dependent on $\hat{B}$ and characteristics of the constraint function c as done in [11,12].

Theorem 3. *Let $t_{\max} = B \cdot t_{\text{epoch}}$ where $t_{\text{epoch}} \geq en\ln(nB^2)$. Then $(1+1)$-EA with archive computes with probability $1-o(1)$ for each bound $\hat{B}$, $0 \leq \hat{B} \leq B$, a solution $x_{\hat{B}}$ with*

$$c(x_{\hat{B}}) \leq \hat{B} \text{ and } f(x_{\hat{B}}) \geq (\alpha_f/2) \cdot (1-e^{-\alpha_f}) \cdot f(x^*_{\hat{B}})$$

In particular it computes a $(\alpha_f/2)(1-e^{-\alpha_f})$-approximation with probability $1-o(1)$ for the given bound B when setting $t_{\max} = B \cdot t_{\text{epoch}}$ with $t_{\text{epoch}} \geq en\ln(nB)$.

Proof. We start with some observations. For each bound B', $0 \leq B' \leq B$ the algorithm maintains a solution x with $c(x) \leq B'$ that has the highest function value among all solution of cost at most B' obtained during the run. This solution is either contained in $A \cup \{\hat{x}\}$ or stored at $x_{\hat{B}}$ when the bound $\hat{B} = B'$ is reached.

According to [12], for a given bound $\hat{B}$ a solution with the stated approximation quality can be obtained by selection the best out of two solutions.

Case 1: The first solution consists of the single element $v^* \in V$ of highest function value among all solution with a single element v for which $c(v) \leq \hat{B}$ holds. If such a $v^*_{\hat{B}}$ with $c(v^*) \leq \hat{B}$ is a $(\alpha_f/2)(1-1/e^{\alpha_f})$-approximation for bound $\hat{B}$, then it can be obtained by flipping the bit corresponding to $v^*_{\hat{B}}$ and no other bit in the initial solution 0^n. The probability for this to happen in a single mutation step applied to 0^n is at least $1/(en)$. The solution 0^n is chosen t_{epoch} times for mutation and the probability that a feasible solution for $\hat{B}$ with function value at least $f(v^*_{\hat{B}})$ has not been obtained is upper bounded by

$$(1-1/en)^{t_{\text{epoch}}} \leq (1-1/en)^{en\ln(nB^2)} \leq 1/(nB^2). \tag{6}$$

Using the union bound the probability that for at least one value of $\hat{B} \in \{1, \ldots, B\}$ such a solution has not been obtained is $1/(nB)$.

Case 2: If selecting element $v^*_{\hat{B}}$ only does not yield a $(\alpha_f/2)(1 - 1/e^{\alpha_f})$-approximation for bound $\hat{B}$, then a solution with the desired approximation quality can be obtained by incrementally adding an element with the largest marginal gain to the solution $y_{\hat{B}}$ with

$$c(y_{\hat{B}}) \leq C_{\hat{B}} \text{ and } f(y_{\hat{B}}) \geq \left(1 - e^{-\alpha_f C_{\hat{B}}/B}\right) \cdot f(x^*_{\hat{B}}). \tag{7}$$

Note that the search point 0^n meets the condition for $C_{\hat{B}} = 0$.

For a given solution x, let $N(x)$ be the set of all solutions that can be obtained from x by flipping a single 0-bit in x. Then we call

$$y = \arg \max_{z \in N(x)} (f(z) - f(x))/(c(z) - c(x))$$

a solution with the largest marginal gain with respect to x and the considered objective and cost function. The element contained in y but not in x is an element with the largest marginal gain. According to [12], adding an element with the largest marginal gain to $y_{\hat{B}}$ results in a solution $y'_{\hat{B}}$ with

$$\begin{aligned}
f(y'_{\hat{B}}) &\geq f(y_{\hat{B}}) + \alpha_f \cdot \frac{c(y'_{\hat{B}}) - c(y_{\hat{B}})}{\hat{B}} \cdot (f(x^*_{\hat{B}}) - f(y_{\hat{B}})) \\
&\geq \left(1 - \alpha_f \cdot \frac{\delta_{\hat{c}}}{\hat{B}}\right) f(y_{\hat{B}}) + \alpha_f \cdot \frac{\delta_{\hat{c}}}{\hat{B}} \cdot f(x^*_{\hat{B}}) \\
&\geq \left(1 - \alpha_f \cdot \frac{\delta_{\hat{c}}}{\hat{B}}\right) \left(1 - e^{-\alpha_f C_{\hat{B}}/\hat{B}}\right) \cdot f(x^*_{\hat{B}}) + \alpha_f \cdot \frac{\delta_{\hat{c}}}{\hat{B}} \cdot f(x^*_{\hat{B}}) \\
&\geq \left(1 - \left(1 - \alpha_f \cdot \frac{\delta_{\hat{c}}}{\hat{B}}\right) \cdot e^{-\alpha_f C_{\hat{B}}/B}\right) f(x^*_{\hat{B}}) \\
&\geq \left(1 - e^{-\alpha_f \cdot \frac{\delta_{\hat{c}}}{\hat{B}}} \cdot e^{-\alpha_f C_{\hat{B}}/\hat{B}}\right) f(x^*_{\hat{B}}) \\
&\geq \left(1 - e^{-\alpha_f (C_{\hat{B}} + \delta_{\hat{c}})/\hat{B}}\right) \cdot f(x^*_{\hat{B}})
\end{aligned}$$

Note that for $C_{\hat{B}} + \delta_{\hat{c}} \geq \hat{B}$, we have

$$f(y'_{\hat{B}}) \geq \left(1 - e^{-\alpha_f (C_{\hat{B}} + \delta_{\hat{c}})/\hat{B}}\right) \cdot f(\hat{x}_{\hat{B}}) \geq (1 - e^{-\alpha_f}) \cdot f(x^*_{\hat{B}}).$$

We consider the case when $c(y'_{\hat{B}}) > \hat{B}$. We have $f(v^*_{\hat{B}}) \geq \alpha_f \cdot (f(y'_{\hat{B}}) - f(y_{\hat{B}}))$ as f is α_f-submodular which implies

$$f(v^*_{\hat{B}}) + f(y_{\hat{B}}) \geq \alpha_f \cdot f(y'_{\hat{B}}) \geq \alpha_f \cdot \left(1 - e^{-\alpha_f}\right) \cdot f(x^*_{\hat{B}})$$

and therefore $\max\{f(v^*_{\hat{B}}), f(y_{\hat{B}}))\} \geq (\alpha_f/2) \cdot (1 - e^{-\alpha_f}) \cdot f(x^*_{\hat{B}})$. If $c(y'_{\hat{B}}) > \hat{B}$ and $v^*_{\hat{B}}$ is not a $(\alpha_f/2)(1 - e^{-\alpha_f})$-approximation for bound $\hat{B}$ then we have $f(y_{\hat{B}}) \geq (\alpha_f/2)(1 - e^{-\alpha_f}) \cdot f(x^*_{\hat{B}})$.

For $\hat{B} = 0$, the solution $x_0 = 0^n$ has the desired approximation quality. Let $B' \in \{1, \ldots, \hat{B}\}$ and consider the solution $y_{B'} \in A \cup \{x_j\}$ that meets the condition for the largest possible value of $C_{B'}$ according to Eq. 7.

We claim that the invariant $C_{B'} \geq \hat{B}$ holds during the run of the algorithm with high probability for any B' and $\hat{B}$ as long as the desired approximation has not been reached. For $\hat{B} = 0$, the solution $x_0 = 0^n$ obviously fullfills the condition for $C_{B'} = 0$ for any B'.

We consider the time when $\hat{B} = C_{B'}$ holds for the first time and show that $C_{B'}$ increases with high probability. Note that this is a pessimistic assumption as $C_{B'}$ might increase earlier by creating an offspring of the current solution x_j. We consider the epoch of the next t_{epoch} steps once reached $\hat{B} = C_{B'}$. Note that in this epoch only solutions x_j with $c(x_j) \leq C_{B'}$ and $f(x_j) \geq f(y_{\hat{B}})$ are chosen for mutation which implies that they meet the condition of Eq. 7 for $C_{B'}$.

The probability that there is no mutation step adding the element with the largest marginal gain to the current solution x_j and therefore increasing $C_{B'}$ by producing an offspring y (line 5 of Algorithm 2) by at least $\delta_c \geq 1$ is at most $1/(nB^2)$ (see Eq. 6).

The value of $C_{B'}$ needs to be increase at most B times and hence we obtain a solution $y_{B'}$ with $f(y_{B'}) \geq (\alpha_f/2)(1 - e^{-\alpha_f}) \cdot f(x^*_{B'})$ with probability at least $1 - 1/(nB)$ if element $v^*_{B'}$ from Case 1 does not give the desired approximation.

There are B different values of B' which implies that for all values of $\hat{B}$, $0 \leq \hat{B} \leq B$, a solution $x_{\hat{B}}$ with $f(x_{\hat{B}}) \geq (\alpha_f/2)(1 - e^{-\alpha_f}) \cdot f(x^*_{\hat{B}})$ is obtained with probability at least $(1 - 1/n - 1/(nB) \geq 1 - 2/n$ within the run of the algorithm. If the case where we are only interested in the desired approximation for the given bound B, choosing $t_{\max} = B \cdot t_{\text{epoch}}$ with $t_{\text{epoch}} \geq en \ln(nB)$ suffices as the element v^*_B is introduced with probability at least $1/(nB)$ in this case and the solution y_B is obtained with probability at least $1/n$ by considering at most B increases of C_B. □

The previous result can be adapted to the case of a uniform constraint $|x|_1 \leq r$ as considered in Sect. 3.2. As the cost of each item is 1, we can obtain the following corollary by adjusting the proofs of Theorem 2 and 3.

Corollary 1. *Let $t_{\max} = r \cdot t_{\text{epoch}}$ where $t_{\text{epoch}} \geq 2en \log n$. Then $(1+1)$-EA with archive computes a $(1 - 1/e)$-approximation with probability $1 - o(1)$.*

5 Experimental Investigations

We now carry out experimental investigations for the two single-objective evolutionary approaches introduced for optimizing constrained submodular problems. We evaluate the algorithms on instances of the NP-hard maximum coverage problem which is a classical submodular optimization problem defined on graphs.

We use standard-bit-mutation-plus (see [11]) as mutation operator for the algorithm and run them on the same instances as done in [11]. The underlying

Table 1. Maximum coverage scores obtained by (1+1) EA with archive and (1+λ)-EA on medium instances

Graph	B	$t_{\max}$	Uniform					Random				
			$(1+1)$ EA		$(1+\lambda)$-EA			$(1+1)$ EA		$(1+\lambda)$-EA		
			Mean	Std	Mean	Std	p-value	Mean	Std	Mean	Std	p-value
ca-CSphd	10	100000	222	0.000	222	0.000	1.000	234	7.764	235	9.296	0.695
	10	500000	222	0.000	222	0.000	1.000	242	11.846	240	11.939	0.399
	10	1000000	222	0.000	222	0.000	1.000	245	12.735	240	12.294	0.165
	43	100000	600	0.728	599	0.711	0.174	601	12.206	605	10.715	0.206
	43	500000	600	0.000	600	0.000	1.000	614	12.788	608	11.688	0.038
	43	1000000	600	0.000	600	0.000	1.000	619	12.417	609	13.414	0.008
	94	100000	927	0.964	928	0.750	0.191	932	11.212	931	12.182	0.589
	94	500000	928	0.365	928	0.000	0.824	947	11.671	940	11.732	0.035
	94	1000000	928	0.000	928	0.000	1.000	950	10.884	946	12.128	0.041
	188	100000	1278	1.413	1279	1.081	0.264	1304	12.423	1300	13.157	0.198
	188	500000	1279	0.702	1279	0.809	0.584	1321	12.863	1317	12.944	0.198
	188	1000000	1279	0.629	1279	0.629	1.000	1326	12.805	1322	13.281	0.212
ca-GrQc	12	100000	505	7.158	506	5.649	0.923	524	19.509	524	23.457	0.947
	12	500000	510	0.000	510	0.000	1.000	551	24.536	544	19.123	0.193
	12	1000000	510	0.000	510	0.000	1.000	563	19.987	555	22.822	0.132
	64	100000	1512	6.872	1512	6.518	0.690	1554	22.583	1547	22.618	0.261
	64	500000	1526	4.312	1526	4.476	0.668	1608	27.234	1596	25.095	0.147
	64	1000000	1529	2.380	1529	1.954	0.807	1631	20.223	1609	20.411	0.000
	207	100000	2742	6.646	2738	9.464	0.076	2766	22.039	2759	18.237	0.209
	207	500000	2769	4.720	2765	5.845	0.008	2844	21.315	2834	15.824	0.072
	207	1000000	2773	5.045	2768	4.480	0.000	2864	19.825	2845	16.747	0.001
	415	100000	3565	6.066	3557	9.051	0.000	3581	14.673	3563	14.940	0.000
	415	500000	3606	4.066	3602	4.468	0.001	3650	11.331	3631	12.922	0.000
	415	1000000	3612	4.838	3605	4.219	0.000	3664	12.773	3646	13.563	0.000
Erdos992	12	100000	601	2.773	600	2.682	0.359	645	15.347	649	16.234	0.333
	12	500000	604	0.000	604	0.000	1.000	668	19.263	669	23.397	0.947
	12	1000000	604	0.000	604	0.000	1.000	687	23.483	678	21.527	0.183
	78	100000	2454	6.173	2453	6.653	0.318	2398	31.071	2451	30.601	0.000
	78	500000	2472	0.890	2472	0.968	0.953	2489	37.452	2507	32.866	0.042
	78	1000000	2472	0.819	2473	0.629	0.988	2522	38.457	2521	37.696	0.988
	305	100000	4707	8.677	4718	12.322	0.000	4563	27.503	4613	34.571	0.000
	305	500000	4771	2.096	4772	1.489	0.061	4723	25.279	4750	21.008	0.000
	305	1000000	4774	0.937	4775	0.845	0.034	4752	25.387	4767	22.048	0.018
	610	100000	5240	5.026	5234	6.182	0.000	5236	13.125	5203	18.913	0.000
	610	500000	5263	0.937	5262	1.196	0.001	5314	8.737	5299	10.183	0.000
	610	1000000	5264	0.730	5263	1.155	0.745	5324	8.568	5312	9.105	0.000

Table 2. Maximum coverage scores obtained by (1+1) EA with archive and (1+λ)-EA on large instances

Graph	B	$t_{\max}$	Uniform					Random				
			$(1+1)$ EA		$(1+\lambda)$-EA		p-value	$(1+1)$ EA		$(1+\lambda)$-EA		p-value
			Mean	Std	Mean	Std		Mean	Std	Mean	Std	
ca-HepPh	13	100000	1807	26.271	1804	29.228	0.584	1845	51.305	1850	39.910	0.723
	13	500000	1842	15.291	1843	9.754	0.695	1913	47.774	1915	52.637	0.525
	13	1000000	1840	5.112	1842	8.632	0.657	1938	61.109	1919	50.702	0.126
	105	100000	4627	28.443	4611	33.286	0.050	4659	35.830	4671	43.630	0.196
	105	500000	4762	17.618	4757	19.511	0.315	4876	50.292	4860	41.453	0.176
	105	1000000	4787	11.162	4787	11.396	0.784	4938	39.003	4908	44.298	0.015
	560	100000	8514	25.487	8486	20.989	0.000	8550	44.136	8517	32.833	0.003
	560	500000	8766	10.316	8745	12.357	0.000	8887	29.469	8847	22.758	0.000
	560	1000000	8796	9.713	8783	11.683	0.000	8956	35.481	8906	29.424	0.000
	1120	100000	10200	20.861	10161	17.307	0.000	10205	24.322	10154	21.882	0.000
	1120	500000	10460	9.395	10438	10.915	0.000	10533	17.225	10488	19.822	0.000
	1120	1000000	10492	7.397	10470	8.358	0.000	10595	18.362	10550	15.359	0.000
ca-AstroPh	14	100000	2862	59.952	2865	48.219	0.988	2856	85.887	2892	92.832	0.130
	14	500000	2965	13.289	2972	9.649	0.019	3005	75.418	3025	70.585	0.274
	14	1000000	2978	4.249	2978	4.690	0.941	3038	80.257	3030	56.286	0.525
	133	100000	8337	60.951	8326	53.037	0.464	8391	58.454	8402	78.692	0.605
	133	500000	8679	28.093	8677	24.020	0.848	8805	54.412	8816	50.448	0.579
	133	1000000	8722	24.139	8725	16.979	0.779	8931	47.486	8912	53.551	0.072
	895	100000	15015	35.029	14948	37.983	0.000	15017	42.388	14973	40.257	0.000
	895	500000	15559	20.849	15538	16.778	0.000	15653	30.599	15609	35.076	0.000
	895	1000000	15638	10.944	15621	10.409	0.000	15779	33.990	15729	23.735	0.000
	1790	100000	17018	24.879	16936	23.255	0.000	17000	20.884	16921	26.926	0.000
	1790	500000	17438	8.670	17412	12.840	0.000	17488	15.484	17437	18.276	0.000
	1790	1000000	17491	8.581	17469	6.715	0.000	17579	11.757	17524	11.488	0.000
ca-CondMat	14	100000	1759	48.528	1753	68.135	0.807	1695	94.934	1758	95.933	0.010
	14	500000	1853	3.960	1853	4.925	0.569	1857	68.802	1885	62.486	0.048
	14	1000000	1856	3.108	1857	2.580	0.079	1875	65.595	1897	68.254	0.287
	146	100000	6633	56.195	6654	63.743	0.225	6546	81.784	6635	77.537	0.000
	146	500000	7030	21.828	7042	22.911	0.030	7050	54.530	7079	52.443	0.026
	146	1000000	7082	10.261	7079	14.222	0.610	7151	59.810	7176	70.114	0.179
	1068	100000	15744	58.965	15672	54.984	0.000	15721	61.110	15634	66.097	0.000
	1068	500000	16671	24.122	16655	28.693	0.013	16761	45.483	16728	44.291	0.006
	1068	1000000	16797	17.002	16789	17.744	0.149	16980	45.050	16924	45.781	0.000
	2136	100000	19150	32.710	18968	48.385	0.000	19130	36.406	18961	46.673	0.000
	2136	500000	20039	18.032	20003	21.860	0.000	20091	24.887	20025	25.165	0.000
	2136	1000000	20167	12.032	20148	10.722	0.000	20292	21.225	20219	22.946	0.000

graphs are sparse graphs from the network repository. In the uniform case, all nodes of weight 1 whereas in the random case each node have been assigned a cost independently of the others chosen uniformly at random in $[0.5, 1.5]$. For each uniform and random setting we consider the same bounds B and number of fitness evaluations $t_{\max}$ as done in [11], namely $B = \log_2 n, \sqrt{n}, \lfloor n/20 \rfloor, \lfloor n/10 \rfloor$ and $t_{\max} = 100000, 500000, 1000000$. A result is called statistically significant if the p-value is at most 0.05.

We first consider the graphs ca-CSphd, ca-GrQc, and Erdos992 consisting of 1882, 4158, and 6100 nodes, respectively. The results are shown in Table 1. For the graph ca-CSphd, the results obtained by the two algorithms are very similar in both the uniform and random setting. For the graph ca-GrQc, the results differ only for large values of B and the statistical test also obtains small p-values in this case. Similar observations can be made for the graph Erdos992. Small p-values can usually be observed together with a better performance of $(1 + 1)$-EA with archive. This is especially the case for the random setting.

We now consider results for the larger graphs ca-HepPh, ca-AstroPh, ca-CondMat, which consist of 11204, 17903, 21363 nodes, respectively. The results are shown in Table 2 and overall follow the same trend. Overall, the advantage of $(1 + 1)$-EA with archive over $(1 + \lambda)$-EA can be observed again for the larger values of B. The difference in mean also becomes larger for both the uniform and random instances and statistical tests show a stronger difference for the large instances then for the medium size ones.

Overall, it can be observed that the quality of solutions obtained by $(1 + \lambda)$-EA and (1+1)-EA with archive are very similar. Only for larger random instances with a large constraint bound, the $(1 + 1)$-EA with archive obtains significantly better results. Doing a cross comparison with the results presented in [11], we observe the results of $(1 + \lambda)$-EA and $(1 + 1)$-EA with archive are better then the ones of GSEMO and slightly inferior then the ones of SW-GSEMO. We can therefore say that our newly introduced simple single-objective algorithms provide a good alternative to the more complex multi-objective setups and even outperform standard approaches based on GSEMO for the considered instances.

6 Conclusions

The maximization of submodular functions under constraints captures a wide range of NP-hard combinatorial optimization problems. In addition to classical greedy algorithms, Pareto optimzation approaches relaxing a given constraint into an additional objective have shown to obtain state of the art results from a theoretical and practical perspective. Contrary to this, it has been shown that standard single-objective approaches using the classical (1+1) EA easily get stuck in local optima. We presented adaptive single-objective approaches increasing the set of feasible solution incrementally during the optimization process. For the $(1+\lambda)$-EA, we have shown that this leads to best possible theoretical performance guarantee for the case of a uniform constraint. For more general monotone cost constraints, we presented a $(1 + 1)$-EA with archive and have

shown that this algorithm obtains state of the art theoretical guarantees. Our experimental investigations show that both algorithms perform quite similar and outperform the standard GSEMO approach for the considered settings.

Acknowledgments. This work has been supported by the Australian Research Council (ARC) through grant FT200100536.

References

1. Crawford, V.G.: Faster guarantees of evolutionary algorithms for maximization of monotone submodular functions. In: IJCAI, pp. 1661–1667. ijcai.org (2021)
2. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. SIAM J. Comput. **40**(4), 1133–1153 (2011)
3. 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)
4. Friedrich, T., Neumann, F.: Maximizing submodular functions under matroid constraints by evolutionary algorithms. Evol. Comput. **23**(4), 543–558 (2015)
5. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Proceedings of CEC 2003, vol. 3, pp. 1918–1925 (2003). https://doi.org/10.1109/CEC.2003.1299908
6. Knowles, J.D., Watson, R.A., Corne, D.W.: Reducing local optima in single-objective problems by multi-objectivization. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 269–283. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44719-9_19
7. Krause, A., Golovin, D.: Submodular function maximization. In: Tractability, pp. 71–104. Cambridge University Press (2014)
8. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions - I. Math. Program. **14**(1), 265–294 (1978)
9. Neumann, A., Neumann, F.: Optimising monotone chance-constrained submodular functions using evolutionary multi-objective algorithms. In: Bäck, T., et al. (eds.) PPSN 2020. LNCS, vol. 12269, pp. 404–417. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58112-1_28
10. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. Nat. Comput. **5**(3), 305–319 (2006)
11. Neumann, F., Witt, C.: Fast Pareto optimization using sliding window selection. In: ECAI. Frontiers in Artificial Intelligence and Applications, vol. 372, pp. 1771–1778. IOS Press (2023)
12. Qian, C., Shi, J., Yu, Y., Tang, K.: On subset selection with general cost constraints. In: IJCAI, pp. 2613–2619 (2017). https://doi.org/10.24963/ijcai.2017/364
13. Qian, C., Yu, Y., Tang, K., Yao, X., Zhou, Z.: Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. Artif. Intell. **275**, 279–294 (2019)
14. Qian, C., Yu, Y., Zhou, Z.: Subset selection by Pareto optimization. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, pp. 1774–1782 (2015)
15. Roostapour, V., Neumann, A., Neumann, F., Friedrich, T.: Pareto optimization for subset selection with dynamic cost constraints. Artif. Intell. **302**, 103597 (2022)

Local Optima in Diversity Optimization: Non-trivial Offspring Population is Essential

Denis Antipov(✉), Aneta Neumann, and Frank Neumann

Optimisation and Logistics, School of Computer and Mathematical Sciences, University of Adelaide, Adelaide, Australia
{denis.antipov,aneta.neumann,frank.neumann}@adelaide.edu.au

Abstract. The main goal of diversity optimization is to find a diverse set of solutions which satisfy some lower bound on their fitness. Evolutionary algorithms (EAs) are often used for such tasks, since they are naturally designed to optimize populations of solutions. This approach to diversity optimization, called EDO, has been previously studied from theoretical perspective, but most studies considered only EAs with a trivial offspring population such as the $(\mu+1)$ EA. In this paper we give an example instance of a k-vertex cover problem, which highlights a critical difference of the diversity optimization from the regular single-objective optimization, namely that there might be a locally optimal population from which we can escape only by replacing at least two individuals at once, which the $(\mu+1)$ algorithms cannot do.

We also show that the $(\mu+\lambda)$ EA with $\lambda \geq \mu$ can effectively find a diverse population on k-vertex cover, if using a mutation operator inspired by Branson and Sutton (TCS 2023). To avoid the problem of subset selection which arises in the $(\mu+\lambda)$ EA when it optimizes diversity, we also propose the $(1_\mu + 1_\mu)$ EA_D, which is an analogue of the $(1+1)$ EA for populations, and which is also efficient at optimizing diversity on the k-vertex cover problem.

Keywords: Diversity Optimization · Population-based Algorithms · Theory · Landscape Analysis · Vertex Cover

1 Introduction

Obtaining a diverse set of good solutions is a complex optimization task, which often arises in real-world problems such as planning [21], satisfiability [33], architectural planning [16], cutting materials [19] and others. The most common reason for the need of a diverse set of solutions is that some objectives or constraints cannot be strictly formalized (e.g., for political, ethical, aesthetic or other reasons), therefore an algorithm user would like to get not a single best solution, but a set of good solutions to choose from. And if this set is not diverse enough, all solutions might occur infeasible in terms of those non-formalizable constraints.

Formalizing diversity is also a non-trivial task, and often it is problem-specific. One of the ways to get a set which can be called diverse is to divide

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 181–196, 2024.
https://doi.org/10.1007/978-3-031-70071-2_12

the search space into regions and to optimize the objective (or objectives) in each region simultaneously [9,27]. This approach, called the quality diversity (or QD for brevity), has been mainly developed in the domains of robotics and games [8,18,25,26,37,40]. Recently this approach has been applied to the traveling thief problem, and various domains such as design and health [24,32,34].

Another way to formalize the problem, which we adopt in this paper, is to define a diversity measure over the space of sets of solutions, and turn the problem into optimizing this measure under some constraints on the quality of solutions in the population.

The popularity of this problem has attracted a lot of attention from the algorithmic community, which resulted in developing multiple approaches from deterministic ones [3,22] to using various randomized search heuristics [4,20]. One of the most efficient approaches is using evolutionary algorithms (EAs), which is called *evolutionary diversity optimization* or EDO for brevity. It has been used in many applications, including classical combinatorial problems such as the traveling salesperson problem [10,30], the traveling thief problem [31], the computation of minimum spanning trees [5], submodular problems [28] and related communication problems in the area of defense [17,29].

The reason behind EDO's spread is that in diversity optimization the aim is to find not a single solution, but a set of solutions, which most optimization algorithms struggle with due to the increased dimensionality of the problem. The EAs, however, have been naturally designed to evolve a population of solutions, hence they do not need much adaptation for the diversity-oriented problems.

The theoretical foundations of EDO have been established by studying this approach on benchmark functions such as OneMax or LeadingOnes [15], as well as on permutation problems without any constraint on the fitness of the solutions [10]. It has also been studied on submodular functions in [11,28]. In all cases upper bounds on the runtime were shown, indicating that the runtime is always finite. It does not appear to be a surprising statement, since most EAs are designed in such a way that they have a finite runtime, e.g., by using standard bit mutation which guarantees a non-zero probability for generating the global optimum in any generation. For this reason in single-objective optimization EAs with a non-trivial parent population such as the $(\mu + 1)$ EA can always find an optimum (probably in ridiculously large n^n time as on the Trap function).

In this paper we demonstrate that in EDO the standard bit mutation does not guarantee a finite runtime. We give an example of a k-vertex cover instance, where none of the algorithms using the $(\mu + 1)$ scheme (that is, the elitist EAs which have a non-trivial parent population and which generate only one offspring per generation) can find the global optimum in finite time. This indicates that in optimization of some diversity measure over a set of populations replacing only one solution is similar to performing a one-bit mutation in single-objective problems, which might get the process stuck in a local optimum on any non-monotone function.

On the positive side, we show that the $(\mu + \lambda)$ EA_D with $\lambda \geq \mu$ can always find an optimal population by generating all individuals from a population with optimal diversity in one iteration. Since selecting a subset with optimal diversity might be a non-trivial task for the $(\mu + \lambda)$ EA_D, we also propose the $(1_\mu + 1_\mu)$ EA_D, which is an analogue of the $(1 + 1)$ EA for population space. We use a jump-and-repair mutation operator from [6] and show that the $(\mu + \lambda)$ EA_D with $\lambda \geq \mu$ and the $(1_\mu + 1_\mu)$ EA_D using this operator can find an optimally diverse set of k-covers on a graph with n vertices in expected number of at most $2^{k+\mu}(1 - o(1))$ iterations.[1] We note that since the main aim of the diversity optimization is to give a decision-maker (usually, a human) a diverse set of solutions, then μ is preferred to be not too large, e.g., constant. Therefore, if we also take $k = O(\log(n))$, then the runtime will be polynomial.

The rest of the paper is organized as follows. In Sect. 2 we formally define the k-vertex cover problem, as well as the diversity measures and the algorithms we study in this paper. Section 3 gives an example of a graph and a population of k-vertex covers on this graph which is locally optimal in the space of populations. We then perform a runtime analysis of the $(1_\mu + 1_\mu)$ EA_D showing its efficiency on this problem in Sect. 4. A discussion of our results concludes the paper in Sect. 5.

2 Preliminaries

2.1 Diversity Optimization

The problem of diversity optimization was defined in [38]. We slightly generalize it as follows. Given a fitness function $f : \Omega \mapsto \mathbb{R}$ (where Ω is the search space), population size μ, quality threshold B and diversity measure D, the goal is to find a population P, which is a multiset of μ elements from Ω, with the best value of $D(P)$ under condition that all $x \in P$ meet the quality threshold, that is, $f(x) \geq B$.

Usually the definition of the diversity measure is problem-specific, but it always reflects how different the solutions are in the search space (but not in the fitness space). In this paper we study the *total Hamming distance*, a diversity measure which has been previously studied in the context of theoretical runtime analysis of EDO [2,12,15]. Given a population P of bit strings, the total Hamming distance is defined as $D(P) = \sum_{x,y \in P} H(x, y)$, where H is the Hamming distance and the sum is over all unique unordered pairs of individuals.

An important property of this measure for pseudo-Boolean optimization (that is, when the search space is the set $\{0, 1\}^n$ of all bit strings of length n) which has been previously used in the analysis of EDO algorithms is that it can be computed through the number of one-bits in each position in the population. It was first shown in [39], and later was used in many works which studied this

[1] As in the vast majority of theoretical studies, we focus on estimating the number of iterations rather than the wall-clock time.

diversity measure. Let n_i be a number of ones in position i over all individuals in population P. Then

$$D(P) = \sum_{i=1}^{n} n_i(|P| - n_i). \tag{1}$$

This property implies that the contribution of position i into diversity is maximized, when $n_i = \lceil \frac{|P|}{2} \rceil$ or $n_i = \lfloor \frac{|P|}{2} \rfloor$. The maximum diversity is obtained when all n_i are either $\lceil \frac{|P|}{2} \rceil$ or $\lfloor \frac{|P|}{2} \rfloor$. However it is not always possible due to the constraints on the fitness of the solutions.

2.2 Vertex Cover

For a given undirected graph $G = (V, E)$ we call any set of vertices such that all edges in E are adjacent to at least one vertex in this set a *vertex cover* (or *cover* for short). The *minimum vertex cover* is a problem of finding a cover of minimum size, and the *k-vertex cover* is a fixed-target[2] variant of this problem, that is, the problem of finding a cover of a size of at most k.

Finding a k-vertex cover for an arbitrary graph and an arbitrary k is a classic NP-hard problem, and therefore, there is no known algorithm which could solve this problem efficiently (that is, in a polynomial time). For this reason the EAs have been previously applied to it in many different ways. In [35,36] it has been shown that the classic EAs can be very ineffective on some instances of the vertex cover, when they use a single-objective formulation. Hybrid evolutionary approaches have been studied in [13], and in [14,23] an effectiveness of the multi-objective approach has been shown. A typical representation of the vertex cover when applying EAs is a bit string of length $n = |V|$, where the i-th bit indicates if the i-th vertex is included into the set. We use this representation in this paper.

It is also well-known that the k-vertex cover is a fixed parameter tractable (FPT) problem [23], that is, there exists a parameter of the instance k such that the time we need to optimize the instance is $f(k) \cdot \mathrm{Poly}(|V|)$. In our case, this parameter k is the size of the optimal cover. To address the FPT property of this problem, in [6] Branson and Sutton used a modified representation for individuals and proposed a jump-and-repair mutation operator which allowed the $(1 + 1)$ EA to find a k-vertex cover in expected number of $O(2^k n^2 \log(n))$ iterations, if such cover exists. The main idea behind that operator is that if there exist a vertex cover y of size at most k such that none of the vertices can be removed from it, then we can get it from any other vertex cover x by removing all vertices belonging to $x \setminus y$ and adding their neighbours (see Lemma 4 in [6]).

In this work we aim at finding a diverse (in terms of the total Hamming distance) set of vertex covers of size at most k for a given graph $G = (V, E)$, assuming that at least one such cover exists. We also assume that the target population size μ and the cover size k are relatively small, namely that $k\mu =$

[2] For more information about fixed-target analysis and notation see [7].

Algorithm 1: The jump-and-repair mutation operator for diversity optimization on k-vertex cover problem on graph $G = (V, E)$ based on Algorithm 4 in [6].

```
1  Input: graph G = (V, E) ;
2  Input: integer k > 0;
3  Input: parent cover x ⊆ V such that |x| ≤ k;
4  S ← ∅;                                  // A set of vertices to remove from x
5  for v ∈ x do
6      With probability 1/2 add v to S;
7  y ← x \ S;
8  for v ∈ S do                  // Adding neighbours of the removed vertices
9      for u ∈ V : (u, v) ∈ E do
10         y ← y ∪ {u};
11 while |y| < k do          // Adding more random vertices to get a k-cover
12     Choose z from V \ y u.a.r.;
13     y ← y ∪ {z};
14 return y;
```

$o(\sqrt{n})$. This implies that by the pigeonhole principle, in any population there will be only $o(\sqrt{n})$ different positions which have at least one-bit, and therefore there will be many positions i, in which all individuals would have a zero-bit (that is, vertex i is not included in any cover in the population). If a population has some individuals which have $\ell < k$ vertices, than adding $k - \ell$ vertices not included into any individual in the population will increase the corresponding terms in Eq. (1) by $|P| = \mu$, hence it never makes sense to have covers of size less than k in the population.

However, the mutation operator used in [6] cannot generate all covers of size k, but only *non-excessive* ones, that are, those covers from which we cannot remove any vertex and keep it a cover. For this reason, in this work we modify their mutation operator. Our modified operator is shown in Algorithm 1. Given a cover x, this jump-and-repair mutation first removes each vertex from it with probability $1/2$ and then adds all neighbours of the removed vertices, similar to the operator from [6]. Since we add all neighbours of the removed vertices, it guarantees that the result of this mutation is a cover, that is, there is no edge for which none of the two adjacent vertices is in the resulting individual. This cover might be of size less than k, and in this case we add some randomly chosen vertices to make the size of the vertex cover exactly k.

The following lemma is an extension of Lemma 4 in [6].

Lemma 1. *Let x be a k-vertex cover of graph G and let y be a non-excessive cover of size at most k. Then the probability that the jump-and-repair mutation (Algorithm 1) applied to x generates y before adding additional random vertices (that is, by line 11 in Algorithm 1) is exactly 2^{-k}.*

Algorithm 2: The $(\mu + 1)$ EA_D maximizing function f and maximizing diversity measure D with fitness threshold B.

Input: population of μ individuals P;
Define $g : x \mapsto \min\{f(x), B\}$;
while *stopping criterion is not met* **do**
    Create a new individual y ; `// We intentionally do not specify how it is created`
    if $g(y) \geq \min_{x \in P} g(x)$ **then**
        $P \leftarrow P \cup \{y\}$;
        $Q \leftarrow \text{argmin}_{x \in P} g(x)$; `// argmin returns a set of individuals (probably, a trivial set of one element)`
        $S \leftarrow \text{argmin}_{x \in Q} D(P \setminus \{x\})$;
        if $|S| \geq 2$ *and* $y \in S$ **then**
            $S \leftarrow S \setminus \{y\}$;
        $x \leftarrow$ random individual from S;
        $P \leftarrow P \setminus \{x\}$;
return P;

Proof. Let S be $x \setminus y$, that is, the vertices in x which are not in y. All neighbours of vertices in S are in y, since otherwise an edge between such a vertex and it neighbour is not covered by y. Therefore, removing S from x and adding their neighbours results in a cover which is a subset of y. Since y is non-excessive, it is exactly y.

The probability that we remove S and keep $x \setminus S$ is exactly 2^{-k}, since each vertex is removed with probability $\frac{1}{2}$. □

2.3 The Considered EAs

In this paper we consider population-based EAs which are commonly used in the diversity optimization. Most of theoretical studies of EDO considered the $(\mu + 1)$ EA_D, which optimizes the diversity only when it breaks ties between candidates for the next generation [5,10,12]. We describe a generalized version of this EA in Algorithm 2. This algorithm stores a population P of μ individuals. We do not specify the way these individuals are initialized. In each iteration this algorithm creates a new individual y by applying variation operators (usually, mutation and crossover) to some randomly chosen individuals from the population. If the fitness of y is not worse than the worst fitness in the population or satisfies the quality threshold, y is added into P, and then we remove an individual with the worst fitness (counting all individuals which meet the quality threshold as equal). If there are several such individuals, we remove the one with the smallest contribution to the diversity. If there are still more than one such individuals, then we choose one of them uniformly at random (but in this case we do not choose y if it is one of these individuals) and remove it. We note that

Algorithm 3: The $(1_\mu + 1_\mu)$ EA_D maximizing diversity measure D under constraint $f(x) \geq B$.

```
Input: population of μ individuals P meeting the minimum fitness threshold;
while stopping criterion is not met do
    P' ← ∅;
    for i ∈ [1..μ] do
        Create a new individual y;
        P' ← P' ∪ {y};
    if ∀x ∈ P : f(x) ≥ B and D(P') ≥ D(P) then
        P ← P'.
return P;
```

since we are searching for a population of individuals which meet the minimum fitness threshold B, all individuals above this threshold are similarly feasible for us. Hence, instead of optimizing the original objective f, the $(\mu + 1)$ EA_D optimizes $g : x \mapsto \min\{f(x), B\}$.

We also consider an elitist EA with a non-trivial offspring population, the $(\mu+\lambda)$ EA_D. The main difference of this algorithm from the $(\mu+1)$ EA_D is that it creates λ offspring in each iteration, each by independently choosing parents (or a parent) from the current population and applying variation operators to them. The main complication of the $(\mu+\lambda)$ EA_D compared to the $(\mu+1)$ EA_D is in the selection of the individuals into the next population. After we add all λ offspring to the current population P, we need to remove λ individuals from P. We first remove individuals with the minimum fitness (according to the modified fitness g) as long as the removal of them does not make size of P less than μ. If after that the size of P is more than μ, we need to break a tie and select $|P| - \mu$ individuals with the worst fitness to remove. We always select a set of $|P| - \mu$ such individuals which has the smallest contribution to the diversity, similar to how we do it in the $(\mu + 1)$ EA_D.

The subset selection problem might be very demanding for the computational resources, especially when we have to break a tie between 2μ individuals, hence in the diversity optimization it makes sense to use the following analogue of the $(\mu+\lambda)$ EA_D, which we call the $(1_\mu+1_\mu)$ EA_D. This algorithm is initialized with a population of μ individuals, all of which meet our constraints on the worst fitness value (if we get at least one such individual, we can use a population of its copies). In each iteration it creates a population P' of μ offspring individuals (in the same way as the $(\mu + \lambda)$ EA_D creates λ offspring) and replaces P with P', if all individuals of P' are feasible and the diversity of P' is not worse than the diversity of P according to the chosen diversity measure. This algorithm can be considered as a variant of the $(1 + 1)$ EA in the population space, for which individuals are bit strings of size μn. Its pseudocode is shown in Algorithm 3.

3 Locally Optimal Population

In this section we give examples of vertex cover instances, for which there exists a population with sub-optimal diversity, such that to improve its diversity we need to change at least two individuals together. This implies that if a $(\mu + 1)$-kind of algorithm gets such a population, it gets stuck in it, since it only changes one individual in each iteration. Note that in this section the diversity is always measured via the total Hamming distance.

We start with a simple example, where the population size is 2 and the graph has 8 vertices. We then extend this simple example to an arbitrary even population size μ and any even problem size $n \geq 10$.

3.1 The Simple Example

Consider graph $G = (V, E)$ with 8 vertices $\{v_1, \ldots, v_8\}$ and edges as shown in Fig. 1.

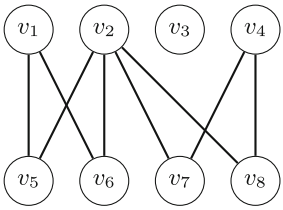


Fig. 1. Graph G, for which we have a population of 4-covers with suboptimal diversity, from which it is impossible to escape by replacing only one individual.

This is a bipartite graph which has 8 vertices and 8 edges. We will show that when we aim at finding the most diverse population of size $\mu = 2$ for a 4-vertex cover on this graph, we might occur in a local optimum which is impossible to escape for the $(\mu + 1)$ kind of algorithms. The next lemma shows the main properties of G.

Lemma 2. *The following statements are true for graph G shown in Fig. 1*

1. *There is no vertex cover of size one or two.*
2. *The only 3-vertex cover is $\{v_1, v_2, v_4\}$.*
3. *The only 4-vertex cover which includes v_3 is $\{v_1, v_2, v_3, v_4\}$.*
4. *The only 4-vertex cover which does not include v_2 is $\{v_5, v_6, v_7, v_8\}$.*

Proof. The proof of all statements is based on the following observation. If we have a set of vertices V' with known degrees (number of adjacent edges), then the number of edges they cover is $\sum_{v \in V'} deg(v) - K$, where K is the number of edges which are covered from both sides.

The First Statement. Graph G has only one vertex with degree 4 (namely, v_2), and all others have degree 2, except v_3 which has degree zero. Hence, one or two vertices can cover at most 4 and 6 edges respectively, which is less than the total number of edges. Hence, no 1-cover or 2-cover exists.

The Second Statement. A 3-vertex cover must include v_2, otherwise the sum of degrees of the vertices will be at most 6, which is less than the number of edges. The other two vertices cannot cover any of the edges covered by v_2, since otherwise the number of covered edges will be strictly less than the sum of their degrees, that is, 8. Hence, we cannot have v_5, v_6, v_7 or v_8 in this cover. Among the rest, the only two vertices which have degree at least 2 are v_1 and v_4, which we must include into the cover. Since $\{v_1, v_2, v_4\}$ covers all the edges, it is a unique 3-vertex cover.

The Third Statement. If we include v_3 into the cover, it does not cover any edges. Hence, the rest of the three vertices must cover all 8 edges. By the previous statement, the only way to do so is to take $\{v_1, v_2, v_4\}$. Hence, $\{v_1, v_2, v_3, v_4\}$ is the only 4-vertex cover that includes v_3.

The Fourth Statement. If we do not include v_2 in the cover, then we need to include all its neighbors to cover the edges adjacent to v_2. Its neighbors are $\{v_5, v_6, v_7, v_8\}$, which form a vertex cover. □

Based on these properties we can build a population with locally optimal diversity. We do it in the following lemma. We note that for the population of size two, the diversity is defined by the Hamming distance between the two individuals.

Lemma 3. *Consider a problem of finding the most diverse population of size two of 4-vertex covers of graph G shown in Fig. 1. The following sets are 4-vertex covers for this graph.*

- $V_1 = \{v_1, v_2, v_7, v_8\}$,
- $V_2 = \{v_2, v_4, v_5, v_6\}$,
- $V_3 = \{v_1, v_2, v_3, v_4\}$,
- $V_4 = \{v_5, v_6, v_7, v_8\}$.

Covers V_1, V_2 and V_4 are non-excessive. Then the unique population of size two with the maximum Hamming distance 8 is (V_3, V_4). Population (V_1, V_2) has a Hamming distance 6 and replacing any individual with a different 3- or 4-vertex cover would reduce the Hamming distance, that is, this population is a local optimum.

Proof. All the sets V_1, V_2, V_3 and V_4 are 4-vertex covers, and the distances between them are the following.

- $H(V_1, V_2) = 6$,
- $H(V_3, V_4) = 8$ and
- all other distances are 4 (since each pair coincides in exactly two included vertices and exactly two non-included).

By Lemma 2, V_3 is the unique 4-cover which has v_3, and V_4 is the unique 4-cover which does not include v_2. Therefore, for any 3- or 4-cover V, covers V_1 and V_2 coincide with it in including v_2 and not including v_3, hence $H(V_1, V) \leq 6$ and $H(V_2, V) \leq 6$. To have $H(V_1, V) = 6$, we need V to be different from V_1 in all vertices, except v_2 and v_3, and the only cover which satisfies it is V_2. It is also true the other way around: the only cover in distance 6 from V_2 is V_1. Hence, if we have population (V_1, V_2), then replacing any of the individuals with another 3- or 4-cover makes the distance between the two individuals smaller, and therefore cannot be accepted by any $(\mu + 1)$ elitist algorithm. □

3.2 Extending the Example to Arbitrary Population and Problem Sizes

Based on the example given in the previous subsection, we now show that a population with sub-optimal diversity which cannot be escaped by the $(\mu+1)$ EA exists also for larger μ and $|V|$. We start with extending our example for larger population sizes, keeping $|V| = 8$.

We consider the same graph G as shown in Fig. 1, and give an example of a locally optimal population in the following lemma.

Lemma 4. *Let $\mu \geq 4$ be even and let $\nu = \frac{\mu}{2} - 1$. Let also V_1, V_2, V_3 and V_4 be the same vertex covers as in Lemma 3. Consider a population which has one individual V_1, one individual V_2, ν individuals V_3, and ν individuals V_4. Then this population has a sub-optimal diversity (the total Hamming distance) and replacing any individual with any different vertex cover of size at most 4 reduces the diversity.*

Proof. We exploit the expression of the total Hamming distance given in Eq. (1). In the given population all n_i (the number of one-bits in position i) are $\frac{\mu}{2}$, except for $i = 2$ and $i = 3$. Since the bit strings representing V_1 and V_2 have a one-bit in position 2 and a zero-bit in position 3, we have $n_2 = \frac{\mu}{2} + 1$ and $n_3 = \frac{\mu}{2} - 1$. Hence, if we change the number of one-bits in any position (except 2 and 3) by one, it decreases the corresponding term in Eq. (1) by one, since we have

$$\left(\frac{\mu}{2}\right)^2 - \left(\frac{\mu}{2} + 1\right)\left(\frac{\mu}{2} - 1\right) = \left(\frac{\mu}{2}\right)^2 - \left(\left(\frac{\mu}{2}\right)^2 - 1\right) = 1.$$

For the same reason, increasing the number of one-bits in position 2 decreases this term by 3 and decreasing the number of one-bits by one increases it by 1. For position 3 it is the other way around.

We now consider different cases of replacing individuals in the population with different 4- or 3-covers, and show that all of them would only decrease the total Hamming distance.

Case 1: Replacing V_1. If we replace the only individual V_1 with V_3, then we increase the number of one-bits in position 3 (and therefore, its contribution to the diversity is increased by one), and we change the number of one-bits in

positions 4, 7 and 8, which decreases the diversity by 3. Therefore, the total diversity is decreased. Similarly, if we replace V_1 with V_4, we make the diversity in position 2 better, but we unbalance positions 1, 5 and 6, hence, we decrease the diversity.

If we replace V_1 with any other 3 or 4-cover, then this cover has the same values in positions 2 and 3 by Lemma 2, and at least one other value should be different. This would decrease the term in Eq. (1) which corresponds to this different position. Hence, the diversity is decreased.

Case 2: Replacing V_2. This case is similar to Case 1. Replacing V_2 with either V_3 or V_4 decreases the diversity by two, and any other replacement decreases it by at least one.

Case 3: Replacing V_3. In this case we consider replacing one of the ν individuals V_3 with a different one. If we replace it with V_4, then we decrease n_2, which increases the diversity by one. However, it also decreases n_3, which decreases the diversity by 3 and changes all other n_i, which decreases the diversity by 6 more. Hence, the diversity is decreased by 8. Since by Lemma 2 all other covers include v_2 and do not include v_3, replacing V_3 with one of such covers does not change n_2 and decreases n_3 by one, which reduces the diversity by 3. The changes in other positions can only reduce diversity even more. Therefore, any replacement of any individual representing V_3 would decrease the diversity.

Case 4: Replacing V_4. This case is similar to Case 3. If we replace it with V_3, we balance position 3, but we unbalance all other positions, which decreases the diversity. Replacing it with any other cover would unbalance at least one position.

Bringing all cases together, we conclude that replacing any individual in this population with a different 3- or 4-vertex cover decreases the total Hamming distance, and therefore is not accepted by the algorithm. □

To extend this result to larger problem sizes it is enough to add to the graph in Fig. 1 a complete bipartite graph $K_{n,n}$, which is not connected with the basic graph and consider the $(n+4)$-cover problem. Then the locally optimal population will be the same as in Lemma 4, but half of the individuals in that population must contain one half of the bipartite graph, and another half of individuals—another half of the bipartite graph. This results in a population in which all positions except 2 and 3 are balanced. Hence, the arguments of Lemma 4 will work on this graph as well, if we note that changing the value of any bit corresponding to an added vertex would result in decreasing the corresponding term in (1) and therefore, reducing the diversity.

4 Large Offspring Populations Are Effective

In this section we show that the $(1_\mu + 1_\mu)$ EA_D and also the $(\mu + \lambda)$ EA_D with $\lambda \geq \mu$ can effectively find a diverse population of k-vertex covers. The main result is the following theorem.

Theorem 1. *Consider the* $(1_\mu + 1_\mu)$ EA_D *or the* $(\mu + \lambda)$ EA_D *with* $\lambda \geq \mu$*, which optimize the total Hamming distance on a* k*-vertex cover instance, for which at least one* k*-cover exist. If we have* $k\mu = o(\sqrt{n})$*, then in each iteration the probability of these two algorithms to find a population of* k*-vertex covers with optimal diversity is at least* $2^{-k\mu}(1 - o(1))$*, and therefore, its runtime is dominated by geometric distribution* $\mathrm{Geom}(2^{-k\mu}(1 - o(1)))$.

Proof. Let P_{opt} be a population of μ individuals which meet the constraint on fitness and which has the best possible diversity. As it was discussed in Sect. 2.2, all individuals in this population have exactly k vertices, since otherwise we could add additional vertices to some of them and increase diversity. Let P'_{opt} be another population of μ covers, where the i-th individual is a non-excessive cover, which is a subset of the i-th individual in P'_{opt}.

To get P_{opt} from P'_{opt} we need to add vertices to individuals which have less than k vertices. If we add a vertex to position i, in which $n_i > 0$ individuals have a one-bit (that is, they include vertex i), then the improvement of the corresponding term in Eq. (1) will be

$$(n_i + 1)(\mu - n_i - 1) - n_i(\mu - n_i) = \mu - n_i - 1 - n_i + 1 < \mu.$$

Hence, we can add this vertex to another position i, in which $n_i = 0$ and improve the diversity more, namely, by μ. Therefore, if at least one vertex is added to a position with $n_i > 0$, then it contradicts with optimality of $D(P_{\mathrm{opt}})$. We also note that any way of adding the missing vertices to P'_{opt} to positions with $n_i = 0$, would yield the same increase in diversity, therefore, to get an optimal diversity, we do not have to get P_{opt}: any other population obtained in this way from P'_{opt} has an optimal diversity.

The probability that we create such a population in one iteration of the $(1_\mu + 1_\mu)$ EA_D or the $(\mu+\lambda)$ EA_D with $\lambda \geq \mu$ is at least the probability that for each i we generate the i-th offspring y_i by first creating the i-th individual x'_i of P'_{opt} and then we add the missing vertices (if there are any) to positions with no vertices in other individuals. For the $(1_\mu + 1_\mu)$ EA_D it gives a population with optimal diversity, and for the $(\mu + \lambda)$ EA_D the population with optimal diversity is a subset of the new offspring, hence the diversity of the next-generation population cannot be sub-optimal.

By Lemma 1, the probability to create an individual from P'_{opt} is 2^{-k}, hence the probability that we create exactly μ such individuals is at least $2^{-k\mu}$. We then add the missing vertices. At each point of time there are at least $n - k\mu$ *good* positions in which there is no vertex in any individual, and there are at most n positions to which we add a vertex, hence the probability that we add it to a good position is at least $(1 - \frac{k\mu}{n})$. Since we need to add at most $k\mu$ vertices, the probability that all of them are added to good positions is at least

$$\left(1 - \frac{k\mu}{n}\right)^{k\mu} \geq 1 - \frac{(k\mu)^2}{n} = 1 - o(1),$$

where we used Bernoulli inequality and the lemma condition $k\mu = o(\sqrt{n})$. Hence, we conclude that the probability to create a population with optimal diversity in each iteration is at least $2^{-k\mu}(1 - o(1))$. □

If we assume that μ is some constant, the expected runtime given by Theorem 1 is at most $O(2^k)$ iterations or fitness evaluations, hence the task of finding a diverse population with the $(1_\mu + 1_\mu)$ EA_D or the $(\mu + \lambda)$ EA_D is easier than finding a k-vertex cover with methods, proposed in [6], where an upper bound of $O(2^k n^2 \log(n))$ iterations was shown for the $(1 + 1)$ EA.

5 Conclusion

In this paper we showed the first example of a local optimum in the diversity optimization problem, from which it is impossible to escape by replacing only one individual in the population if we optimize diversity in an elitist way. This result illustrates that when optimizing in the space of populations, the $(\mu + 1)$ algorithms can be interpreted as local search in that space. To get a positive probability of finding an optimally diverse population in any iteration we have to be able to perform global changes on the population, which demands from us creating at least μ offspring. This idea brought us to the $(1_\mu + 1_\mu)$ EA_D, which is an analogue of the $(1 + 1)$ EA, where in role of individuals we have populations represented with bit strings of size $n\mu$.

The first signs of this result might have been seen in the previous empirical study [29]. There it was shown that the amount of diversity of the obtained solutions in the context of constructing a wireless communication network can be increased in most of the cases by even slightly increasing the offspring population size. This also rises a question on how many individuals should we replace at once to escape such local optima. Creating the number of offspring which is at least the size of the parent population is definitely enough, and, as we have shown in this paper, might be effective when the parent population size is not too large. However, if we want to obtain large diverse populations, then counting on generating the whole optimal population in one iteration is not promising, and our hope would be on improving the diversity via replacing a small number of individuals per iteration.

Before we find an answer to the question on what the offspring population size should be, a lazy approach to such problems would be using variable population size in the $(\mu+\lambda)$ EA_D. A good strategy for this might be choosing λ according to the power-law distribution, which on the one hand gives us a decent probability to have a large population size, but also preserves a small expected cost of one iteration, as it was shown in [1].

Acknowledgements. This work has been supported by the Australian Research Council through grants DP190103894 and FT200100536.

References

1. Antipov, D., Buzdalov, M., Doerr, B.: Fast mutation in crossover-based algorithms. Algorithmica **84**(6), 1724–1761 (2022)
2. Antipov, D., Neumann, A., Neumann, F.: Rigorous runtime analysis of diversity optimization with GSEMO on OneMinMax. In: Foundations of Genetic Algorithms, FOGA 2023, pp. 3–14. ACM (2023)
3. Baste, J., et al.: Diversity of solutions: an exploration through the lens of fixed-parameter tractability theory. Artif. Intell. **303**, 103644 (2022)
4. Benke, L., Miller, T., Papasimeon, M., Lipovetzky, N.: Diverse, top-k, and top-quality planning over simulators. In: ECAI. Frontiers in Artificial Intelligence and Applications, vol. 372, pp. 231–238. IOS Press (2023)
5. Bossek, J., Neumann, F.: Evolutionary diversity optimization and the minimum spanning tree problem. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 198–206. ACM (2021)
6. Branson, L., Sutton, A.M.: Focused jump-and-repair constraint handling for fixed-parameter tractable graph problems closed under induced subgraphs. Theor. Comput. Sci. **951**, 113719 (2023)
7. Buzdalov, M., Doerr, B., Doerr, C., Vinokurov, D.: Fixed-target runtime analysis. Algorithmica **84**(6), 1762–1793 (2022)
8. Chatzilygeroudis, K., Cully, A., Vassiliades, V., Mouret, J.-B.: Quality-diversity optimization: a novel branch of stochastic optimization. In: Pardalos, P.M., Rasskazova, V., Vrahatis, M.N. (eds.) Black Box Optimization, Machine Learning, and No-Free Lunch Theorems. SOIA, vol. 170, pp. 109–135. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-66515-9_4
9. Cully, A., Mouret, J.: Behavioral repertoire learning in robotics. In: Genetic and Evolutionary Computation Conference, 2013, pp. 175–182. ACM (2013)
10. Do, A.V., Guo, M., Neumann, A., Neumann, F.: Analysis of evolutionary diversity optimization for permutation problems. ACM Trans. Evol. Learn. Optim. **2**(3), 11:1–11:27 (2022)
11. Do, A.V., Guo, M., Neumann, A., Neumann, F.: Diverse approximations for monotone submodular maximization problems with a matroid constraint. In: International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5558–5566. ijcai.org (2023)
12. Doerr, B., Gao, W., Neumann, F.: Runtime analysis of evolutionary diversity maximization for OneMinMax. In: Genetic and Evolutionary Computation Conference, GECCO 2016, pp. 557–564. ACM (2016)
13. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Analyses of simple hybrid algorithms for the vertex cover problem. Evol. Comput. **17**(1), 3–19 (2009)
14. 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)
15. Gao, W., Neumann, F.: Runtime analysis for maximizing population diversity in single-objective optimization. In: Genetic and Evolutionary Computation Conference, GECCO 2014, pp. 777–784. ACM (2014)
16. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. Evol. Comput. **18**(3), 335–356 (2010)
17. Gounder, S., Neumann, F., Neumann, A.: Evolutionary diversity optimisation for sparse directed communication networks. In: Genetic and Evolutionary Computation Conference, GECCO 2024. ACM (2024, to appear)

18. Gravina, D., Khalifa, A., Liapis, A., Togelius, J., Yannakakis, G.N.: Procedural content generation throuertegh quality diversity. In: IEEE Conference on Games, CoG 2019, pp. 1–8. IEEE (2019)
19. Haessler, R.W., Sweeney, P.E.: Cutting stock problems and solution procedures. Eur. J. Oper. Res. **54**, 141–150 (1991)
20. Ingmar, L., de la Banda, M.G., Stuckey, P.J., Tack, G.: Modelling diversity of solutions. In: AAAI Conference on Artificial Intelligence, AAAI 2020, pp. 1528–1535. AAAI Press (2020)
21. Katz, M., Sohrabi, S.: Reshaping diverse planning. In: AAAI Conference on Artificial Intelligence, AAAI 2020, pp. 9892–9899. AAAI Press (2020)
22. Kellerhals, L., Renken, M., Zschoche, P.: Parameterized algorithms for diverse multistage problems. In: ESA. LIPIcs, vol. 204, pp. 55:1–55:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
23. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. Algorithmica **65**(4), 754–771 (2013)
24. Macedo, J., Lopes, D., Correia, J., Machado, P., Costa, E.: Evolving visually-diverse graphic design posters. In: Johnson, C., Rebelo, S.M., Santos, I. (eds.) EvoMUSART 2024. LNCS, vol. 14633, pp. 265–278. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-56992-0_17
25. Medina, A., Richey, M., Mueller, M., Schrum, J.: Evolving flying machines in minecraft using quality diversity. In: Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 1418–1426 (2023)
26. Mkhatshwa, S., Nitschke, G.: The impact of morphological diversity in robot swarms. In: Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 65–74 (2023)
27. Mouret, J., Clune, J.: Illuminating search spaces by mapping elites. CoRR abs/1504.04909 (2015)
28. Neumann, A., Bossek, J., Neumann, F.: Diversifying greedy sampling and evolutionary diversity optimisation for constrained monotone submodular functions. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 261–269. ACM (2021)
29. Neumann, A., et al.: Diversity optimization for the detection and concealment of spatially defined communication networks. In: Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 1436–1444. ACM (2023)
30. Nikfarjam, A., Bossek, J., Neumann, A., Neumann, F.: Entropy-based evolutionary diversity optimisation for the traveling salesperson problem. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 600–608. ACM (2021)
31. Nikfarjam, A., Neumann, A., Neumann, F.: Evolutionary diversity optimisation for the traveling thief problem. In: Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 749–756. ACM (2022)
32. Nikfarjam, A., Neumann, A., Neumann, F.: On the use of quality diversity algorithms for the traveling thief problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 260–268 (2022)
33. Nikfarjam, A., Rothenberger, R., Neumann, F., Friedrich, T.: Evolutionary diversity optimisation in constructing satisfying assignments. In: Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 938–945. ACM (2023)
34. Nikfarjam, A., Stanford, T., Neumann, A., Dumuid, D., Neumann, F.: Quality diversity approaches for time use optimisation to improve health outcomes. In: Genetic and Evolutionary Computation Conference, GECCO 2024. ACM (2024, to appear)

35. Oliveto, P.S., He, J., Yao, X.: Evolutionary algorithms and the vertex cover problem. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 1870–1877. IEEE (2007)
36. Oliveto, P.S., He, J., Yao, X.: Analysis of population-based evolutionary algorithms for the vertex cover problem. In: IEEE Congress on Evolutionary Computation, CEC 2008, pp. 1563–1570. IEEE (2008)
37. Pugh, J.K., Soros, L.B., Szerlip, P.A., Stanley, K.O.: Confronting the challenge of quality diversity. In: Genetic and Evolutionary Computation Conference, GECCO 2015, pp. 967–974. ACM (2015)
38. Ulrich, T., Thiele, L.: Maximizing population diversity in single-objective optimization. In: Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 641–648. ACM (2011)
39. Wineberg, M., Oppacher, F.: The underlying similarity of diversity measures used in evolutionary computation. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1493–1504. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45110-2_21
40. Zardini, E., Zappetti, D., Zambrano, D., Iacca, G., Floreano, D.: Seeking quality diversity in evolutionary co-design of morphology and control of soft tensegrity modular robots. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 189–197. ACM (2021)

Proven Runtime Guarantees for How the MOEA/D: Computes the Pareto Front from the Subproblem Solutions

Benjamin Doerr[1], Martin S. Krejca[1(✉)], and Noé Weeks[2]

[1] Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
martin.krejca@polytechnique.edu
[2] École Normale Supérieure, Paris, France

Abstract. The decomposition-based multi-objective evolutionary algorithm (MOEA/D) does not directly optimize a given multi-objective function f, but instead optimizes $N+1$ single-objective subproblems of f in a co-evolutionary manner. It maintains an archive of all non-dominated solutions found and outputs it as approximation to the Pareto front. Once the MOEA/D found all optima of the subproblems (the g-optima), it may still miss Pareto optima of f. The algorithm is then tasked to find the remaining Pareto optima directly by mutating the g-optima.

In this work, we analyze for the first time how the MOEA/D with only standard mutation operators computes the whole Pareto front of the OneMinMax benchmark when the g-optima are a strict subset of the Pareto front. For standard bit mutation, we prove an expected runtime of $O(nN\log n + n^{n/(2N)}N\log n)$ function evaluations. Especially for the second, more interesting phase when the algorithm start with all g-optima, we prove an $\Omega(n^{(1/2)(n/N+1)}\sqrt{N}2^{-n/N})$ expected runtime. This runtime is super-polynomial if $N = o(n)$, since this leaves large gaps between the g-optima, which require costly mutations to cover.

For power-law mutation with exponent $\beta \in (1,2)$, we prove an expected runtime of $O\left(nN\log n + n^\beta \log n\right)$ function evaluations. The $O\left(n^\beta \log n\right)$ term stems from the second phase of starting with all g-optima, and it is independent of the number of subproblems N. This leads to a huge speedup compared to the lower bound for standard bit mutation. In general, our overall bound for power-law suggests that the MOEA/D performs best for $N = O(n^{\beta-1})$, resulting in an $O(n^\beta \log n)$ bound. In contrast to standard bit mutation, smaller values of N are better for power-law mutation, as it is capable of easily creating missing solutions.

Keywords: MOEA/D · multi-objective optimization · runtime analysis · power-law mutation

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 197–212, 2024.
https://doi.org/10.1007/978-3-031-70071-2_13

1 Introduction

Many real-world problems require the simultaneous optimization of different objectives. In this setting, known as *multi-objective optimization*, different solutions may not be comparable based on their objective values, as one solution can win over another solution in one objective, but lose in another objective. This results in a set of incomparable optimal objective values, commonly referred to as *Pareto front*. The aim in multi-objective optimization is to find the Pareto front of a problem, or a good approximation thereof.

Due to their population-based and heuristic nature, evolutionary algorithms lend themselves very well to multi-objective optimization, and they have been successfully applied for decades to a plethora of hard multi-objective optimization problems [32]. This strong interest has led to a variety of algorithms [24,32], following different paradigms.

From the early days of theoretical analyses of evolutionary algorithms on, multi-objective evolutionary algorithms have been analyzed also via theoretical means [13,17,21,22]. This area saw a boost of activity in the last two years, when the first mathematical runtime analysis of the NSGA-II [31] inspired many deep analyses of this important algorithm and variants such as the NSGA-III or SMS-EMOA [1–4,10–12,20,23,26–30].

A substantially different, yet also very important algorithm is the *decomposition-based multi-objective evolutionary algorithm* (MOEA/D) [25]. It decomposes a multi-objective optimization problem f into various single-objective subproblems of f. These subproblems are optimized in parallel. While doing so, the non-dominated solutions for f are maintained in an archive.

Despite its popularity in empirical research and good performance in real-world problems [24,32], the MOEA/D has not been extensively studied theoretically [5,15,16,19]. In particular, it is currently not known how the basic MOEA/D using only standard mutation operators as variation operators finds Pareto optima that are not already an optimum of one of the subproblems (we refer to Sect. 2 for a detailed discussion of the previous works). We recall that the MOEA/D solves a number of single-objective subproblems essentially via single-objective approaches. Hence, if all Pareto optima of the original problem appear as optima of subproblems (we call these *g-optima* in the remainder), this is the setting regarded, e.g., in [19], then the multi-objective aspect of the problem vanishes and the only question is how efficiently the single-objective approaches solve the subproblems.

Our Contribution. Naturally, in a typical application of the MOEA/D, one cannot assume that the subproblems are sufficiently numerous and evenly distributed so that each Pareto optimum appears as g-optimum. To better understand how the MOEA/D copes with such situations, we study in this work mathematically how the MOEA/D computes the full Pareto front when started with a population consisting of all g-optima. In this first work on this topic, as often in the mathematical runtime analysis, we consider the basic OneMinMax

(OMM) benchmark [14]. As in most previous works, we assume that the $N+1$ subproblems are constructed in a manner that the corresponding g-optima are spread equidistantly, with respect to the Hamming distance, across the Pareto front. We regard the basic MOEA/D using standard bit mutation as only variation operator. Interestingly, this is the first theoretical analysis of the MOEA/D in this setting (apart from the case that the g-optima cover the Pareto front [19]). Hence our results (Theorem 2), together with standard estimates on this time to compute the g-optima (Corollary 3), also give the first estimates on the full runtime of the MOEA/D in this setting (Corollary 1).

Since our results show that the main bottleneck for computing points on the Pareto front that are far from all g-optima is that standard bit mutation rarely flips many bits (due to the concentration behavior of the binomial distribution), we also resort to the heavy-tailed mutation operator proposed by Doerr et al. [9]. Interestingly, this allows for drastic speed-ups when considering the phase of the optimization that starts with all g-optima (Theorem 3).

In detail, our various results provide us with expected-runtime estimates for the MOEA/D with either mutation operator to optimize OMM (Corollaries 1 and 2). These results prove, respectively, an expected number of $O(nN \log n + n^{n/(2N)} N \log n)$ function evaluations for the MOEA/D with standard bit mutation where n is the problem dimension and $N+1$ is the population size, and $O\left(nN \log n + n^{\beta} \log n\right)$ expected function evaluations for power-law mutation with power-law exponent $\beta \in \mathbb{R}_{>1}$ being a constant. In both results, the second term refers to the interesting phase where the algorithm is initialized with all g-optima and is tasked to find the remaining Pareto optima of OMM.

Our overall bound for standard bit mutation yields $O(n^2 \log n)$ in the best case of $N = n$, matching the result by Li et al. [19, Proposition 4]. For general N, the second term in our bound suggests that the MOEA/D performs best if $N \in [\frac{n}{2}..n]$ and that the runtime is super-polynomial once $N = o(n)$. Moreover, we prove a lower bound of $\Omega(n^{(1/2)(n/N+1)}\sqrt{N}2^{-n/N})$ for this second term (Theorem 2), which supports this runtime characterization. However, this lower bound is not necessarily applicable to the *entire* optimization of the MOEA/D on OMM, as we only prove it for a certain phase of the optimization process. Nonetheless, we believe that it is true for sufficiently large n. We go more into detail about this behavior in Sect. 5 and especially in Sect. 5.1. Overall, our bounds suggest that standard bit mutation performs better when N is large.

Our upper bound for power-law mutation is best once $N = O(n^{\beta-1})$, resulting in a runtime bound of $O(n^{\beta} \log n)$. The bound $O(n^{\beta} \log n)$ stems from the second phase of the optimization, where the algorithm is initialized with all g-optima (Theorem 3). It is noteworthy that this bound does not depend on N, as, roughly, the time to fill in the gaps between g-optima is inversely proportional to the cost of performing $N+1$ function evaluations each iterations. Hence, the parameter N only influences our bound for the first phase of finding all g-optima. Overall, this suggests that the MOEA/D performs better when N is small, which is opposite of the behavior with standard bit mutation. Moreover,

the bound for power-law mutation is $O(n^2 \log n)$ in the worst case, matching the best case of our bound for standard bit mutation. This suggests that power-law mutation is more preferable than standard bit mutation in our setting, as power-law mutation exhibits a far more robust runtime behavior.

Last, for each of the two phases we consider, we get independent results that hold for a larger class of algorithms (Lemmas 3 and 4) or functions (Theorems 2 and 3). Moreover, we prove an anti-concentration bound for the hypergeometric distribution close around its expected value (Lemma 2). Due to the more general setting, we believe that these results are of independent interest.

2 Related Work

Theoretical analyses of the MOEA/D so far are scarce. Most results do not consider the MOEA/D with only standard mutation operators. And those that do make simplifying assumptions about the decomposition, using problem-specific knowledge. We also note that we could not find any proofs for how the MOEA/D finds its reference point (one of its parameters), which is important in the general scenarios. If the reference point is mentioned, it is immediately assumed to be best-possible.

Li et al. [19] conducted the first mathematical analysis of the MOEA/D. They study the runtime of the algorithm on the classic bi-objective OneMinMax (OMM) [14] and LeadingOnesTrailingZeros (LOTZ)[1] benchmarks of size n, that is, the number of objective-function evaluations until the MOEA/D finds all Pareto optima of the problem. Both benchmarks have a Pareto front of size $n + 1$. The authors consider a decomposition of each problem into $n + 1$ subproblems—matching the size of the Pareto front—, and they assume that the MOEA/D uses standard bit mutation. The authors prove that if the subproblems are chosen such that the $n + 1$ g-optima correspond one-to-one to the Pareto front of the problem, then the MOEA/D optimizes OMM in $O(n^2 \log n)$ expected function evaluations, and LOTZ in $O(n^3)$. We note that this requires a different subproblem structure for OMM and LOTZ. Moreover, the authors show that if the MOEA/D uses the commonly used subproblem structure of OMM for LOTZ, then the g-optima do not cover the entire Pareto front of LOTZ, for sufficiently large n. In this case, the authors argue that optimization takes a long time, as the missing solutions need to be found, but these arguments are not made formal.

Huang and Zhou [15] analyze the MOEA/D when using contiguous hypermutations, a non-standard mutation operator stemming from artificial immune systems. The authors study two versions of contiguous hypermutation, and they consider the OMM and the LOTZ benchmarks as well as a deceptive bi-objective

[1] We note that the authors call OMM COCZ, and that they consider a version of LOTZ, called LPTNO, that considers strings over $\{-1, 1\}$ instead of binary strings, effectively replacing 0s with -1s. This changes some values for certain parameters but effectively results in the same bounds, to the best of our knowledge. Hence, we use the names OMM and LOTZ throughout this section.

problem and one containing a plateau. Moreover, the authors consider a general decomposition of each problem into $N + 1$ subproblems. This decomposition is always the same and corresponds to the commonly used one, which results in evenly spread g-optima on the Pareto front of OMM, also analyzed by Li et al. [19] above. Huang and Zhou [15] prove that both MOEA/D variants optimize each of the four benchmarks in $O(Nn^2 \log n)$ expected function evaluations. Moreover, they prove the same runtime bounds for the 4-objective versions of OMM and LOTZ. Overall, their results suggest that a choice of $N = O(1)$ is more beneficial, as the hypermutation operators are capable to cover larger distances than standard bit mutation and can thus find Pareto optima efficiently that do not correspond to g-optima.

Huang et al. [16] analyze a variant of the MOEA/D that employs standard bit mutation as well as one-point crossover at a rate of 0.5. When performing crossover, one of the parents is the best solution of the current subproblem g and the other one is chosen uniformly at random among the non-dominated solutions of the subproblems closest to g with respect to the Euclidean distance in one of their parameters. The problem decomposition is always the commonly used one with equidistant g-optima, as in the two papers discussed above. The authors consider essentially the same four[2] problems as Huang and Zhou [15] above, and they also consider a general number of $N + 1$ subproblems. For LOTZ and the plateau function, the authors proved an expected runtime of $O(Nn^2)$ function evaluations, and an $O(Nn \log n)$ bound for OMM and the deceptive function.

Very recently, Do et al. [5] analyzed the MOEA/D on the multi-objective minimum-weight-base problem, which is an abstraction of classical NP-hard combinatorial problems. Their MOEA/D variant uses a decomposition based on weight scalarization, different from the previous works above. The authors then prove that this variant finds an approximation of the Pareto front of the problem within expected fixed-parameter polynomial time.

3 Preliminaries

We denote the natural numbers by $\mathbb{N}$, including 0, and the real numbers by $\mathbb{R}$. For $a, b \in \mathbb{R}$, let $[a..b] = [a, b] \cap \mathbb{N}$ and $[a] = [1..a]$.

Let $n \in \mathbb{N}_{\geq 1}$. We consider *bi-objective* optimization problems, that is, functions $f\colon \{0,1\}^n \to \mathbb{R}^2$. We always assume that the dimension $n \in \mathbb{N}_{\geq 1}$ is given implicitly. When using big-O notation, it refers to asymptotics in this n. In this sense, an event occurs *with high probability* if its probability is at least $1 - o(1)$.

We call a point $x \in \{0,1\}^n$ an *individual* and $f(x)$ the *objective value* of x. For all $i \in [n]$ and $j \in [2]$, we let x_i denote the i-th component of x and $f_j(x)$ the j-th component of $f(x)$. Moreover, let $|x|_0$ denote the number of 0s of x, and let $|x|_1$ denote its number of 1s.

For all $u, v \in \mathbb{R}^2$, we say that v *weakly dominates* u (written $v \succeq u$) if and only if for all $i \in [2]$ holds that $f_i(v) \geq f_i(u)$. We say that v *strictly dominates*

[2] The authors actually treat LOTZ and LPTNO as two functions, but the results are the same, which is why we count it as one problem.

u if and only if one of these inequalities is strict. We extend this notation to individuals, where a dominance holds if and only if it holds for their respective objective values.

We consider the maximization of bi-objective functions f, that is, we are interested in $\succeq$-maximal elements, called *Pareto-optimal* individuals. The set of all objective values that are not strictly dominated, that is, the set $F^* := \{v \in \mathbb{R}^2 \mid \exists y \in \{0,1\}^n \nexists x \in \{0,1\}^n \colon v = f(y) \wedge f(x) \succ v\}$, is called the *Pareto front* of f.

OneMinMax. We analyze the OneMinMax (OMM) benchmark [14] problem, which returns for each individual the number of 0s as the first objective, and the number of 1s as the second objective. Formally, OMM: $x \mapsto (|x|_0, |x|_1)$.

Note that each individual is Pareto optimal. The Pareto front of OneMinMax is $\{(i, n-i), i \in [0..n]\}$.

Mathematical Tools. We use the well-known multiplicative drift theorem [7] with tail bounds [6]. We present the theorem in a fashion that is sufficient for our purposes. Throughout this article, if we write for a stopping time T and two unary formulas P and Q that for all $t \in \mathbb{N}$ with $t < T$ holds that $P(t) \geq Q(t)$, then we mean that for all $t \in \mathbb{N}$ holds that $P(t) \cdot \mathbb{1}\{t < T\} \geq Q(t) \cdot \mathbb{1}\{t < T\}$, where $\mathbb{1}$ denotes the indicator function.

Theorem 1 (Multiplicative drift [7], upper tail bound [6][18, Theorem 2.4.5]). *Let $n \in \mathbb{N}$, let $(X_t)_{t \in \mathbb{N}}$ be a random process over $[0..n]$, and let $T = \inf\{t \in \mathbb{N} \mid X_t = 0\}$. Moreover, assume that there is a $\delta \in \mathbb{R}_{>0}$ such that for all $t \in \mathbb{N}$ with $t < T$ holds that $\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq \delta X_t$. Then, $\mathrm{E}[T] \leq \frac{1}{\delta}(1 + \ln n)$, and for all $r \in \mathbb{R}_{\geq 0}$ holds that $\Pr[T > \frac{1}{\delta}(r + \ln n)] \leq \mathrm{e}^{-r}$.*

4 The MOEA/D

We analyze the *decomposition-based multi-objective evolutionary algorithm* (MOEA/D; Algorithm 1) [25] for multi-objective optimization. The MOEA/D decomposes its objective function into a pre-specified number of single-objective subproblems. These subproblems are optimized in parallel. The MOEA/D maintains an archive of the $\succeq$-best solutions found so far, allowing it to find Pareto-optimal solutions for the original problem while only explicitly optimizing the subproblems.

More formally, given, besides others, a bi-objective optimization problem f as well as a *decomposition number* $N \in [n]$ and a *weight vector* $w \in [0,1]^{N+1}$, the MOEA/D optimizes f by decomposing f into $N+1$ single-objective subproblems $\{g_i \colon \{0,1\}^n \times \mathbb{R}^2 \to \mathbb{R}\}_{i \in [0..N]}$, weighted by w. Each of these subproblems is subject to *minimization* (of an error). The MOEA/D maintains a population of $N+1$ individuals $(x_i)_{i \in [0..N]} \in (\{0,1\}^n)^{N+1}$ as well as a *reference point* $z^* \in \mathbb{R}^2$ such that for each $i \in [0..N]$, individual x_i is the currently best solution found for subproblem i with respect to z^*. Ideally, the reference point is a point such

Algorithm 1: The MOEA/D [25] maximizing a bi-objective problem $f\colon \{0,1\}^n \to \mathbb{R}^2$. See also Section 4.

Input: A decomposition number $N \in \mathbb{N}_{\geq 1}$, a weight vector $w \in [0,1]^{N+1}$, subproblems $\{g_i\}_{i\in[0..N]}$, a mutation operator $\text{mut}\colon \{0,1\}^n \to \{0,1\}^n$, and a termination criterion.

1 *Initialization*: for each $i \in [0..N]$, choose x_i uniformly at random from $\{0,1\}^n$; set $z_1^* = \max_{i\in[0..N]} f_1(x_i)$, $z_2^* = \max_{i\in[0..N]} f_2(x_i)$, and iteratively add $f(x_i)$ to P if there is no $j \in [0..i-1]$ such that x_i is weakly dominated by x_j;
2 **while** *stopping criterion is not met* **do**
3 **for** *each subproblem* $i \in [0..N]$ **do**
4 *Mutation*: $y \leftarrow \text{mut}(x_i)$;
5 *Update* z^*: set $z_1^* \leftarrow \max(z_1^*, f_1(y))$, $z_2^* \leftarrow \max(z_2^*, f_2(y))$;
6 *Update* x_i: if $g_i(y, z^*) \leq g_i(x_i, z^*)$, then $x_i \leftarrow y$;
7 *Update* P: remove all elements weakly dominated by $f(y)$ from P and add $f(y)$ to P if it is not weakly dominated by an element of P;

that for all $j \in [2]$, value z_j^* is optimal for objective f_j. To this end, the MOEA/D updates z^* whenever it optimizes a subproblem. Moreover, the algorithm maintains a set $P \subseteq \mathbb{R}^2$ (the *Pareto front*) of non-dominated objective values.

We consider subproblems that measure the maximum distance to the reference point, known as Chebyshev approach. That is, for all $i \in [0..N]$, $x \in \{0,1\}^n$, and $z^* \in \mathbb{R}^2$, it holds that

$$g_i(x, z^*) = \max\big(w_i \cdot |z_1^* - f_1(x)|, (1 - w_i) \cdot |z_2^* - f_2(x)|\big). \tag{1}$$

When minimizing subproblem $i \in [0..N]$, the MOEA/D picks x_i as parent and mutates it according to a given mutation operator. Afterward, it compares the offspring of the mutation to x_i and selects the better one.[3]

We define the *runtime* of the MOEA/D on f as the number of function evaluations of f until the Pareto front P of the algorithm is equal to the Pareto front F^* of f for the first time.

In this article, we consider the MOEA/D with different mutation operators.

Mutation Operators. We consider both *standard bit mutation* as well as *power-law mutation* [9]. Let $x \in \{0,1\}^n$ be the input (the *parent*) of the mutation. Both operators create a new individual (the *offspring*) by first copying x and then adjusting its bit values. Standard bit mutation flips, for all $i \in [n]$, bit x_i independently with probability $1/n$.

Power-law mutation requires a parameter $\beta > 1$ (the *power-law exponent*) as input and utilizes the power-law distribution $\text{Pow}(\beta, n)$ over $[n]$, defined as follows. Let $C_\beta = \sum_{i\in[n]} i^{-\beta}$ as well as $X \sim \text{Pow}(\beta, n)$. For all $i \in [n]$, it holds

[3] We note that the general MOEA/D allows to specify neighborhoods among the subproblems, which exchange non-dominated solutions among each other. In this article, we focus on no exchange.

that $\Pr[X = i] = i^{-\beta}/C_\beta$. The power-law mutation first draws $X \sim \mathrm{Pow}(\beta, n)$ (the *mutation strength*) and then flips an X-cardinality subset of positions in x chosen uniformly at random.

The following lemma bounds the probability to mutate an individual with at most $\frac{n}{4}$ 0s into one with at most $\frac{n}{2}$, showing that the probability is proportional to the distance in the number of 0s. Its proof makes use of the anti-concentration of the *hypergeometric distribution* (Lemma 2) around its expectation, which we discuss after Lemma 1. We note that, due to space restrictions, the proofs of these results are on arXiv [8].

Lemma 1. *Let $x \in \{0,1\}^n$ with $u := |x|_0 \in [0..\frac{n}{4}]$ and let $v \in [u+1..\frac{n}{2}-1]$. Moreover, let $\beta \in \mathbb{R}_{>1}$, and let mut_β denote the power-law mutation with power-law exponent β. Then $\Pr[|\mathrm{mut}_\beta(x)|_0 = v] = \Omega\big((v-u)^{-\beta}\big)$.*

Hypergeometric Distribution. The hypergeometric distribution (Hyp) takes three parameters, namely, $n \in \mathbb{N}$, $k \in [0..n]$, and $r \in [0..n]$, and it has a support of $[\max(0, r+k-n)..\min(k,r)]$. A random variable $X \sim \mathrm{Hyp}(n,k,r)$ describes the number of good balls drawn when drawing r balls uniformly at random without replacement from a set of n balls, out of which k are good. That is, for all $i \in [\max(0, r+k-n)..\min(k,r)]$ holds $\Pr[X = i] = \binom{k}{i}\binom{n-k}{r-i}/\binom{n}{r}$. Moreover, $\mathrm{E}[X] = r\frac{k}{n}$ as well as $\mathrm{Var}[X] = r\frac{k}{n}(1-\frac{k}{n})\frac{n-r}{n-1}$. In the context of power-law mutation, n represents the number of bits, k the number of specific bits to flip (for example, 0-bits), and r represents the mutation strength.

The following lemma shows that the hypergeometric distribution has a reasonable probability of sampling values that deviate only by the standard deviation from its expected value.

Lemma 2. *Let $n \in \mathbb{N}$, $k \in [0..\frac{n}{2}]$, and $r \in [0..\frac{3}{4}n]$.*

Moreover, let $H \sim \mathrm{Hyp}(n,k,r)$. Then there exists a constant $\gamma \in \mathbb{R}_{>0}$ such that for all $x \in [\mathrm{E}[H] - 2\sqrt{\mathrm{Var}[H]}, \mathrm{E}[H] + 2\sqrt{\mathrm{Var}[H]}]$ it holds that $\Pr[H = x] \geq \gamma/\sqrt{\mathrm{Var}[H]}$.

5 Runtime Analysis

We analyze the MOEA/D (Algorithm 1) on the OneMinMax (OMM) function (of dimension $n \in \mathbb{N}_{\geq 1}$) with subproblems spread uniformly across the Pareto front of OMM, which is the typical way to pick weights [25] and was also used in the first mathematical analysis of the MOEA/D [19]. To this end, we make the following assumptions about the input of the algorithm, which we refer to as the *parametrization*: We consider decomposition numbers $N \in [n]$, we define the weight vector as $w = (i\frac{n}{N})_{i \in [0..N]}$, and we consider the subproblems as defined in equation (1). In our calculations, we assume that N divides n, as this avoids rounding, but we note that all results are equally valid if N does not divide n, although the computations become more verbose, adding little merit. We note that, due to space restrictions, some of our proofs are only on arXiv [8].

Our main results are the following, bounding the expected runtime for both standard bit mutation and power-law mutation.

Corollary 1. *Consider the MOEA/D maximizing* OMM *with standard bit mutation and with the parametrization specified at the beginning of Sect. 5. Then its expected runtime is* $O(nN \log n + n^{n/(2N)} N \log n)$ *function evaluations.*

Corollary 2. *Consider the MOEA/D maximizing* OMM *with power-law exponent* $\beta \in \mathbb{R}_{>1}$ *and with the parametrization specified at the beginning of Sect. 5. Then its expected runtime is* $O\left(nN \log n + n^{\beta} \log n\right)$ *function evaluations.*

Both runtime results present two terms, which stem from the two *phases* into which we separate our analysis. The first term in the results is an upper bound of the first phase, which is the number of function evaluations it takes the MOEA/D to optimize all subproblems. We call the solutions to these subproblems *g-optima*. Our bounds for either mutation operator for the first phase are the same (Corollary 3). The optimization mainly resembles performing $N+1$ times an optimization similar to that of the well-known $(1+1)$ evolutionary algorithm on the OneMax benchmark function. For $N = n$, our result for standard bit mutation recovers the result by Li et al. [19, Proposition 4]. We go into detail about the analysis in Sect. 5.1.

The second phase starts immediately after the first phase and stops once the Pareto front of the MOEA/D covers the Pareto front of OMM. During this analysis, we only assume that the MOEA/D found the g-optima so far. Thus, in the worst case, it still needs to find all other Pareto-optima of OMM. To this end, each such optimum needs to be created via mutation *directly* from one of the g-optima, as the latter are not being replaced, due to them being optimal. Depending on the *gap size*, that is, the number of Pareto optima between two g-optima of neighboring subproblems, the mutation operator makes a big difference on the expected runtime bound. We analyze both mutation operators separately in Sect. 5.2.

Regarding Corollary 1, we see that the upper bound for standard bit mutation only handles values for $N \in [\frac{n}{2}..n]$ without any slowdown in comparison to the first phase. For smaller values, the upper bound is dominated by the second term and becomes super-polynomial once $N = o(n)$. In Theorem 2, we prove a lower bound for the second phase that shows that the expected runtime is also at least super-polynomial once $N = o(n)$. However, as this lower bound only applies to the second phase, it may be possible during the whole optimization of OMM that the algorithm finds most of the OMM-Pareto optima already during the first phase, although we conjecture this not to happen for sufficiently small N.

For power-law mutation (Corollary 2), the bound for the second phase is independent of N (Theorem 3). This shows that the power-law mutation picks up missing Pareto optima fairly quickly. In fact, even for an optimal value of $N = n$ for standard bit mutation, which results in a bound of $O(n^2 \log n)$ for the second phase, the bound for power-law mutation is still smaller by a factor of $n^{\beta-2}$, which is less than 1 if we further assume that $\beta \in (1, 2)$, which is a typical range of values used for β in practice. Moreover, for the whole optimization, the upper bound for power-law mutation is best once $N = O(n^{\beta-1})$, that is, for smaller values of N. This is in contrast to the upper bound for standard bit mutation, which gets better for larger values of N.

Overall, our results suggest that standard bit mutation profits from having many subproblems, as creating initially skipped solutions may be hard to create once all subproblems are optimized. In contrast, power-law mutation is slowed down by optimizing many subproblems in parallel. Instead, it profits from optimizing fewer such subproblems and then creating initially skipped solutions.

In the following, we first analyze the first phase (Sect. 5.1) and then the second phase (Sect. 5.2).

5.1 First Phase

Recall that the first phase considers optimization of OMM only until all subproblems are optimized, that is, until all g-optima are found. Our main result is the following and shows that finding the g-optima is not challenging for the MOEA/D, regardless of the mutation operator.

Corollary 3. *Consider the MOEA/D maximizing* OMM *with the parametrization specified at the beginning of Sect. 5. Then the expected time until P contains all g-optima of* OMM *is $O(nN \log n)$ function evaluations for both standard bit mutation and power-law mutation with power-law exponent $\beta \in \mathbb{R}_{>1}$.*

For our proof of Corollary 3, we split the first phase into two parts. The first part considers the time until the reference point z^* is optimal, that is, until $z^* = (n, n)$. For this part, only the optimization of g_0 and g_N is relevant.

The second part starts with an optimal reference point and considers the time until all g-optima are found. For this part, we consider the optimization of an arbitrary subproblem and multiply the resulting time by roughly $N \log N$, as we consider $N + 1$ subproblems and we wait until all of them are optimized.

In order to prove results that hold for the MOEA/D with standard bit mutation as well as with power-law mutation, we consider a general mutation operator, which we call *general mutation*. It takes one parameter $p_1 \in (0, 1]$ and works as follows: It chooses to flip exactly one bit in the parent with probability p_1 (and it flips any other number with probability $1 - p_1$). Conditional on flipping exactly one bit, it flips one of the n bits of the parent uniformly at random and returns the result. Note that standard bit mutation is general mutation with $p_1 = (1 - \frac{1}{n})^{n-1}$ and that power-law mutation with power-law exponent β is general mutation with $p_1 = 1/C_\beta$. Both of these values are constants.

For the first part, we get the following result.

Lemma 3. *Consider the MOEA/D maximizing* OMM *with the parametrization specified at the beginning of Sect. 5 and with general mutation with parameter $p_1 \in (0, 1]$. Then the expected time until $z^* = (n, n)$ holds is $O(\frac{n}{p_1} N \log n)$ function evaluations.*

Proof. Let T be the first iteration such that $z_1^* = n$. Without loss of generality, we only analyze $\mathrm{E}[T]$, as the analysis for $z_2^* = n$ is identical when changing all 1s into 0s and vice versa in the individuals in all arguments that follow. Thus, the expected runtime that we aim to prove is at most $2\,\mathrm{E}[T]$ by linearity

of expectation. Hence, we are left to show that $\mathrm{E}[T] = O(\frac{n}{p_1} \log n)$, where the factor of N in the bound of Lemma 3 stems from the MOEA/D making $N+1$ function evaluations per iteration. To this end, we only consider the optimization of g_N.

By Eq. (1), the choice of the weight vector w, and the definition of OMM, it follows for all $x \in \{0,1\}^n$ and $z \in \mathbb{R}^2$ that $g_N(x,z) = \max(|z_1 - |x|_1|, 0) = |z_1 - |x|_1|$. In each iteration, let x_N denote the best-so-far solution for g_N at the beginning of the iteration, and let y denote its offspring generated via mutation. Note that, due to how z^* is initialized and updated, in each iteration, it holds that $z_1^* \geq \max(f_1(x_N), f_1(y)) = \max(|x_N|_1, |y|_1)$. Thus, if $|y|_1 > |x_N|_1$, then $g(y, z^*) < g(x_N, z)$, and thus x_N is updated to y at the end of the iteration. Hence, the optimization of g_N proceeds like a $(1+1)$-EA-variant with general mutation optimizing OneMax.

More formally, let $(X_t)_{t \in \mathbb{N}}$ such that for each $t \in \mathbb{N}$, the value X_t denotes the number of 0s in x_N at the end of iteration t. Note that $X_T = 0$. We aim to apply the multiplicative drift theorem (Theorem 1) to X with T. By the definition of the mutation operator, it follows for all $t < T$ that $\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq X_t \frac{p_1}{n}$, since it is sufficient to choose to flip one bit (with probability p_1) and then to flip one of the X_t 0s of x_N (at the beginning of the iteration), which are chosen uniformly at random. Thus, by Theorem 1, it follows that $\mathrm{E}[T] \leq \frac{n}{p_1}(1 + \ln n)$, concluding the proof. □

For the second part, we get the following result.

Lemma 4. *Consider the MOEA/D maximizing* OMM *with the parametrization specified at the beginning of Sect. 5 and with* $z^* = (n, n)$ *and with general mutation with parameter* $p_1 \in (0, 1]$. *Then the expected time until* P *contains all* g*-optima of* OMM *is* $O(\frac{n}{p_1} N \log n)$ *function evaluations.*

Proof. Let $i \in [0, N]$. We bound with high probability the time T until g_i is optimized, only counting the function evaluations for subproblem i. The result of Lemma 4 follows then by considering the maximum runtime among all values of i and multiplying it by $N+1$, as we perform $N+1$ function evaluations per iteration and optimize all subproblems in parallel. We bound the maximum with high probability by taking a union bound over all $N+1$ different values for i. If the maximum of T over all i is at least $B \in \mathbb{R}_{\geq 0}$ with probability at most $q \in [0, 1)$, then we get the overall expected runtime by repeating our analysis $\frac{1}{1-q}$ times in expectation, as the actual runtime is dominated by a geometric random variable with success probability $1 - q$. The overall expected runtime is then $O(BN\frac{1}{1-q})$ Thus, it remains to show that $\Pr[T > \frac{n}{p_1}\big(\ln(n) + 2\ln(N+1)\big)] \leq (N+1)^{-2}$, as it then follows that $q \leq (N+1)^{-1}$ and thus $\frac{1}{1-q} \leq 2$. We aim to prove this probability bound with the multiplicative drift theorem (Theorem 1).

Let $(X_t)_{t \in \mathbb{N}}$ be such that for all $t \in \mathbb{N}$, value X_t denotes the value of $g_i(x_i, z^*)$ at the beginning of the iteration. Note that $X_T = 0$ and that for all $t < T$ holds that $X_t \in [0..n]$ and that for X_t to reduce (if $X_t > 0$), it is sufficient to flip one of the X_t bits that reduce the distance. Thus, by the definition of the mutation

operator, it follows that $\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq X_t \frac{p_1}{n}$. Overall, by (Theorem 1), it follows that $\Pr[T > \frac{n}{p_1}(\ln(n)+2\ln(N+1))] \leq (N+1)^{-2}$. The proof is concluded by noting that $N \leq n$ and thus $\ln(n) + 2\ln(N+1) = O(\log n)$. □

By the linearity of expectation, the expected time of the first phase is the sum of the expected runtimes of both parts. Moreover, since standard bit mutation and power-law mutation are both a general mutation with $p_1 = \Theta(1)$, we can omit p_1 in the asymptotic notation. Overall, we obtain Corollary 3.

5.2 Second Phase

Recall that the second phase assumes that the MOEA/D starts with the g-optima as its solutions to the subproblems, and it lasts until all OMM-Pareto optima are found.

For this phase, the actual objective function is not very important. All that matters is that if the solutions $(x_i)_{i\in[0..N]}$ of the MOEA/D are such that for all $i \in [0..N]$ holds that $|x_i|_1 = i\frac{n}{N}$, then x_i is optimal for g_i. We refer to such a situation as *evenly spread g-optima*.

Since there is a drastic difference between the runtimes of standard bit mutation and power-law mutation, we analyze these two operators separately.

Standard Bit Mutation. Since the standard bit mutation is highly concentrated around flipping only a constant number of bits, it does not perform well when it needs to fill in larger gaps. The following theorem is our main result, and it proves an upper and a lower bound for the expected runtime of the second phase. These bounds are not tight, but they show that the runtime is already super-polynomial once $N = o(n)$.

Theorem 2. *Consider the MOEA/D maximizing a bi-objective function with evenly spread g-optima and with standard bit mutation, using the parametrization specified at the beginning of Sect. 5. Moreover, assume that $\frac{n}{2N}$ is integer and that the algorithm is initialized with $(x_i)_{i\in[0..N]}$ such that for all $i \in [0..N]$ holds that $|x_i|_1 = i \cdot \frac{n}{N}$. Then the expected runtime until for each $j \in [0..n]$ at least one individual with j 0s is created via mutation is $O(n^{n/(2N)}N\log n) \cap \Omega(n^{(1/2)(n/N+1)}\sqrt{N}2^{-n/N})$ function evaluations.*

Power-Law Mutation. The power-law mutation allows to create individuals at larger distance from its parent with far higher probability than standard bit mutation. Our main result is the following theorem, which shows that the MOEA/D with power-law mutation optimizes the second phase of OMM efficiently. As before, we state this theorem in a more general fashion.

Theorem 3. *Consider the MOEA/D optimizing a bi-objective function with evenly spread g-optima, using the parametrization specified at the beginning of Sect. 5. Moreover, assume that the algorithm is initialized with $\{i \cdot \frac{n}{N}\}_{i\in[0..N]}$. Then the expected runtime until for each $j \in [0..n]$ at least one individual with j 0s is created via mutation is $O(n^\beta \log n)$.*

The bound of Theorem 3 does not depend on N, in contrast to the bound on the first phase (Corollary 3). The reason for our bound not depending on N is roughly that the effort to fill in a gap between to g-optima is inversely proportional to the cost of an iteration, namely, $N+1$. Thus, a smaller value of N leads to faster iterations but more iterations spend to fill the gaps, and vice versa.

Our (omitted) proof of Theorem 3 makes use of the following lemma, which bounds the probability to create a specific individual from any of the g-optima in a single iteration of the MOEA/D.

Lemma 5. *Consider a specific iteration during the optimization of the MOEA/D of a bi-objective function with evenly spread g-optima, using the parametrization specified at the beginning of Sect. 5. Moreover, assume that the algorithm is initialized with $\{i \cdot \frac{n}{N}\}_{i \in [0..N]}$. Last, let $u \in [0..\frac{n}{2}]$. Then the probability that an individual with u 0-bits is produced during mutation in this iteration is $\Omega\left(N(n^{-\beta})\right)$.*

Proof. Let i be such that $ni/N < u \leq n(i+1)/N$. Clearly, $i \leq N/2$, as $u \leq n/2$. Note that are at least $i/4$ values of $j \in [0..N/4]$ such that $jn/N \leq n/4$. By Lemma 1, each individual x_j mutates into an individual with u 0-bits with probability $\Omega\big(((i+1)n/N\big)^{-\beta}\big)$. Because there are $\Omega(i)$ such possible mutations during an iteration, the probability of generating at least one during an iteration is $\Omega\left(i^{1-\beta}n^{-\beta} \cdot N^{\beta}\right) = \Omega(N\big(\frac{N}{i}\big)^{\beta-1}n^{-\beta}) = \Omega\left(Nn^{-\beta}\right)$, concluding the proof. □

6 Conclusion

We studied the impact of the decomposition number N of the MOEA/D [25] on the classic multi-objective benchmark OneMinMax (OMM) [14] theoretically. Our analyses considered subproblems that are evenly spread out on the Pareto front of OMM. Especially, we studied the expected runtime when starting with all optima of the subproblems (the *g-optima*) and requiring to find the remaining Pareto optima of OMM.

One of our theoretical results (Theorem 3) shows that using power-law mutation allows the MOEA/D to efficiently find the entire Pareto front of OMM even if it is initialized only with the g-optima. Interestingly, this bound is independent of the number of problems N and thus the number of initially missing Pareto optima between two neighboring g-optima. Together with our general bound for finding all g-optima (Corollary 3), this shows that the MOEA/D with power-law mutation always optimizes OMM efficiently (Corollary 2). Depending on N, our total-runtime bound ranges from $O(n^2 \log n)$ in the worst case to $O(n^{\beta} \log n)$ in the best case of $N = O(n^{\beta-1})$. This suggests that the MOEA/D is, in fact, slowed down when using many subproblems, despite large values of N implying that the subproblems cover the Pareto front of OMM better than smaller values. The reason is that a large value of N roughly translates to optimizing the same problem N times. With the power-law mutation, it is better to optimize fewer and therefore more diverse subproblems and to then find the remaining Pareto optima efficiently via the power-law mutation.

For standard bit mutation, when starting with all g-optima, we show (Theorem 2) that the MOEA/D is not capable of efficiently finding all Pareto optima if N is sufficiently small, as our lower bound is super-polynomial for $N = o(n)$. Nonetheless, for $N = \Theta(n)$, the expected runtime of the MOEA/D with standard bit mutation is polynomial in this part. This translates to an overall polynomial expected runtime (Corollary 1) for $N = \Theta(n)$ that it is even $O(n^2 \log n)$ for $N \in [\frac{n}{2}..n]$, matching our worst-case bound with power-law.

Overall, our results suggest a clear benefit of power-law mutation over standard bit mutation of the MOEA/D in the setting we considered. Not only is the power-law variant faster, it is also far more robust to the choice of N and thus to how the problem is decomposed.

For future work, it would be interesting to improve the upper bounds or prove matching lower bounds. A similar direction is to consider an exchange of best-so-far solutions among the subproblems. The classic MOEA/D supports such an exchange, which could potentially lead to finding the g-optima more quickly. Another promising direction is the study of different problem decompositions, for example, not-evenly spread subproblems or subproblem definitions different from equation (1). Last, we considered the OMM setting, with a stronger generalization some of our results (Theorems 2 and 3). However, it is not clear to what extent the benefit of power-law mutation carries over to problems with an entirely different structure, such as LeadingOnesTrailingZeros.

Acknowledgments. This research benefited from the support of the FMJH Program Gaspard Monge for optimization and operations research and their interactions with data science.

Disclosure of Interests. The authors declare no competing interests.

References

1. Bian, C., Qian, C.: Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) Parallel Problem Solving From Nature, PPSN 2022, pp. 428–441. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14721-0_30
2. Cerf, S., Doerr, B., Hebras, B., Kahane, J., Wietheger, S.: The first proven performance guarantees for the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) on a combinatorial optimization problem. In: International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5522–5530. ijcai.org (2023)
3. Dang, D.C., Opris, A., Salehi, B., Sudholt, D.: Analysing the robustness of NSGA-II under noise. In: Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 642–651. ACM (2023)
4. Dang, D.C., Opris, A., Salehi, B., Sudholt, D.: A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. In: Conference on Artificial Intelligence, AAAI 2023, pp. 12390–12398. AAAI Press (2023)

5. Do, A.V., Neumann, A., Neumann, F., Sutton, A.M.: Rigorous runtime analysis of MOEA/D for solving multi-objective minimum weight base problems. In: Advances in Neural Information Processing Systems, pp. 36434–36448. Curran Associates (2023)
6. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. Algorithmica **65**, 224–250 (2013)
7. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**, 673–697 (2012)
8. Doerr, B., Krejca, M.S., Weeks, N.: Proven runtime guarantees for how the MOEA/D computes the pareto front from the subproblem solutions. CoRR abs/2405.01014 (2024)
9. 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)
10. Doerr, B., Qu, Z.: A first runtime analysis of the NSGA-II on a multimodal problem. IEEE Trans. Evol. Comput. **27**, 1288–1297 (2023)
11. Doerr, B., Qu, Z.: From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In: Conference on Artificial Intelligence, AAAI 2023, pp. 12408–12416. AAAI Press (2023)
12. Doerr, B., Qu, Z.: Runtime analysis for the NSGA-II: provable speed-ups from crossover. In: Conference on Artificial Intelligence, AAAI 2023, pp. 12399–12407. AAAI Press (2023)
13. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Congress on Evolutionary Computation, CEC 2003, pp. 1918–1925. IEEE (2003)
14. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. Evol. Comput. **18**, 335–356 (2010)
15. Huang, Z., Zhou, Y.: Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework. In: Conference on Artificial Intelligence, AAAI 2020, pp. 2359–2366. AAAI Press (2020)
16. Huang, Z., Zhou, Y., Chen, Z., He, X., Lai, X., Xia, X.: Running time analysis of MOEA/D on pseudo-Boolean functions. IEEE Trans. Cybern. **51**, 5130–5141 (2021)
17. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. IEEE Trans. Evol. Comput. **8**, 170–182 (2004)
18. 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://doi.org/10.1007/978-3-030-29414-4_2, also available at https://arxiv.org/abs/1712.00964
19. Li, Y.L., Zhou, Y.R., Zhan, Z.H., Zhang, J.: A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **20**, 563–576 (2016)
20. Opris, A., Dang, D.C., Neumann, F., Sudholt, D.: Runtime analyses of NSGA-III on many-objective problems. In: Genetic and Evolutionary Computation Conference, GECCO 2024. ACM (2024, to appear)
21. Rudolph, G.: Evolutionary search for minimal elements in partially ordered finite sets. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (eds.) EP 1998. LNCS, vol. 1447, pp. 345–353. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0040787

22. Thierens, D.: Convergence time analysis for the multi-objective counting ones problem. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Thiele, L., Deb, K. (eds.) EMO 2003. LNCS, vol. 2632, pp. 355–364. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36970-8_25
23. Wietheger, S., Doerr, B.: A mathematical runtime analysis of the Non-dominated Sorting Genetic Algorithm III (NSGA-III). In: International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5657–5665. ijcai.org (2023)
24. Zhang, J., Xing, L.: A survey of multiobjective evolutionary algorithms. In: International Conference on Computational Science and Engineering (CSE), pp. 93–100. IEEE (2017)
25. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**, 712–731 (2007)
26. Zheng, W., Doerr, B.: Better approximation guarantees for the NSGA-II by using the current crowding distance. In: Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 611–619. ACM (2022)
27. Zheng, W., Doerr, B.: Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). Artif. Intell. **325**, 104016 (2023)
28. Zheng, W., Doerr, B.: Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for many objectives. IEEE Trans. Evol. Comput. (2023, in press). https://doi.org/10.1109/TEVC.2023.3320278
29. Zheng, W., Doerr, B.: Runtime analysis of the SMS-EMOA for many-objective optimization. In: Conference on Artificial Intelligence, AAAI 2024. AAAI Press (2024)
30. Zheng, W., Li, M., Deng, R., Doerr, B.: How to use the metropolis algorithm for multi-objective optimization? In: Conference on Artificial Intelligence, AAAI 2024. AAAI Press (2024)
31. Zheng, W., Liu, Y., Doerr, B.: A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In: Conference on Artificial Intelligence, AAAI 2022, pp. 10408–10416. AAAI Press (2022)
32. 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**, 32–49 (2011)

Ranking Diversity Benefits Coevolutionary Algorithms on an Intransitive Game

Mario Alejandro Hevia Fajardo(✉) and Per Kristian Lehre

University of Birmingham, Birmingham B15 2TT, UK
m.heviafajardo@bham.ac.uk

Abstract. Competitive coevolutionary algorithms (CoEAs) often encounter so-called coevolutionary pathologies particularly cycling behavior, which becomes more pronounced for games where there is no clear hierarchy of superiority among the possible strategies (intransitive games). In order to avoid these pathologies and ensure an efficient optimisation, it has been suggested that it is critical to choose a *good* evaluation environment (set of solutions used for evaluation).

In this paper, we use runtime analysis to increase our understanding of the essential characteristics that the evaluation environments should possess to ensure efficient runtime on the intransitive problem class $\textsc{Bilinear}_{\alpha,\beta}$. For this problem class, we observe that it is beneficial to maintain a high *diversity of rankings* in the evaluation environment, that is, a set of individuals used for evaluation which are diverse in how they rank opponents.

We propose and analyse two mechanisms that implement this idea. In the first approach, we ensure diversity of rankings through an archive. In the second approach, we introduce a CoEA without an archive, but with a ranking diversity mechanism. Both approaches optimise $\textsc{Bilinear}_{\alpha,\beta}$ in expected polynomial time.

Keywords: Runtime analysis · Competitive coevolution · Archives · Maximin optimisation

1 Introduction

Within the field of Evolutionary Computation, coevolutionary algorithms stand out for harnessing the dynamic interplay between solutions (individuals) to tackle optimisation problems where computing the fitness of isolated individuals is infeasible [17]. Coevolutionary algorithms are categorized into cooperative and competitive variants based on the nature of interactions among the evolved solutions. The focus of this work is on competitive coevolutionary algorithms.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-70071-2_14.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 213–229, 2024.
https://doi.org/10.1007/978-3-031-70071-2_14

Competitive coevolutionary algorithms (we omit the competitive label and simply call them CoEAs), simultaneously evolve one or more populations of solutions that compete and adapt to one another. CoEAs do not rely on a global fitness function to assign fitness values to sampled solutions. Instead, they use the aggregation of outcomes from interactions between competing solutions to rank solutions and make selection decisions. Thus, this optimisation technique is well suited for problems that have intrinsically competitive domains or for problems where finding a suitable global fitness measure is hard. Accordingly, in this work, we use coevolutionary algorithms to find a maximin-optimum, that is, a pair of solutions (x, y) such that the performance of x against the least favourable y is the best, and the same for y.

Despite their success at solving non-trivial problems (e.g. zero-sum games [1,29], software engineering [2,14], and generative models [10,11]) CoEAs often encounter so-called *coevolutionary pathologies* [21,30]. This is particularly likely if the problem is intransitive, that is, if there is no clear hierarchy of superiority for all possible solutions [17]. Aggravating the situation, Czarnecki et al. [3] showed that real-world games tend to have intransitive dynamics.

In order to avoid these pathologies and ensure an efficient optimisation, it has been suggested that it is critical to choose a *good* evaluation environment (set of opponents) [17,22,27]. The rationale for this is that CoEAs use a varying set of opponents to assign fitness and select individuals. This may result in apparent progress (*local* progress) against a poorly selected set of opponents. However, this does not necessarily translate into real progress toward the optimal solution when comparing against the entire set of possible opponents (*global* progress) [22]. Therefore, a *good* evaluation environment that represents all possible opponents could enable global progress and as a consequence an efficient optimisation.

Previous studies have focused on guaranteeing monotonic global progress (see Sect. 2). But Popovici et al. [26] warned that monotonic progress does not guarantee an efficient optimisation and recommends instead to focus on studying the performance of algorithms and in particular the authors propose the use of runtime analysis to understand CoEAs better.

We aim to use runtime analysis to increase our understanding of the key characteristics required for evaluation environments to ensure efficient runtime on intransitive problems. In particular, we study the class of intransitive problems called $\textsc{Bilinear}_{\alpha,\beta}$ [18]. $\textsc{Bilinear}_{\alpha,\beta}$ is a challenging class of maximin-problems where CoEAs have been shown to lack a clear signal towards the optimal solutions during most of the optimisation steps and the algorithms tend to cycle around the optimum due to its intransitive properties [12,13]. We aim to find the properties of an evaluation environment that gives a clear signal towards the optimal solution, accelerating the optimisation and avoiding the cycling behaviour characteristic of these intransitive problems.

We start our analyses by isolating the evaluation environments, separating the search populations from the evaluation environments using archive populations (Archived Tournament Selection CoEA – Algorithm 1). Doing this, we observe that it is beneficial to maintain a high *diversity of rankings* in the evaluation environment, that is, given that the search populations are ranked (ordered)

by the payoff when interacting with an opponent, the individuals in the evaluation environment rank these solutions in a diverse way.

Given this insight, we propose two mechanisms. In the first approach, we ensure diversity of rankings through an archive update scheme that maintains a diverse evaluation environment. In the second approach, we introduce a CoEA without an archive that we call RankDivCoEA (Ranking Diversity CoEA). This algorithm implements an intuitive ranking diversity mechanism that maintains solutions in the current populations that rank their competitors differently without the overhead of maintaining an archive.

In Sect. 5 we then analyse ATS-CoEA with our proposed archive update scheme and show that ATS-CoEA solves $\textsc{Bilinear}_{\alpha,\beta}$ for all α and β in $O(\lambda^4 n)$ expected function evaluations, with λ representing the population size and n the problem size. Finally, in Sect. 6 we show that RankDivCoEA solves $\textsc{Bilinear}_{\alpha,\beta}$ with $\alpha n, \beta n \notin \mathbb{Z}$ in $O(\lambda^3 n)$ expected function evaluations.

2 Related Work

Archives. A common strategy for maintaining a robust evaluation environment in CoEAs involves the use of archives. An archive is a set of solutions that is systematically updated throughout the coevolutionary process and used to compare evolved solutions. The aim is that the archives accumulate knowledge of the problem and maintain diversity and progress [17].

Rosin and Belew [28] proposed the *hall of fame* archive, one of the first and simplest archives for CoEAs. The hall of fame is an unbounded archive that adds individuals that are successful against the current population and a subset of the archive. This helps CoEAs select individuals that outperform their ancestors. A simple extension of the hall of fame where the archive is bounded was made by Nolfi and Floreano [24].

Ficici and Pollack [9] proposed an unbounded archive built upon the game-theoretic principle of Nash Equilibrium called Nash memory. This archive guarantees global monotonic progress in symmetric zero-sum games [8,9]. Oliehoek et al. [25] extended it to asymmetric games maintaining the progress guarantee.

De Jong [4] introduced an unbounded archived CoEA called IPCA (Incremental Pareto Coevolution Archive). In IPCA one population is considered as a set of objectives in the sense of Evolutionary Multi-Objective Optimisation and aims to find the Pareto-front. IPCA guarantees global monotonic progress. A bounded variant named LAPCA (LAyered Pareto Coevolution Archive) was later proposed by De Jong [5] without the monotonicity guarantees. Based on the same concept Yang et al. [31] proposed EPCA (Efficient Pareto Coevolution Archive) showing experimentally that it can outperform IPCA.

Another approach when using archives is to extract dimension information (underlying objectives implicitly defined) in order to create a suitable archive. De Jong and Bucci [6] proposed one of the first archives of this kind, named DECA (Dimension Extracting Coevolutionary Algorithm) and showed empirically that it is at least as good as IPCA and LAPCA for the problems

tested. Based on this idea Yang et al. [32] and Jaśkowski and Krawiec [16] proposed SCA-BDE (Simple Coevolution Archive based on Bidirectional Dimension Extraction) and COSA (Coordinate System Archive) respectively, showing improved performances to all previous archives in the problems tested.

We remark that all mentioned archive mechanisms ensuring global monotonic progress require unlimited memory. Additionally, as shown in the later studies, archives without this guarantee can outperform these archives in practice.

Runtime Analysis of Coevolutionary Algorithms. Due to their complexity, there is little rigorous understanding of the algorithm dynamics in (cooperative and competitive) CoEAs.

Cooperative CoEAs: Jansen and Wiegand [15] rigorously analysed the runtime of a cooperative CoEA on *separable* functions and showed that problem separability does not guarantee a speedup over traditional EAs. Lehre and Lin [19] showed that the cooperative CoEA CC-$(1+1)$ EA solves all linear functions in the same asymptotic time as the $(1+1)$ EA.

Competitive CoEAs: Lehre [18] analysed for the first time the runtime of a competitive CoEA, the PD-CoEA, on some instances of the pseudo-Boolean Bilinear problem, showing that given the correct parameters the algorithm finds an ε-approximation efficiently, but a too high mutation rate leads to exponential runtime. Similarly, Hevia Fajardo and Lehre [12] analysed a $(1, \lambda)$ CoEA on a version of Bilinear that uses an integer lattice as search space, showing that the algorithm finds a solution near the maximin-optima in polynomial time with high probability if the worst-case interaction is used to assign fitness (i. e. the fitness of a solution is its worst performance) but is inefficient if the average of all interactions is used instead.

Hevia Fajardo et al. [13] analysed the runtime and total regret of a CoEA named RLS-PD on some instances of a slight variation of the pseudo-Boolean Bilinear problem. The authors showed that despite finding the optimum in $O(n^{1.5})$ the algorithm quickly forgets this solution and stays far away from it.

3 Preliminaries

We define $[n] := \{1, \dots, n\}$. We denote the 1-norm of a bit string $x \in \{0,1\}^n$ by $\|x\| := \sum_{i=1}^{n} x_i$, i.e., the number of 1-bits in x. We use the function

$$\operatorname{sign}(x) := \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0. \end{cases}$$

We denote the power set of a set $\mathcal{S}$ as $\mathbb{P}(\mathcal{S})$.

Due to space constraints, we removed the proofs and provide proof sketches instead; the detailed proofs can be found in the supplementary material[1].

In this work we introduce two algorithms, ATS-CoEA (Algorithm 1) and RankDivCoEA (Algorithm 2). The algorithms are defined for payoff functions $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ defined over arbitrary (finite) search spaces $\mathcal{X}$ and $\mathcal{Y}$.

Mutation Operator. Both Algorithms 1 and 2 can use any mutation operators $\text{mut}_x(\cdot) : \mathcal{X} \to \mathcal{X}$, $\text{mut}_y(\cdot) : \mathcal{Y} \to \mathcal{Y}$. Given that for our analysis $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$, we consider $\text{mut}_x(\cdot) = \text{mut}_y(\cdot)$. Furthermore, we use the following unbiased mutation operator.

Definition 1 (Unbiased Mutation Operator $\text{mut}_{\mathcal{D}}$). *For a probability distribution $\mathcal{D}$ on $\{0\} \cup [n]$ with probabilities $r_{(0)}, \ldots, r_{(n)} \geq 0$. Let $\text{mut}_{\mathcal{D}}$ be the mutation operator that samples $k \sim \mathcal{D}$ and then flips a uniform random set of exactly k positions.*

We note that any unary unbiased mutation operator (see [20]) can be expressed as $\text{mut}_{\mathcal{D}}$ (Lemma 1 in [7]).

Algorithm 1: Archived Tournament Selection CoEA (ATS-CoEA)

Require: Search spaces $\mathcal{X}$ and $\mathcal{Y}$.
Require: Payoff function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
Require: Population size $\lambda \in \mathbb{N}$.
Require: Tournament and competition size $2 \leq k \in [\lambda]$ $2 \leq \ell \in [\lambda]$.
Require: $\text{mut}_x(\cdot) : \mathcal{X} \to \mathcal{X}$, $\text{mut}_y(\cdot) : \mathcal{Y} \to \mathcal{Y}$.
Require: Initial populations $P_0 \in \mathcal{X}^\lambda$ and $Q_0 \in \mathcal{Y}^\lambda$.
Require: Initial archives $V_{-1} \in \mathbb{P}(\mathcal{X})$ and $W_{-1} \in \mathbb{P}(\mathcal{Y})$.
Require: Archive update scheme $\text{arch\_update}(\cdot,\cdot,\cdot,\cdot)$.
for $t \in \mathbb{N}_0$ *until termination condition satisfied* **do**
  $V_t, W_t := \text{arch\_update}(V_{t-1}, W_{t-1}, P_t, Q_t)$;
  for $i \in [\lambda]$ **do**
    Sample k predators $x_1, \ldots, x_k \sim \text{Unif}(P_t)$ i. i. d.;
    Sample ℓ prey $y_1, \ldots, y_\ell \sim \text{Unif}(W_t)$ i. i. d.;
    Select parent index $j := \arg\max_{r \in [k]} \min_{s \in [\ell]} g(x_r, y_s)$ where $j \in [k]$, breaking ties u. a. r.;
    Create $P_{t+1}(i) := \text{mut}_x(x_j)$;
  for $i \in [\lambda]$ **do**
    Sample k prey $y_1, \ldots, y_k \sim \text{Unif}(Q_t)$ i. i. d.;
    Sample ℓ predators $x_1, \ldots, x_\ell \sim \text{Unif}(V_t)$ i. i. d.;
    Select parent index $j := \arg\min_{s \in [k]} \max_{r \in [\ell]} g(x_r, y_s)$ where $j \in [\ell]$, breaking ties u. a. r.;
    Create $Q_{t+1}(i) := \text{mut}_y(y_j)$;

Algorithmic Description. ATS-CoEA uses two populations $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$ which we sometimes will refer to as the "predators" and the "prey" and two archives $V \in \mathbb{P}(\mathcal{X})$ and $W \in \mathbb{P}(\mathcal{Y})$. Initially, we make minimal assumptions about the archives: the archives can only contain individuals seen by the algorithm and are updated at the start of each generation using the current archives and the

[1] https://mariohevia.github.io/assets/pdf/PPSN_2024_sup_material.pdf.

current populations. In our analyses we assume that the initial archives are empty, but in practice they can be seeded with known solutions.

ATS-CoEA selects λ parents by using λ tournaments with k solutions sampled u. a. r. from the populations P or Q that compete against ℓ competitors sampled exclusively from the archives V or W. The best worst-case solution (a solution's fitness is its worst performance) within the tournament is selected as the parent. Afterwards an offspring is created by mutating the parent and the offspring is added to the next population.

RankDivCoEA uses two populations $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$ which we sometimes will refer to as the "predators" and the "prey". A central component of RankDivCoEA is a diversity mechanism which is defined in the same way for the predators and the prey. We now describe how it works for the predators. Each predator x imposes a total order $\leq_x$ on the set of prey $\mathcal{Y}$ given by $y_1 \leq_x y_2$ if and only if $g(x, y_1) \leq g(x, y_2)$. Clearly, different predators x lead to different orders/ranks. To achieve this, in each iteration the algorithm samples two predators x_1 and x_2, and two prey y_1 and y_2. First, the algorithm checks if $g(x_1, y_1) = g(x_1, y_2)$ and $g(x_2, y_1) = g(x_2, y_2)$. In such a case y_1 and y_2 appear to have the same phenotype, bypassing the diversity mechanism for x_1 and x_2. Therefore, in this case a random bit is flipped of either y_1 or y_2 until $g(x_1, y_1) \neq g(x_1, y_2)$ or $g(x_2, y_1) \neq g(x_2, y_2)$[2].

Algorithm 2: Ranking Diversity CoEA (RankDivCoEA)

Require: Search spaces $\mathcal{X}$ and $\mathcal{Y}$.
Require: Payoff function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
Require: Population size $\lambda \in \mathbb{N}$.
Require: $\mathrm{mut}_x(\cdot) : \mathcal{X} \to \mathcal{X}$, $\mathrm{mut}_y(\cdot) : \mathcal{Y} \to \mathcal{Y}$.
Require: Initial populations $P_0 \in \mathcal{X}^\lambda$ and $Q_0 \in \mathcal{Y}^\lambda$.
for $t \in \mathbb{N}_0$ *until termination condition satisfied* **do**
  for $i \in [\lambda]$ **do**
    Sample $x_1, x_2, x_3 \sim \mathrm{Unif}(P_t)$ and $y_1, y_2 \sim \mathrm{Unif}(Q_t)$;
    while $g(x_1, y_1) = g(x_1, y_2) \wedge g(x_2, y_1) = g(x_2, y_2)$ **do** Flip one bit u. a. r. from either y_1 or y_2 ;
    $s_1 \leftarrow \mathrm{sign}(g(x_1, y_1) - g(x_1, y_2))$;
    $s_2 \leftarrow \mathrm{sign}(g(x_2, y_1) - g(x_2, y_2))$;
    $s_3 \leftarrow \mathrm{sign}(g(x_3, y_1) - g(x_3, y_2))$;
    $x' \leftarrow \begin{cases} x_1 & \text{if } s_1 \neq s_2 \wedge s_1 \neq s_3 \\ x_1 & \text{if } s_1 = s_2 \wedge \min(g(x_1, y_1), g(x_1, y_2)) \geq \min(g(x_2, y_1), g(x_2, y_2)) \\ x_2 & \text{otherwise.} \end{cases}$
    Create $P_{t+1}(i) := \mathrm{mut}_x(x')$;
    Sample $y_1, y_2, y_3 \sim \mathrm{Unif}(Q_t)$ and $x_1, x_2 \sim \mathrm{Unif}(P_t)$;
    while $g(x_1, y_1) = g(x_2, y_1) \wedge g(x_1, y_2) = g(x_2, y_2)$ **do** Flip one bit u. a. r. from either x_1 or x_2 ;
    $s_1 \leftarrow \mathrm{sign}(g(x_1, y_1) - g(x_2, y_1))$;
    $s_2 \leftarrow \mathrm{sign}(g(x_1, y_2) - g(x_2, y_2))$;
    $s_3 \leftarrow \mathrm{sign}(g(x_1, y_3) - g(x_2, y_3))$;
    $y' \leftarrow \begin{cases} y_1 & \text{if } s_1 \neq s_2 \wedge s_1 \neq s_3 \\ y_1 & \text{if } s_1 = s_2 \wedge \max(g(x_1, y_1), g(x_2, y_1)) \leq \max(g(x_1, y_2), g(x_2, y_2)) \\ y_2 & \text{otherwise.} \end{cases}$
    Create $Q_{t+1}(i) := \mathrm{mut}_y(y')$;

[2] This mechanism does not change the original solutions in the population, they are just changed locally for the diversity mechanism.

Afterwards, if x_1 and x_2 order y_1 and y_2 differently (e. g. $y_1 \leq_{x_1} y_2$ and $y_1 \geq_{x_2} y_2$), then the algorithm samples a third predator x_3 uniformly at random. If x_2 and x_3 order y_1 and y_2 in the same way (e. g. $y_1 \geq_{x_2} y_2$ and $y_1 \geq_{x_3} y_2$), then (probabilistically) the predator x_1 orders the prey more uniquely than x_2, and the algorithm selects x_1. (See illustration in Fig. 1). Otherwise, if x_1 and x_3 order y_1 and y_2 in the same way (e. g. $y_1 \leq_{x_1} y_2$ and $y_1 \leq_{x_3} y_2$), then the predator x_2 (probabilistically) orders the prey in a more unique way than x_1 and x_2 is selected. If both x_1 and x_2 order y_1 and y_2 identically (e. g. $y_1 \leq_{x_1} y_2$ and $y_1 \leq_{x_2} y_2$), then the algorithm selects the best worst-case solution, that is, the solution with the highest payoff value for its worst interaction.

Problem Definition. Maximin optimisation is a decision-making approach that seeks to find a candidate solution that maximise the possible payoff, assuming that the adversary takes the least favourable action for that solution. Formally, given a function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ representing the payoff, maximin optimisation involves finding a solution $x \in \mathcal{X}$ such that $\max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y)$ is achieved.

We consider the function $\textsc{Bilinear}_{\alpha,\beta} : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$ [18] defined for arbitrary, not necessarily constant, $\alpha, \beta \in (0,1)$ as $\textsc{Bilinear}_{\alpha,\beta}(x, y) := \|y\|(\|x\| - \beta n) - \alpha n\|x\|$. The maximin-optima of $\textsc{Bilinear}_{\alpha,\beta}$ are all pairs (x, y) with $\|x\| = \beta n$ and $\|y\| = \alpha n$.

We focus on $\textsc{Bilinear}_{\alpha,\beta}$ because it has been shown to be a challenging class of problems to solve due to its intransitive behaviour (rock-paper-scissors-like interaction). Most studied algorithms do not have a clear signal towards/away the optimal solutions during most of the optimisation, leaving the algorithms susceptible to dynamics akin to random walks, cycling around the target solutions [12,13].

During our analysis, we partition the search space into four *quadrants* (Fig. 1). We say that a pair of search points (x, y) is in: the first quadrant if $\|x\| < \beta n \wedge \|y\| \geq \alpha n$, the second quadrant if $\|x\| \geq \beta n \wedge \|y\| > \alpha n$, the third quadrant if $\|x\| > \beta n \wedge \|y\| \leq \alpha n$, and the fourth quadrant if $\|x\| \leq \beta n \wedge \|y\| < \alpha n$.

3.1 Level-Based Theorem

In this work we use the level-based theorem for coevolutionary processes from [18]. In addition we also use a slight modification shown in the appendix (Theorem 4). In this modified version, rather than assuming that the initial levels cover the whole search space, we allow for arbitrary initial levels, as long as it can be ensured that a sufficiently large proportion of the populations are initialised within these initial levels. Furthermore, the runtime is stated as a tail bound rather than an upper bound on the expected runtime.

We now divide the search in a suitable sequence $(A_j \times B_j)_{j \in m}$ of subsets of $\mathcal{X} \times \mathcal{Y}$ (levels) where $A_m \times B_m$ is the target set. Given our objective is to ensure the general applicability of our results across a wide range of algorithms (i.e. Algorithm 1 with any appropriate archive and Algorithm 2), we give a general

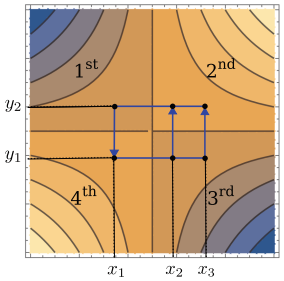


Fig. 1. Contour of $\textsc{Bilinear}_{\alpha,\beta}$ ($\alpha = \beta = 1/2$) with the quadrants enumerated and illustration of diversity mechanism in RankDivCoEA, where the blue arrows indicate increasing g-values. Given a uniformly selected predator x_3 and uniformly selected prey y_1 and y_2, predator x_1 is preferred over predator x_2 because it has a more unique ordering of y_1 and y_2. (Color figure online)

definition of a level that partitions the search space in three customisable regions for each population. Then, when analysing a particular algorithm, we can define a particular sequence of levels with this definition.

Following [18] and as illustrated in Fig. 2 (a), for any $r_1 \leq r_2$ and any $s_2 \leq s_1$, we partition $\mathcal{X}$ and $\mathcal{Y}$ into three sets each,

$$\begin{aligned} R_0(r_1) &:= \{x \in \mathcal{X} \mid 0 \leq \|x\| < r_1\}, & S_0(s_1) &:= \{y \in \mathcal{Y} \mid s_1 < \|y\| \leq n\}, \\ R_1(r_1, r_2) &:= \{x \in \mathcal{X} \mid r_1 \leq \|x\| \leq r_2\}, & S_1(s_2, s_1) &:= \{y \in \mathcal{Y} \mid s_2 \leq \|y\| \leq s_1\}, \\ R_2(r_2) &:= \{x \in \mathcal{X} \mid r_2 < \|x\| \leq n\}, & S_2(s_2) &:= \{y \in \mathcal{Y} \mid 0 \leq \|y\| < s_2\}. \end{aligned}$$

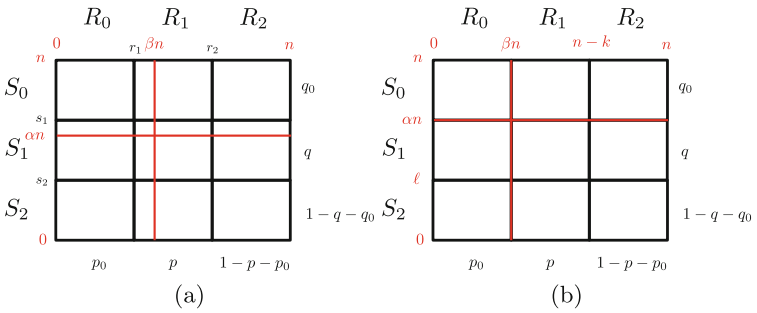


Fig. 2. Partitioning of search space $\mathcal{X} \times \mathcal{Y}$ of Bilinear. (Adapted from [18])

For the sake of clarity, when the parameters r_1, r_2, s_1 and s_2 are clear from context, we will just denote these sets as R_0, R_1, R_2, S_0, S_1 and S_2.

In Sect. 6, we redefine slightly the partition of the search space, as illustrated in Fig. 2 (b). For any $k \in [0, (1-\beta)n]$ and $\ell \in [0, \alpha n)$, we partition $\mathcal{X}$ and $\mathcal{Y}$ into

three sets each,

$$R_0 := \{x \in \mathcal{X} \mid 0 \leq \|x\| < \beta n\}, \qquad S_0 := \{y \in \mathcal{Y} \mid \alpha n \leq \|y\| \leq n\},$$
$$R_1(k) := \{x \in \mathcal{X} \mid \beta n \leq \|x\| < n - k\}, \quad S_1(\ell) := \{y \in \mathcal{Y} \mid \ell \leq \|y\| < \alpha n\},$$
$$R_2(k) := \{x \in \mathcal{X} \mid n - k \leq \|x\| \leq n\}, \quad S_2(\ell) := \{y \in \mathcal{Y} \mid 0 \leq \|y\| < \ell\}.$$

For ease of notation, when the k and ℓ are clear from the context, we will simply refer to these sets as R_0, R_1, R_2, S_0, S_1, and S_2.

We now proceed to define the probability of sampling an individual inside each of these regions within the search space from the current populations.

$$p_0 = \Pr_{x \sim \text{Unif}(P)}[x \in R_0], \qquad q_0 = \Pr_{y \sim \text{Unif}(Q)}[y \in S_0],$$
$$p = \Pr_{x \sim \text{Unif}(P)}[x \in R_1], \qquad q = \Pr_{y \sim \text{Unif}(Q)}[y \in S_1].$$

Furthermore, for the random parents x' and y' selected in lines 14 and 19 of Algorithm 1 and lines 13 and 20 of Algorithm 2, for all $C \subseteq \mathcal{X} \times \mathcal{Y}$, let $p_{\text{sel}}(C) := \Pr((x', y') \in C)$. In addition, we define the probability of sampling an individual with more, less and exactly the same number of ones as the optimal solution from the current archives.

$$v_1 = \Pr_{x \sim \text{Unif}(V)}[\|x\| < \beta n], \qquad w_1 = \Pr_{y \sim \text{Unif}(W)}[\|y\| < \alpha n],$$
$$v_2 = \Pr_{x \sim \text{Unif}(V)}[\|x\| = \beta n], \qquad w_2 = \Pr_{y \sim \text{Unif}(W)}[\|y\| = \alpha n],$$
$$v_3 = \Pr_{x \sim \text{Unif}(V)}[\|x\| > \beta n], \qquad w_3 = \Pr_{y \sim \text{Unif}(W)}[\|y\| > \alpha n].$$

4 What is a Good Evaluation Environment?

In this section we explore what are the characteristics that the evaluation environment (archive population) used by Algorithm 1 needs to guarantee conditions (G2a) and (G2b) in the level-based theorem for coevolution [18] when using a tournament of size $k = 2$ and a competition size $\ell = 2$ on $\textsc{Bilinear}_{\alpha,\beta}$. In the context of $\textsc{Bilinear}_{\alpha,\beta}$ (and likely many other problems), this is the most challenging scenario. This is because the algorithm must simultaneously select a *good* individual from the population to participate in the tournament and also a diverse set of competitors to ensure an accurate ranking, while sampling only two times from the population and two times from the archive. If the tournament size or the competition size is increased we conjecture that the restrictions on the archive would be lessened.

To begin this section we assume that a *good* archive would make the individuals in the current populations P and Q approach the maximin-optima. Using the partitions from Sect. 3.1 we can define the levels that we will use in this section. $A(r_1, r_2) := R_1(r_1, r_2)$, $B(s_2, s_1) := S_1(s_2, s_1)$. In this section, we always use $A_1 := A(0, n) = \mathcal{X}$ and $B_1 := B(0, n) = \mathcal{Y}$.

In Lemma 1 we show that for any level with $\beta n - r_1 = r_2 - \beta n$ and $\alpha n - s_2 = s_1 - \alpha n$ (Fig. 5 (a)) the archive populations V and W need to have a certain ranking diversity. More specifically for archive V (W) there needs to be at least $1 - \frac{1}{\sqrt{2}} \approx 0.29$ fraction of individuals with more than βn (αn) 1-bits ranking $y \in Q$ ($x \in P$) with respect to the number of 0-bits, and $1 - \frac{1}{\sqrt{2}}$ fraction of individuals with less than βn (αn) 1-bits ranking $y \in Q$ ($x \in P$) with respect to the number of 1-bits.

We note that the values needed to meet the conditions of Lemma 1 for $r_{(0)}$ are relatively high. This is in part because the tournament size is small but also because the proof uses approximations and some pessimistic assumptions to ease the computations.

Lemma 1. *Let $r_1 \leq \beta n$, $\beta n \leq r_2$, $s_2 \leq \alpha n$, $\alpha n \leq s_1$ with $\beta n - r_1 = r_2 - \beta n$ and $\alpha n - s_2 = s_1 - \alpha n$. Let $\varepsilon > 0$ be a constant. Let x and y be the offspring created in Lines 15 and 20 of Algorithm 1 with $k = \ell = 2$, and $r_{(0)}$ be the probability of not flipping a bit during mutation. If $v_1, v_3, w_1, w_3 \geq 1 - \frac{1}{\sqrt{2}} + \varepsilon$ and there exist constants $\delta, \delta' > 0$ such that $\frac{1+\delta}{1+(1-\delta')\left(1+\sqrt{2}\right)\varepsilon} \leq r_{(0)}^2 \leq 1$, then, for all $\gamma \in (0, \delta'/2]$ any population with $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$ with $|(P \times Q) \cap (A(r_1, r_2) \times B(s_2, s_1))| \geq \gamma\lambda^2$ guarantees that*

$$\Pr\left[x \in A(r_1, r_2)\right] \Pr\left[y \in B(s_2, s_1))\right] \geq (1 + \delta)\gamma.$$

It is clear that if the algorithm does not know beforehand that the optimisation problem is $\textsc{Bilinear}_{\alpha,\beta}$, it is unlikely that its archive populations are initialised in such a way that the conditions of Lemma 1 are met. Nonetheless, with the insights of Lemma 1 we can design algorithms that create and maintain such diversity of rankings (Fig. 3).

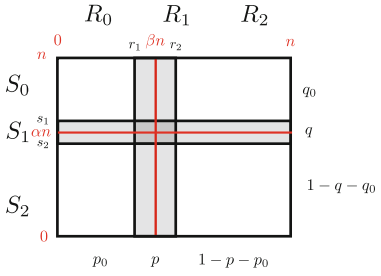


Fig. 3. Levels for Lemma 1 on $\textsc{Bilinear}_{\alpha,\beta}$.

5 An Example Archive with Efficient Runtime on BILINEAR

In the previous section we learnt that if the competitors in the archives rank solutions in the current populations P and Q differently then the algorithm tends towards the maximin-optima.

Based on this notion we propose an archive update scheme (Algorithm 3) that we call Diverse Ranking Archives Update Scheme (DRAUS). This update scheme is used in line 10 of Algorithm 1. It uses the concept of archive domination (Definition 2) to add to the archive any solution in the current population that ranks solutions differently than other solutions already in the archive.

Definition 2 (Archive Domination). *Given a function $g\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, a population $Q \in \mathcal{Y}^\lambda$ and two archives $V_1, V_2 \subseteq \mathcal{X}$, we say that V_1 dominates V_2 with respect to Q and g denoted $V_1 \succeq_Q V_2$ if and only if:*

1. *for all $x_2 \in V_2$ and all $(y_1, y_2) \in Q \times Q$ where $g(x_2, y_1) > g(x_2, y_2)$ there exist $x_1 \in V_1$ st. $g(x_1, y_1) > g(x_1, y_2)$ and*
2. *for all $x_2 \in V_2$ and all $(y_1, y_2) \in Q \times Q$ where $g(x_2, y_1) = g(x_2, y_2)$ there exist $x_1 \in V_1$ st. $g(x_1, y_1) = g(x_1, y_2)$.*

Analogously, $W_1 \succeq_P W_2$ denotes that W_1 dominates W_2 with respect to P and g.

Algorithm 3: Diverse Ranking Archives Update Scheme (DRAUS)

```
1 Require: Search spaces 𝒳 and 𝒴.
2 Require: Payoff function g : 𝒳 × 𝒴 → ℝ.
3 Require: Current archives V ⊆ 𝒳, W ⊆ 𝒴.
4 Require: Current populations P ∈ 𝒳^λ, Q ∈ 𝒴^λ
5 for i ∈ [λ] do
6     if ¬(V ⪰_Q {P(i)}) then V := V ∪ {P(i)} ;
7     if ¬(W ⪰_P {Q(i)}) then W := W ∪ {Q(i)} ;
8 return V, W
```

To check whether $V \succeq_Q \{x\}$ for a solution $x \in P$ takes at most $(|V|+1)|Q| = (|V|+1)\lambda$ evaluations. If the results of the evaluations between the archive V and the population Q are stored, then to check $V \succeq_Q \{x'\}$ for a new solution $x' \in P$ only requires λ extra evaluations. In total this results in at most $(|V|+\lambda)\lambda$ evaluations to update the archive V. The same argument can be applied to the archive W. This shows the bounds in Lemma 2.

Lemma 2. *Let $|V|$ and $|W|$ be the size of the archives. Then, Algorithm 3 uses at most $(|V|+\lambda)\lambda + (|W|+\lambda)\lambda$ evaluations to update the archive.*

On $\text{BILINEAR}_{\alpha,\beta}$ both the predator and prey individuals can only be ranked in three different ways each. Here we explain the three ranks that the predators induce in the prey. All predators x with $\|x\| > \beta n$ order all $y \in \mathcal{Y}$ by their number of 1-bits in descending order, that is, the higher the number of 1-bits in y the smaller their payoff when compared against x with $\|x\| > \beta n$. In turn all

predators x with $\|x\| < \beta n$ order all $y \in \mathcal{Y}$ by their number of 1-bits in ascending order. Finally all predators x with $\|x\| = \beta n$ give the same payoff to all $y \in \mathcal{Y}$.

Therefore the maximum size of both the predator and prey archives using DRAUS (Algorithm 3) is three. Thanks to this, the maximum number of evaluations needed to update the archive on $\textsc{Bilinear}_{\alpha,\beta}$ is in the worst case $O(\lambda^2)$. We remark that this might be different on other problems and it could grow exponentially on the problem size n. Therefore, we recommend to account for that if it is used on other problems.

Now, we show that ATS-CoEA (Algorithm 1) using DRAUS (Algorithm 3) builds a diverse archive and optimise $\textsc{Bilinear}_{\alpha,\beta}$ efficiently for $\alpha, \beta \in (0, 1)$.

Theorem 1. *Let $\alpha, \beta \in (0, 1)$. Consider Algorithm 1 using Algorithm 3 as archive update scheme on $\textsc{Bilinear}_{\alpha,\beta}$. Define $\mathrm{OPT} := \{(x, y) \in (\mathcal{X} \times \mathcal{Y}) \mid \|x\| = \beta n \wedge \|y\| = \alpha n\}$ and $T := \min\{\lambda^2 t \mid P_t \times Q_t \cap \mathrm{OPT} \neq \emptyset\}$. Then if there are constants $\delta, \delta' > 0$ such that the probability $r_{(0)}$ of the mutation operator is at least*

$$\max\left\{\frac{8(1+\delta)}{14(1-\delta')}, \sqrt{\frac{6(1+\delta)}{6+(1-\delta')\left(2-\sqrt{2}\right)}}\right\},$$

$r_{(1)} > 0$ is constant and for a sufficiently large constant c, $c \log n \leq \lambda \in \mathrm{poly}(n)$ then, it holds that $\mathrm{E}[T] = O(\lambda^4 n)$.

The main idea of the proof is that if the algorithm does not have an individual for each possible ranking in both archives, the archives guide the search towards a region of the search space where that ranking exist. After finding all individuals that rank their opponents differently, then the archive meets the conditions for Lemma 1 and the algorithm has a clear signal towards the optimum.

6 Enforcing Diversity Without an Archive

In this section we analyse RankDivCoEA on $\textsc{Bilinear}_{\alpha,\beta}$ with $\alpha n, \beta n \notin \mathbb{Z}$. This algorithm maintains populations that rank their opponents in distinct ways without the overhead of maintaining an archive.

6.1 Balancing of Populations Across βn and αn

We will show that the diversity mechanism will eventually ensure that the number of predators in R_0 converge to approximately $\lambda/2$ (i.e., half the predators). The reason is that predators in R_0 order prey differently than predators outside R_0. If the predators in R_0 are in a minority, the diversity mechanism will prefer them over predators outside R_0, and *vice versa*. With overwhelmingly high probability, not much more than half of the predators are R_0 (Lemma 3 with $A_0 = R_0$). Furthermore, if a constant, but less than half, fraction of the predators are in R_0, then the number of such predators will increase with overwhelmingly high probability (Lemma 4). Due to symmetry of the problem and the algorithm, analogous statements can be proved about the prey population with respect to the region S_0.

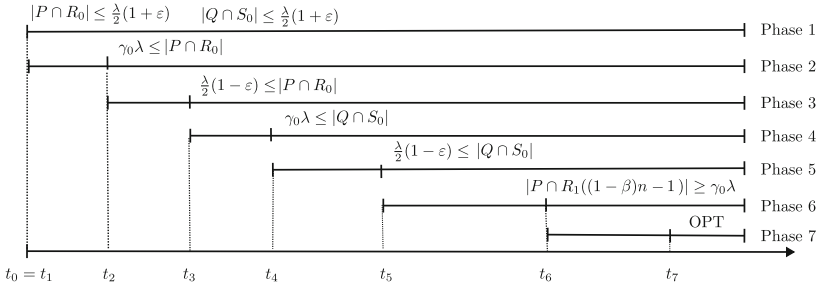


Fig. 4. Overview of an era.

Lemma 3. *Consider Algorithm 2 with x_1, x_2, y_1, y_2 from line 8 and x' from line 13. Let $\varepsilon \in (0, 1/2)$ be any constant. Let $A \subset \mathcal{X}$ be any subset such that $\forall y_1, y_2 \in \mathcal{Y}, \forall x_1 \in A, \forall x_2 \in \mathcal{X} \setminus A$, $g(x_1, y_1) \geq g(x_1, y_2)$ if and only if $g(x_2, y_1) \leq g(x_2, y_2)$, and for all $x' \in \mathcal{X} \setminus A$, $\Pr_{x \sim mut_x(x')}(x \in A) \leq \varepsilon/6$. For all $t \geq 0$, if $|P_t \cap A| \leq \frac{\lambda}{2}(1+\varepsilon)$, then $\Pr\left[|P_{t+1} \cap A| > \frac{\lambda}{2}(1+\varepsilon)\right] = e^{-\Omega(\lambda)}$.*

Lemma 4. *Consider Algorithm 2 with x_1, x_2, y_1, y_2 from line 8 and x' from line 13. Let $\varepsilon \in (0, 1/2)$ be any constant. Let $A \subset \mathcal{X}$ be any subset such that $\forall y_1, y_2 \in \mathcal{Y}, \forall x_1 \in A, \forall x_2 \in \mathcal{X} \setminus A$, $g(x_1, y_1) \geq g(x_1, y_2)$ if and only if $g(x_2, y_1) \leq g(x_2, y_2)$, and for all $x' \in A$, $\Pr_{x \sim mut_x(x')}(x \in A) \geq 1 - \frac{\varepsilon}{4}$. For all $t \in \mathbb{N}$, define $X_t := |P_t \cap A|$. Then, for all $t \geq 0$, $\Pr\left[X_{t+1} \geq (1 + \frac{\varepsilon}{16}) \min\{X_t, \frac{\lambda}{2}(1-\varepsilon)\} \mid X_t\right] \geq 1 - e^{-\Omega(X_t)}$.*

6.2 Runtime Analysis Sketch

We will divide the run of the algorithm into *eras*, where each era consists of seven overlapping *phases* as illustrated in Fig. 4. Phase $i \in [7]$ starts at deterministic generation number t_{i-1} and lasts until the last generation number t_7. Each phase $i \in [7]$ has a "warm-up" sub-phase from generation t_{i-1}, lasting for $\tau_i := t_i - t_{i-1}$ generations. We let $\tau := \sum_{i=1}^{7} \tau_i = t_7 - t_0$ denote the duration of one era. Each phase i is associated with a predicate $\mathcal{E}_i$ defined on the set of possible populations $\mathcal{X}^\lambda \times \mathcal{Y}^\lambda$. Phase i is called *successful* if for every generation $t \in [t_i, t_7]$, predicate $\mathcal{E}_i(P_t, Q_t)$ holds. The final event $\mathcal{E}_7(P, Q)$ corresponds to the populations P and Q containing optimal solutions. To compute the expected time until the algorithm finds the optimum, we compute the expected number of eras until all phases within an era are successful.

It will become apparent from the analysis that once the populations are approximately equally divided in the four quadrants of the search space, the populations will steadily evolve towards the optimum. The first phases capture the fact that the algorithm will eventually reach a configuration where the predators are almost evenly divided above and below βn, and the prey are almost evenly

divided above and below αn. More precisely,

$$\frac{1-\varepsilon}{2} \leq \frac{|P \cap R_0|}{\lambda} \leq \frac{1+\varepsilon}{2} \quad \text{and} \quad \frac{1-\varepsilon}{2} \leq \frac{|Q \cap S_0|}{\lambda} \leq \frac{1+\varepsilon}{2}. \tag{1}$$

Unless the populations are already in this configuration in generation t_0, there must exist one quadrant which contains the majority of the predators and prey. Without loss of generality, we will assume that the era starts in generation t_0 with a large fraction of the predator-prey pairs in the third quadrant. More precisely, we will make the following assumption:

Assumption 1. *The populations at the start of the era satisfy*

$$\frac{|P_{t_0} \cap R_0|}{\lambda} < \frac{1+\varepsilon}{2} \quad \textit{and} \quad \frac{|Q_{t_0} \cap S_0|}{\lambda} < \frac{1+\varepsilon}{2}. \tag{2}$$

Clearly, other initial configurations are possible. However, due to symmetry of the problem and the algorithm, the analysis starting from other configurations is the same. Given the assumption about the initial populations, the phases with corresponding predicates are given in Definition 3, and illustrated in Fig. 4.

Definition 3. *Let $\varepsilon, \gamma_0, \delta$ be some constants satisfying $\delta \in (0, 1/5)$, $0 < \gamma_0 \leq (1+3\varepsilon)/4$ and $\gamma_0 + \varepsilon \leq \frac{2}{3}(1-\delta)$. Define the predicates*

$$\mathcal{E}_1(P,Q) := |Q \cap S_0| \leq \frac{\lambda}{2}(1+\varepsilon) \wedge |P \cap R_0| \leq \frac{\lambda}{2}(1+\varepsilon) \tag{3}$$

$$\mathcal{E}_2(P,Q) := |P \cap R_0| \geq \gamma_0 \lambda \tag{4}$$

$$\mathcal{E}_3(P,Q) := |P \cap R_0| \geq \frac{\lambda}{2}(1-\varepsilon) \tag{5}$$

$$\mathcal{E}_4(P,Q) := |Q \cap S_0| \geq \gamma_0 \lambda \tag{6}$$

$$\mathcal{E}_5(P,Q) := |Q \cap S_0| \geq \frac{\lambda}{2}(1-\varepsilon) \tag{7}$$

$$\mathcal{E}_6(P,Q) := |P \cap R_1((1-\beta)n - 1)| \geq \gamma_0 \lambda \tag{8}$$

$$\mathcal{E}_7(P,Q) := \exists x \in P, \exists y \in Q, \|x\| = \beta n \wedge \|y\| = \alpha n. \tag{9}$$

In addition, we assume the following for the mutation operator.

Assumption 2. *The probabilities $r_{(1)}$ and $r_{(0)}$ of the mutation operator satisfy $r_{(1)} > 0$ is constant, $r_{(0)}^2 \geq 1 - \delta/2$ and $r_{(0)} \geq 1 - \varepsilon/6$, with δ and ε given in Definition 3, and the population size is $\lambda \geq c \log(n)$ for a sufficiently large constant c.*

Following the proof sketch above we can show the main result of this section. We note that preliminary experiments show that the restriction $\alpha n, \beta n \notin \mathbb{Z}$ in Theorem 2 is not necessary, but it simplifies the already lengthy proofs.

Theorem 2. *For all $\alpha n, \beta n \notin \mathbb{Z}$, Algorithm 2 with parameter satisfying Assumption 2 has expected runtime $O(n\lambda^3 - n\lambda \ln(\alpha\beta(1-\beta)(1-\alpha)))$ on* $\textsc{Bilinear}_{\alpha,\beta}$.

7 Conclusions

We have shown that for the intransitive problem class $\textsc{Bilinear}_{\alpha,\beta}$ it is beneficial to maintain a high *diversity of rankings* in the evaluation environment, that is, a set of individuals used for evaluation which are diverse in how they rank opponents. We proposed two algorithms that ensure high diversity of rankings via archive populations (ATS-CoEA with DRAUS) and an intrinsic ranking diversity mechanism (RankDivCoEA). We rigorously showed that both approaches optimise $\textsc{Bilinear}_{\alpha,\beta}$ in $O(\lambda^4 n)$ and $O(\lambda^3 n)$ expected evaluations respectively.

The $\textsc{Bilinear}_{\alpha,\beta}$ problem class only has a small number of possible rankings for each population. We showed that the algorithms proposed here work well for this case, but it remains an open problem whether these mechanisms are effective on intransitive problems with a large number of possible rankings.

Acknowledgments. This research was supported by a Turing AI Fellowship (EPSRC grant ref EP/V025562/1).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Angeline, P.J., Pollack, J.B.: Competitive environments evolve better solutions for complex tasks. In: Proceedings of the International Conference on Genetic Algorithms, pp. 264–270. Morgan Kaufmann Publishers Inc., San Francisco (1993)
2. Arcuri, A., Yao, X.: A novel co-evolutionary approach to automatic software bug fixing. In: 2008 IEEE Congress on Evolutionary Computation, pp. 162–168 (2008)
3. Czarnecki, W.M., et al.: Real world games look like spinning tops. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 17443–17454. Curran Associates, Inc. (2020)
4. Jong, E.D.: The incremental pareto-coevolution archive. In: Deb, K. (ed.) GECCO 2004. LNCS, vol. 3102, pp. 525–536. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24854-5_55
5. De Jong, E.D.: Towards a bounded pareto-coevolution archive. In: Proceedings of the 2004 Congress on Evolutionary Computation, vol. 2, pp. 2341–2348 (2004)
6. De Jong, E.D., Bucci, A.: DECA: dimension extracting coevolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2006, pp. 313–320. Association for Computing Machinery, New York (2006)
7. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2016, pp. 1123–1130. Association for Computing Machinery, New York (2016)
8. Ficici, S.G.: Solution concepts in coevolutionary algorithms. Ph.D. thesis, Department of Computer Science, Brandeis University, Waltham, MA (2004)

9. Ficici, S.G., Pollack, J.B.: A game-theoretic memory mechanism for coevolution. 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. 286–297. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45105-6_35
10. Flores, D., Hemberg, E., Toutouh, J., O'Reily, U.M.: Coevolutionary generative adversarial networks for medical image augumentation at scale. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 367–376. Association for Computing Machinery, New York (2022)
11. Hemberg, E., Toutouh, J., Al-Dujaili, A., Schmiedlechner, T., O'Reilly, U.M.: Spatial coevolution for generative adversarial network training. ACM Trans. Evol. Learn. Optim. **1**(2), 1–28 (2021)
12. Hevia Fajardo, M.A., Lehre, P.K.: How fitness aggregation methods affect the performance of competitive CoEAs on bilinear problems. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2023, pp. 1593–1601. Association for Computing Machinery, New York (2023)
13. Hevia Fajardo, M.A., Lehre, P.K., Lin, S.: Runtime analysis of a co-evolutionary algorithm: overcoming negative drift in maximin-optimisation. In: Proceedings of the ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA 2023, p. p73–83. Association for Computing Machinery, New York (2023)
14. Hillis, W.: Co-evolving parasites improve simulated evolution as an optimization procedure. Physica D **42**(1–3), 228–234 (1990)
15. Jansen, T., Wiegand, R.P.: The cooperative coevolutionary (1+1) EA. Evol. Comput. **12**(4), 405–434 (2004)
16. Jaśkowski, W., Krawiec, K.: Coordinate system archive for coevolution. In: IEEE Congress on Evolutionary Computation, pp. 1–10 (2010)
17. Krawiec, K., Heywood, M.: Solving complex problems with coevolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 832–858. Association for Computing Machinery, New York (2020)
18. Lehre, P.K.: Runtime analysis of competitive co-evolutionary algorithms for maximin optimisation of a bilinear function. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 1408–1416. Association for Computing Machinery, New York (2022)
19. Lehre, P.K., Lin, S.: Is CC-(1+1) EA more efficient than (1+1) EA on separable and inseparable problems? In: 2023 IEEE Congress on Evolutionary Computation (CEC), pp. 1–9 (2023)
20. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Algorithmica **64**(4), 623–642 (2012)
21. Luke, S., Wiegand, R.P.: When coevolutionary algorithms exhibit evolutionary dynamics. In: Genetic and Evolutionary Computation Conference Workshop Program, pp. 236–241 (2002)
22. Miconi, T.: Why coevolution doesn't "Work": superiority and progress in coevolution. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 49–60. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01181-8_5
23. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
24. Nolfi, S., Floreano, D.: Coevolving predator and prey robots: Do "Arms Races" arise in artificial evolution? Artif. Life **4**(4), 311–335 (1998)

25. Oliehoek, F.A., De Jong, E.D., Vlassis, N.: The parallel Nash Memory for asymmetric games. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2006, pp. 337–344. Association for Computing Machinery, New York (2006)
26. Popovici, E., Bucci, A., Wiegand, R.P., De Jong, E.D.: Coevolutionary principles. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) Handbook of Natural Computing, pp. 987–1033. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-92910-9_31
27. Rosin, C.D., Belew, R.K.: Methods for competitive co-evolution: finding opponents worth beating. In: Proceedings of the International Conference on Genetic Algorithms, pp. 373–381. Morgan Kaufmann Publishers Inc., San Francisco (1995)
28. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. Evol. Comput. **5**(1), 1–29 (1997)
29. Sims, K.: Evolving 3D morphology and behavior by competition. Artif. Life **1**(4), 353–372 (1994)
30. Watson, R.A., Pollack, J.B.: Coevolutionary dynamics in a minimal substrate. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2001, pp. 702–709. Morgan Kaufmann Publishers Inc., San Francisco (2001)
31. Yang, L., Huang, H., Yang, X.: An efficient pareto-coevolution archive. In: Proceedings of the Third International Conference on Natural Computation, ICNC 2007, vol. 04, pp. 484–488. IEEE Computer Society (2007)
32. Yang, L., Huang, H., Yang, X.: A simple coevolution archive based on bidirectional dimension extraction. In: Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence, AICI 2009, vol. 01, pp. 596–600. IEEE Computer Society (2009)

On the Equivalence Between Stochastic Tournament and Power-Law Ranking Selection and How to Implement Them Efficiently

Duc-Cuong Dang(✉), Andre Opris, and Dirk Sudholt

University of Passau, Passau, Germany
duccuong.dang@uni-passau.de

Abstract. Tournament selection is a popular parent selection mechanism in evolutionary algorithms. Bian and Qian (PPSN 2022) proved that choosing the tournament size uniformly at random, called *stochastic tournament selection*, in combination with crossover significantly improves the performance of NSGA-II on some benchmark functions. We show that this selection mechanism is asymptotically equivalent to the *power-law ranking selection* proposed in Covantes Osuna et al. (*Theor. Comput. Sci.* 832, 2020) with the exponent of 2. Thus asymptotic runtime bounds proven for one operator also hold when one operator is replaced with the other.

We also investigate how to implement these operators efficiently for NSGA-II on the problems considered in the previous papers. We propose to implement the stochastic tournament with a pre-computed selection distribution to save on random numbers. Experiments on high dimensional problems demonstrate the superiority of this method compared to the standard implementation. Overall, the power-law ranking selection is the most efficient selection mechanism for the studied problems. Remarkably, we also find that the way ties are broken between equally fit solutions can make the difference between the best and the worst approach, especially when crossover is involved.

Keywords: Selection operators · tournament selection · power-law ranking · multi-objective optimisation · runtime analysis · algorithm engineering

1 Introduction

Selection together with variation are the driving forces of evolution. Selection favours more adapted individuals thus allows their useful traits to pass on to future generations, so without selection adaptation is not possible. Early research in evolutionary computation has considered various selection mechanisms such as fitness proportionate selection, tournament selection, ranking selections, and their properties and effects on the population per generation [4,26].

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 230–245, 2024.
https://doi.org/10.1007/978-3-031-70071-2_15

As theory research has gained momentum in evolutionary computation over the years [14,30], selection mechanisms can now be studied under the lens of rigorous runtime analysis, specifically their impacts to the optimisation times have been proven for different settings. The inefficiency of fitness proportionate selection (also called roulette wheel selection), a popular parent selection mechanism for evolutionary algorithms [19], was investigated in [27–29]. Specifically, Oliveto and Witt [29] showed that with high probability the Simple Genetic Algorithm with a population size below $n^{1/4}$, using this selection requires exponential runtime to optimise the simple OneMax function, even when crossover is enabled. Lehre [24,25] provided general tools to analyse the negative and positive effects of selection mechanisms in non-elitist populations that only use mutations as variation operators, and also proved a similar negative result for fitness proportionate selection, but for population sizes above n^3. Other selection mechanisms were also considered in [25], such as tournament, truncation selection (also known as comma selection), and linear ranking to show that the right balance between selection and mutation is the key to the efficient optimisation of standard benchmark functions. By proving a lower bound on the expected running of comma selection on $\textsc{Jump}_k$, Doerr [13] argued that there is no apparent benefit of comma selection over plus selection (i.e. elitist algorithms) in escaping local optima. Dang et al. [8] pointed out the issue of the comma selection is due to the linearity of its cumulative selection distribution, then provided a class of problems with local optima in which non-elitist algorithms with appropriate selection mechanisms excel while elitist algorithms struggle. This rigorous research also led to the introduction of new effective selection operators, such as the power-law ranking selection for non-elitist populations by [9,10], inverse ranking selection for steady-state algorithms [6], or stochastic survival selection in [3] for multi-objective optimisation.

We are interested the following two results on the speed-up of multi-objective evolutionary algorithms (EMO) using novel parent selection mechanisms.

Covantes Osuna et al. [7] proposed to apply diversity metrics in the parent selection. The idea is to focus on individuals located in poorly explored areas of the search space to increase the chance of creating new non-dominated solutions. Their approach ranks possible parents (non-dominated solutions in (G)SEMO [18,23]) according to their crowding distance or hypervolume contribution and then picks individuals from a fixed distribution that favours better ranks. One such distribution is the power-law distribution with exponent 2 that we call *power-law ranking selection*, in which the probability of selecting an individual at rank i in the population is proportional to $1/i^\beta$ for a parameter $\beta > 1$ (this is a different selection compared to the one from [9,10], because the latter imposes a power-law on the cumulative selection distribution). The paper showed speedups by a factor of order n for (G)SEMO on OneMinMax (OMM) and for SEMO on LeadingOnesTrailingZeroes (LOTZ) compared to uniform parent selection as the Pareto front is explored a lot more effectively by focusing on extreme search points and others with high diversity score.

Bian and Qian [2] analysed the performance of NSGA-II using crossover on LOTZ. Tournament selection commonly uses a fixed tournament size k and returns the best individuals from k population members chosen uniformly at random. Bian and Qian [2] (and its extended version [1]) considered a modification to tournament selection where the tournament size is chosen uniformly at random between 1 and the population size. They proved that this *stochastic tournament selection* allows a significant improvement of the asymptotic expected runtime bound of the NSGA-II algorithm [11] with crossover on LOTZ [2] and other benchmark functions [1]. The reason is that parent selection focuses on the extreme points. This implies that the Pareto-optima 0^n and 1^n are discovered quickly, and then one-point crossover is able to create all other Pareto-optimal solutions $\{1^i0^{n-i} \mid 1 \leq i \leq n-1\}$ by crossing 1^n with 0^n.

There are several similarities between these two papers. The runtime bound of $O(n^2)$ for NSGA-II on LOTZ in [2] matches that for SEMO on LOTZ in [7]. Both independently developed the idea of focusing on solutions with a high diversity score through defining a new selection operator, albeit the way of exploring the Pareto front is different ([2] relies on one-point crossover of 1^n and 0^n and [7] relies solely on mutation).

1.1 Our Contributions

We prove that the selection distributions of power-law ranking selection with exponent 2 and stochastic tournament selection are asymptotically equivalent: both sample the i-th ranked search point with a probability proportional to $1/i^2$. This implies that all bounds on the expected runtime proven for algorithms relying on one of these mechanisms automatically hold for the same algorithms if the other selection mechanism is used instead. In other words, the results on (G)SEMO in [7] also hold for stochastic tournament selection, and those for NSGA-II in [2] also hold for power-law ranking selection. Furthermore, we point out that all the results on the speed-up on synthetic benchmark functions OMM, LOTZ, CountingOnesCountingZeroes (COCZ) by NSGA-II gained by using stochastic tournament selection in the latter paper also hold for the use of all power-law ranking selections with exponent $\beta > 1$.

The rigorous runtime analyses results only cover the runtime in terms of the asymptotic number of fitness evaluations and the leading constant is not specified. In practice, the leading constant in the running time can also be important. Additional factors such as the cost of producing random numbers and other basic operations for selection or mutation operators can impact the computational time to produce satisfactory solutions (see [21] for a discussion of these issues and [22] for a recent analysis of the cost of randomness). We therefore investigate how the two equivalent selection mechanisms can be efficiently implemented and conduct experiments with NSGA-II on the above-mentioned functions to compare the true performance of these selection mechanisms in terms of CPU times.

We propose an implementation of the stochastic tournament which is similar to how the power-law ranking selection is implemented, i. e. by pre-computing

the selection distribution in each generation. The advantage of this approach compared to the naive implementation, i. e. actually comparing random numbers of individuals, is that it saves on random numbers in each generation from the quadratic order of the population (in expectation or with probability at least 1/2) to (surely) linear. Experiments on high dimensional problems confirm the efficiency of this implementation over the traditional one. Overall the power-law ranking selection optimises the studied function using the shortest CPU times, hence it is the most efficient mechanism. This is because numerically (and not asymptotically) the mechanism attributes larger probabilities to select top-ranked individuals, compared to the stochastic tournament.

An additional insight from our study is that the seemingly innocent question of how to break ties between equally good individuals can be of utmost importance since in our scenario it makes the difference between the best and the worst implementations. A naive, arbitrarily fixed ranking gives the worst performance. In contrast, when equally fit elements receive the same probability mass in the selection step, we obtain the best performance. The reason for discrepancy is due to the important role that crossover plays during the optimisation process [2]: it is crucial to make sure that when applying crossover two different extreme solutions, i. e. 0^n and 1^n in the case of LOTZ and OMM, and $1^{n/2}0^{n/2}$ and 1^n in the case of COCZ, that are equally fit, are selected to ensure the creation of successful (Pareto-optimal) offspring.

2 Preliminaries

The set of natural numbers, reals, complex numbers are denoted $\mathbb{N}, \mathbb{R}, \mathbb{C}$ respectively. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. The logarithm function of base 2 is denoted $\log(x)$ for $x \in \mathbb{R}^+$. The Riemann's Zeta function is denoted $\zeta(x)$ for $x \in \mathbb{C}$. The Euler constant is e.

We use standard asymptotic notation with symbols $\mathcal{O}, \Omega, \Theta, o$ [5]. Let $f(i)$ and $g(i)$ be two functions on $[0, +\infty)$. We say that they are asymptotically equal, denoted $f \sim g$, if and only if there exist constants $c_2 \geq c_1 > 0$ and $i_0 \geq 0$ such that $c_1 g(i) \leq f(i) \leq c_2 g(i)$ for all $i \geq i_0$.

Let $P := (x_1, \ldots, x_\mu)$ be a sequence (or population) of μ search points (on any search space) sorted according to some criteria which imposes a total order on P. In a k-tournament selection, k individuals from P are picked uniformly at random with replacement and the one with the smallest index in the sorted population, i. e. the best one, is returned. The advantage of this operator is that it is easy to implement and efficient if k is small as $k-1$ pairwise comparisons using the same criteria are sufficient to determine the best individual to return. When $k = 2$, this selection is referred as binary tournament selection.

NSGA-II [11] is a popular EMO algorithm summarised in Algorithm 1. It uses the so-called $(\mu + \mu)$ elitist survival selection scheme. The algorithm starts from a randomly initialised population of μ solutions. In each generation new μ offspring solutions (the population Q_t) are generated. Then the 2μ solutions of $R_t := P_t \cup Q_t$ compete for survival, so that the next population P_{t+1} again has

Algorithm 1: NSGA-II [11] on $\{0,1\}^n$,

Initialise $P_0 \sim \text{Unif}((\{0,1\}^n)^\mu)$;
for $t := 0$ *to* ∞ **do**
  Initialise $Q_t := \emptyset$;
  for $i = 1$ *to* μ **do**
    Sample p_1, p_2 from P_t using a selection mechanism;
    With probability p_c create x by applying 1-point crossover on p_1, p_2, otherwise create x as a clone of p_1;
    Apply the bitwise mutation with mutation rate $1/n$ on x;
    Update $Q_t := Q_t \cup \{x\}$;
  Set $R_t := P_t \cup Q_t$;
  Partition R_t into layers $F^1_{t+1}, F^2_{t+1}, \dots$ using the non-dominated sorting algorithm [11];
  Compute the critical layer $i^* \geq 1$ such that $\sum_{i=1}^{i^*-1} |F^i_{t+1}| < \mu$ and $\sum_{i=1}^{i^*} |F^i_{t+1}| \geq \mu$;
  Set $Y_t := \bigcup_{i=1}^{i^*-1} F^i_{t+1}$;
  Select a multiset $\tilde{F}^{i^*}_{t+1} \subset F^{i^*}_{t+1}$ of individuals such that $|Y_t \cup \tilde{F}^{i^*}_{t+1}| = \mu$ using the crowding distances;
  Create the next population $P_{t+1} := Y_t \cup \tilde{F}^{i^*}_{t+1}$;

size μ. We use the same setting as [2] for the production of offspring: each offspring is created independently by first picking two parents, also independently, using a selection mechanism. Then, with a probability p_c, one-point crossover is applied to produce the offspring. Otherwise (i. e. with probability $1 - p_c$), the offspring is identical to the first parent. The offspring is then mutated using standard bit mutation with rate $1/n$, i. e. each bit is flipped independently from the others with probability $1/n$. In the survival selection, R_t is partitioned into a sequence of non-dominated layers $(F^1_{t+1}, F^2_{t+1}, \dots)$ by the *non-dominated sorting algorithm* (the readers are referred to [1,2,11] for details). The algorithm follows this sequence to include the first encountered layers into the next generation P_{t+1} until it reaches a critical layer $F^{i^*}_{t+1}$ with $\sum_{i=1}^{i^*-1} |F^i_{t+1}| < \mu$ and $\sum_{i=1}^{i^*} |F^i_{t+1}| \geq \mu$. The remaining $r := \mu - \sum_{i=1}^{i^*-1} |F^i_{t+1}|$ slots of P_{t+1} (if $r > 0$) are then taken by solutions from $F^{i^*}_{t+1}$. The criterion used for selecting those solutions is the crowding distances. Let $M := (x_1, x_2, \dots, x_{|M|})$ be a multi-set of search points, the crowding distance $\textsc{cDist}(x_i, M)$ of x_i with respect to M is determined as follows. At first sort M as $M = (x_{k_1}, \dots, x_{k_{|M|}})$ with respect to each objective $k \in [d]$ separately using a stable sort. Then $\textsc{cDist}(x_i, \text{M}) := \sum_{k=1}^{d} \textsc{cDist}_k(x_i, M)$, where

$$\textsc{cDist}_k(x_{k_i}, M) := \begin{cases} \infty & \text{if } i \in \{1, |M|\}, \\ \frac{f_k(x_{k_{i-1}}) - f_k(x_{k_{i+1}})}{f_k(x_{k_1}) - f_k(x_{k_M})} & \text{otherwise.} \end{cases}$$

The first and last ranked individuals are always assigned an infinite crowding distance. The remaining individuals are then assigned the differences between

the values of f_k of those ranked directly above and below the search point and normalised by the difference between f_k of the first and last ranked. NSGA-II then takes r solutions from $F_t^{i^*}$ with the largest computed crowding distances from $F_t^{i^*}$ to complete P_{t+1} where ties are broken uniformly at random.

All selection mechanisms considered in this paper for NSGA-II use the same criteria in the following lexicographical order to determine whether a search point in P_t is better than another: (i) first by the index of the non-dominated layer that the search point belong to (the smaller the better), (ii) then by its crowding distance (the larger the better).

The benchmark functions LOTZ (LeadingOnesTrailingZeroes), OMM (OneMinMax), and COCZ (CountingOnesCountingZeroes) [23] are:

$$\begin{aligned}
\mathrm{LOTZ}(x) &:= \left(\sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j) \right), \\
\mathrm{OMM}(x) &:= \left(\sum_{i=1}^{n} x_i, n - \sum_{i=1}^{n} x_i \right), \\
\mathrm{COCZ}(x) &:= \left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n/2} x_i + \sum_{i=n/2+1}^{n} (1 - x_i) \right).
\end{aligned}$$

LOTZ optimises the number of leading ones (a prefix of ones) and the number of trailing zeros (a suffix of zeros). OMM minimises and maximises the number of ones. COCZ does the same in the right half of the bit string, and both objectives also include the common goal of maximising the number of ones in the first half.

3 Proving Asymptotic Equivalence

In stochastic tournament selection [2] the parameter k is first sampled $k \sim \mathrm{Unif}([\mu])$, i. e. uniformly between 1 and μ, and then a k-tournament selection is conducted. In a power-law ranking selection with exponent β, the probability of selecting individual x_i of the sorted population $(x_1, \ldots, x_\mu)$ is

$$r_i := \frac{1/i^\beta}{\sum_{i=1}^{\mu} (1/i^\beta)}. \tag{1}$$

The following result shows that the stochastic tournament selection is asymptotically equivalent to the power-law ranking selection with exponent 2.

Lemma 1. *Let $P := (x_1, \ldots, x_\mu)$ be a sequence of points which is sorted according to some criteria, then the stochastic tournament selection on P using the same criteria samples x_i with probability $\sim 1/i^2$.*

Proof. We first determine the probability $p_{i,k}$ that x_i is the winner of a k-tournament for fixed integers $i, k \in [1, \mu]$. Let A_i be the event that $x_1, \ldots, x_i$ are

not among the k sampled individuals, then $\Pr(A_i) = (1 - i/\mu)^k$. Note that x_i is the winner of the k-tournament if and only if A_{i-1} occurs but not A_i, so

$$p_{i,k} = \Pr(A_{i-1} \setminus A_i) = \left(1 - \frac{i-1}{\mu}\right)^k - \left(1 - \frac{i}{\mu}\right)^k.$$

Now, let p_i be the probability that x_i is the winner of a stochastic tournament selection. Since $k \sim \text{Unif}([\mu])$, then by the law of total probability:

$$p_i := \sum_{k=1}^{\mu} \frac{p_{i,k}}{\mu} = \frac{1}{\mu} \sum_{k=1}^{\mu} \left(\left(1 - \frac{i-1}{\mu}\right)^k - \left(1 - \frac{i}{\mu}\right)^k \right). \tag{2}$$

Then we show that $\frac{c_1}{i^2} \leq p_i \leq \frac{c_2}{i^2}$ for every $i \in [\mu]$ where $c_1 := 1 - \frac{2}{e}$ and $c_2 := 2 + \frac{4}{e^2}$. We consider two cases:

Case 1: $i = 1$. Using the geometric sum $\sum_{i=0}^{n} a^i = \frac{1-a^{n+1}}{1-a}$ for $a \neq 1$, we get

$$\begin{aligned} p_1 &= \frac{1}{\mu} \cdot \sum_{k=1}^{\mu} \left(1 - \left(1 - \frac{1}{\mu}\right)^k\right) = 1 + \frac{1}{\mu} - \frac{1}{\mu} \sum_{k=0}^{\mu} \left(1 - \frac{1}{\mu}\right)^k \\ &= 1 + \frac{1}{\mu} - \frac{1}{\mu} \cdot \frac{1 - (1 - 1/\mu)^{\mu+1}}{1 - (1 - 1/\mu)} = \frac{1}{\mu} + \left(1 - \frac{1}{\mu}\right)^{\mu+1}. \end{aligned} \tag{3}$$

Hence $p_1 > (1 - 1/\mu)^2 (1 - 1/\mu)^{\mu-1} > 1/e > 1 - 2/e = c_1/i^2$ and also $p_1 < 1 + 1 < c_2 = c_2/i^2$ for $i = 1$.

Case 2: $2 \leq i \leq \mu$. Again using the geometric sum gives

$$\begin{aligned} p_i &= \frac{1}{\mu} \left(\sum_{k=0}^{\mu} \left(\left(1 - \frac{i-1}{\mu}\right)^k \right) - \sum_{k=0}^{\mu} \left(1 - \frac{i}{\mu}\right)^k \right) \\ &= \frac{1}{\mu} \cdot \left(\frac{1 - (1 - \frac{i-1}{\mu})^{\mu+1}}{1 - (1 - \frac{i-1}{\mu})} - \frac{1 - (1 - \frac{i}{\mu})^{\mu+1}}{1 - (1 - \frac{i}{\mu})} \right) \\ &= \frac{1 - \left(1 - \frac{i-1}{\mu}\right)^{\mu+1}}{i-1} - \frac{1 - \left(1 - \frac{i}{\mu}\right)^{\mu+1}}{i} \\ &= \frac{1 + (i-1)\left(1 - \frac{i}{\mu}\right)^{\mu+1} - i\left(1 - \frac{i-1}{\mu}\right)^{\mu+1}}{i(i-1)} = \frac{1 + b(i)}{i(i-1)} \end{aligned} \tag{4}$$

where

$$b(i) := (i-1)\left(1 - \frac{i}{\mu}\right)^{\mu+1} - i\left(1 - \frac{i-1}{\mu}\right)^{\mu+1}.$$

It then suffices to show that $-\frac{2}{e} = c_1 - 1 \leq b(i) \leq \frac{1}{2}c_2 - 1 = \frac{2}{e^2}$ because this implies the claim for $i \geq 2$:

$$\frac{c_1}{i^2} \leq \frac{b(i)+1}{i^2} \leq \frac{b(i)+1}{i(i-1)} \leq \frac{2b(i)+2}{i^2} \leq \frac{c_2}{i^2}.$$

Since $f(x) := xe^{-x}$ is monotonically decreasing on $[1, +\infty)$ as its derivative $f'(x) = (1-x)e^{-x} \leq 0$, then for $x \geq 2$ we get $f(x) = xe^{-x} \leq f(2) = \frac{2}{e^2}$, and using this gives

$$b(i) \leq i \cdot \left(1 - \frac{i}{\mu}\right)^{\mu+1} \leq i \cdot \left(1 - \frac{i}{\mu}\right)^{\mu} \leq i \cdot e^{-i} \leq \frac{2}{e^2} = \frac{1}{2}c_2 - 1,$$

and

$$-b(i) \leq i \cdot \left(1 - \frac{i-1}{\mu}\right)^{\mu+1} \leq i \cdot \left(1 - \frac{i-1}{\mu}\right)^{\mu} \leq i \cdot e^{-(i-1)} = e \cdot i \cdot e^{-i} \leq \frac{2}{e}.$$

□

This lemma immediately implies that the asymptotic results are transferable between the two selection mechanisms, examples of which are those in [2,7]. The analysis in [7] defines "good" parents as those that have a Pareto-optimal Hamming neighbour not yet present in the population. According to Lemma 4.5 in [7], the probability of selecting a good parent is at least $\min\{p_1, p_2, p_3\}$ since either selecting an extreme point with crowding distance ∞ can create a search point with an even more extreme objective value. Or, in case both 0^n and 1^n have already been found, a third-ranked individual with a finite crowding distance will be neighboured to a gap in the Pareto front. By Lemma 1 we have $\min\{p_1, p_2, p_3\} = \Omega(1)$ and by Lemmas 5.1 and 6.1 in [7] we immediately obtain upper bounds of $\mathcal{O}(n \log n)$ for OMM and $\mathcal{O}(n^2)$ for LOTZ.

Theorem 2 (based on Theorems 5.2 and 6.2 in [7]). *SEMO and GSEMO algorithms using stochastic tournament selection covers the whole Pareto front of* OMM *in* $\mathcal{O}(n \log n)$ *expected fitness evaluations. SEMO using stochastic tournament selection covers the whole Pareto front of* LOTZ *in* $\mathcal{O}(n^2)$ *expected fitness evaluations.*

For stochastic tournament selection, Lemma 1 in [1,2] shows that (i) the probability of selecting individual ranked i is $\Omega(1)$ if $i \in \mathcal{O}(1)$ and (ii) the probability of selecting the worst individual is $\Omega(1/\mu^2)$. However, only (i) was used in the subsequent results in that paper for runtime analysis of NSGA-II, while (ii) was only used to show that the probability of selecting the worst individual is still asymptotically equal to that of binary tournament selection. We have the same property (i) for power-law ranking selections with $\beta \geq 1$.

Lemma 3. *In power-law ranking selection with a constant exponent* $\beta > 1$*, the probability of selecting an individual ranked* $i \in \mathcal{O}(1)$ *is* $\Omega(1)$*.*

Proof. The probability of selecting an individual ranked i is

$$\frac{1/i^\beta}{\sum_{i=1}^{\mu} 1/i^\beta} > \frac{1/i^\beta}{\sum_{i=1}^{\infty} 1/i^\beta} = \frac{1}{\zeta(\beta) i^\beta} = \Omega(1)$$

since $\zeta(\beta) = \mathcal{O}(1)$ for $\beta > 1$, and both $\beta, i \in O(1)$.

Thus we have the following theorem for NSGA-II.

Theorem 4 (based on Theorems 4, 5, 6 in [1]). *NSGA-II using power-law ranking selection with any constant exponent $\beta > 1$ covers the whole Pareto front of* OMM*, or* LOTZ*, or* COCZ *in expected $\mathcal{O}(n^2)$ fitness evaluations if $\mu \in \mathcal{O}(n) \cap [2(n+1), \infty)$ for* OMM *and* LOTZ*, and if $\mu \in \mathcal{O}(n) \cap [n+2, \infty)$ for* COCZ*.*

As we later show experiments with binary tournament selection, we recall the results of [2] for binary tournament selection.

Theorem 5 (Theorem 1, 2, 3 in [1]). *NSGA-II using binary tournament selection covers the whole Pareto front of* OMM*, and* LOTZ *in $\mathcal{O}(n^2 \log n)$, $\mathcal{O}(n^3)$ expected fitness evaluations respectively if $\mu \in \mathcal{O}(n) \cap [2(n+1), \infty)$, and of* COCZ *in $\mathcal{O}(n^2 \log n)$ expected fitness evaluations if $\mu \in \mathcal{O}(n) \cap [n+2, \infty)$.*

Comparing Theorems 4 and 5 confirms that upper bounds for power-law ranking selection are smaller than those for binary tournament selection by a factor of order n. We believe that the latter upper bounds are tight, though this has only been proven for OMM and NSGA-II using fair parent selection [15].

4 Differences in Implementation

In this section, we explain different ways to implement the power-law ranking selection and stochastic tournament selection, and detail their time complexity. We will separate the complexity in terms of basic operations (e. g. comparisons, arithmetic operations, etc.) versus the number of random numbers in $(0, 1)$ (e. g. 64-bit `float`) required since this is the common way pseudo-random numbers are used (this differs from the precise number of random bits analysed in [22] under the name "cost of randomness"). We will also specify the costs that occur by the selection mechanisms at initialisation (before any generation is executed), and in each generation of a (μ+λ) EA, e. g. NSGA-II, or (μ,λ) EA. For NSGA-II, this complements our theoretical results as they only consider the number of function evaluations and do not account for overheads or computational effort during preprocessing for the operators.

4.1 Power-Law Ranking

For the power-law selection of any exponent β, once the population size has been decided at initialisation, the selection probabilities r_i of (1) can be computed in $\Theta(\mu)$ basic operations and stored throughout the run. Then in each generation, if all current individuals have different fitness and hence distinctive ranks, the stored values can be used immediately as the distribution for sampling parents. However, it is often the case that there are equally fit individuals in the population, thus their treatment needs to be specified.

One approach is to assign an arbitrary ranking to equally fit solutions. Then these solutions will receive different probability mass. We denote this variant

of power-law ranking selection by $\textsc{Pow}_{\text{fixed}}$. The other approach, denoted by $\textsc{Pow}_{\text{adjusted}}$, is to assign equal probability mass to equally fit individuals by redistributing their total probability mass. For example, if solutions ranked $3, 4, 5$ are equally fit then they will be selected with the same probability $(r_3+r_4+r_5)/3$. Note that compared to $\textsc{Pow}_{\text{fixed}}$, $\textsc{Pow}_{\text{adjusted}}$ favours more selecting diverse equally fit individuals by the following reasoning. Assume only those ranked 3 and 4 are equally fit. The chance of selecting both these two in two independent applications of $\textsc{Pow}_{\text{fixed}}$ is $s_1 := 2r_3r_4$, while that of $\textsc{Pow}_{\text{adjusted}}$ is $s_2 := 2((r_3 + r_4)/2)^2$ and we note that $s_2 \geq s_1$ because this is equivalent to $(r_3 - r_4)^2 \geq 0$.

The cost of the re-adjustment is $\Theta(\mu)$ basic operations per generation as only going through the values of r_i a few times is required. This cost is asymptotically superseded by that of sorting the population to identify the rank for each individual, which is in the order of $\mathcal{O}(\mu \log \mu)$. Once the selection distribution is adjusted, in order to select λ parents (where $\lambda = 2\mu$ for NSGA-II) to produce offspring, fast sampling methods for custom distributions such as the *alias method* of sampling [12,32,33] or the inverse transform method [12] can be used so that the cost in terms of random numbers is constant per parent selection. An overall cost of $\Theta(\lambda)$ basic operations also occurs as the indices of the selected parents in the population have to be stored.

Proposition 6. $\textsc{Pow}_{\text{adjusted}}$ *and* $\textsc{Pow}_{\text{fixed}}$ *requires* $\Theta(\mu)$ *basic operations at initialisation and* $\mathcal{O}(\mu \log \mu + \lambda)$ *basic operations and* $\Theta(\lambda)$ *random numbers in each generation if* λ *parents are sampled.*

4.2 Tournament Selection

Binary tournament selection, denoted as Bin, is popular since it is inexpensive. No cost occurs during initialisation and in each generation when each parent selection uses $\mathcal{O}(1)$ basic operations and $\mathcal{O}(1)$ random numbers, i. e. sampling two random numbers u_1, u_2 in $(0, 1)$ and then returning the best individual among $P_t(\lceil u_1\mu \rceil)$ and $P_t(\lceil u_2\mu \rceil)$.

Proposition 7. Bin *sampling* λ *parents uses* $\Theta(\lambda)$ *basic operations and* $\Theta(\lambda)$ *random numbers.*

Stochastic tournament selection can also be implemented directly based on its description in Sect. 3, and this implementation is denoted Sto. No cost occurs during initialisation for this implementation. However, the cost for each parent selection in each generation is high because the tournament size sampled from $\text{Unif}([\mu])$ is at least $\mu/2 = \Theta(\mu)$ in expectation and also in median, and each member of the tournament requires a random number. Thus by linearity of expectation, to sample λ parents, $\lambda\mu/2 = \Theta(\mu\lambda)$ random numbers are used in expectation. This is by a factor of μ more expensive than the previous mechanisms. Furthermore, the number of parents sampled using more than $\mu/2$ random numbers dominates $\text{Bin}(\lambda, 1/2)$, thus by a Chernoff bound the probability having less than $\lambda/3$ such parents sampled is $2^{-\Omega(\lambda)}$. This implies with probability $1 - 2^{-\Omega(\lambda)}$, at least $(\lambda/3)(\mu/2) = \Theta(\lambda\mu)$ random numbers are required. These arguments also hold for the number of basic operations.

Proposition 8. STO *sampling* λ *parents uses* $\Theta(\lambda\mu)$ *basic operations and* $\Theta(\lambda\mu)$ *random numbers in expectation, and also with a probability of at least* $1-2^{-\Omega(\lambda)}$.

Because this high cost, particularly in terms of randomness resource, we suggest to implement the selection similarly to the power-law ranking selection as described in the previous section. This implementation also has two variants depending on whether the selection probabilities, now p_i from (2), are adjusted to equally fit individuals or not. These variants are denoted $\text{STO}_{\text{adjusted}}$ and $\text{STO}_{\text{fixed}}$, respectively. Note also that if we naively use (2) to compute each p_i then this costs $\Theta(\mu)$, thus overall the cost of basic operations at initialisation is $\Theta(\mu^2)$ as there are μ values to compute. However, this cost is only $\Theta(\mu)$ if we use the formulations of (3) and (4) instead.

Proposition 9. $\text{STO}_{\text{adjusted}}$ *and* $\text{STO}_{\text{fixed}}$ *sampling* λ *parents use* $\mathcal{O}(\mu \log \mu + \lambda)$ *basic operations and* $\Theta(\lambda)$ *random numbers, in addition to a preprocessing cost of* $\Theta(\mu)$ *basic operations at initialisation.*

5 Empirical Results

Now we conduct experiments with NSGA-II on the functions studied in [1,2] to complement our theoretical results and to get insights into hidden constants.

5.1 Experimental Setup

Our code is written in Python and we use the implementation of NSGA-II from the DEAP library [17] as it allows the possibility to redefine existing operators and introduce new ones and features the fast non-dominated sorting from [16]. A drawback of DEAP at the time we write this paper is that it mixes up the use of pseudo-random numbers from both the standard Python library (Python's `random`) and from that of NumPy across the operators used by NSGA-II. This can cause problems for reproducibility of the results as one has to make sure to initialise the random seeds for both streams of pseudo-random numbers. For this reason, we redefine the necessary operators, i. e. copying from the library and making necessary changes, to make sure that only pseudo-random numbers from NumPy are used. For the mutation operator, since we only deal with the bitwise mutation with standard mutation rate $1/n$, we implement the fast operator using samples from a geometric distribution, see [20,31].

All the variants BIN, $\text{POW}_{\text{fixed}}$, $\text{POW}_{\text{adjusted}}$, STO, $\text{STO}_{\text{fixed}}$, $\text{STO}_{\text{adjusted}}$ are implemented. Operators STO and $\text{STO}_{\text{adjusted}}$ always use the same distribution to select parents and only differ in their implementation. All other pairs of operators use different distributions to select parents when the population contains equally fit individuals. Our experiments are conducted on all the three functions OMM, LOTZ, and COCZ. To sample from custom distributions in constant time we use the alias method of sampling [33] and follow the implementation of [9,10].

The population is always set to the minimal requirement by Theorem 4, i. e. $\mu = 2(n+1)$ for LOTZ and OMM, and $\mu = n+2$ for COCZ. For each configuration of the algorithm running on a specific problem size, 100 independent runs

are produced and each run is initialised with different random seed. The experiments are conducted on a server machine with AMD EPYC 7443 processor and each run is reserved one computer core with 2 GB of RAM. Our code along with all the results are available at https://gitlab.com/d2cmath/nsga2-sts-pow.

5.2 Results on Low Dimension Problems

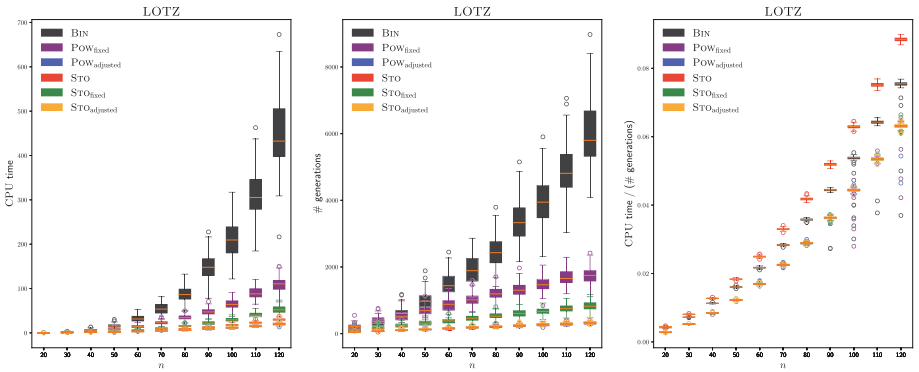


Fig. 1. Empirical results on LOTZ for $n \in \{20, 30, \ldots, 120\}$. The CPU times are in seconds. The plot to the right shows the average CPU time spent per generation for each variant of NSGA-II.

The purpose of this series of experiments is to complement the findings by [2] that Bin is inefficient in terms of numbers of generations and fitness evaluations compared to the other selections on all studied problems, cf. Theorems 4 and 5. For this, we only use the problem dimensions between 20 and 120 with a step size of 10. Figure 1 illustrates the results for LOTZ, and the other results can be found in the provided link. From the middle plot of the figure, we notice that Bin requires significant more generations to optimise the functions as predicted by the theory [2]. This led to the high computational time in total as shown in the left plot, although Bin spends less CPU time per generation than Sto as shown in the right plot.

5.3 Results on High Dimension Problems

We take the inefficient selection Bin out and consider the remaining selection operators for higher dimension problems, i.e. with n from 20 up to 400 with a step size of 20, see Fig. 2. From the first and second row of the figure, we notice that both $\text{Pow}_{\text{fixed}}$ and $\text{Sto}_{\text{fixed}}$ are far worse than the other implementations in terms of number of generations and overall CPU times. The discrepancy between $\text{Pow}_{\text{adjusted}}$ and $\text{Pow}_{\text{fixed}}$ is astounding, and this is due to $\text{Pow}_{\text{adjusted}}$ favouring more selecting equally fit but diverse individuals as explained in Sect. 4.1.

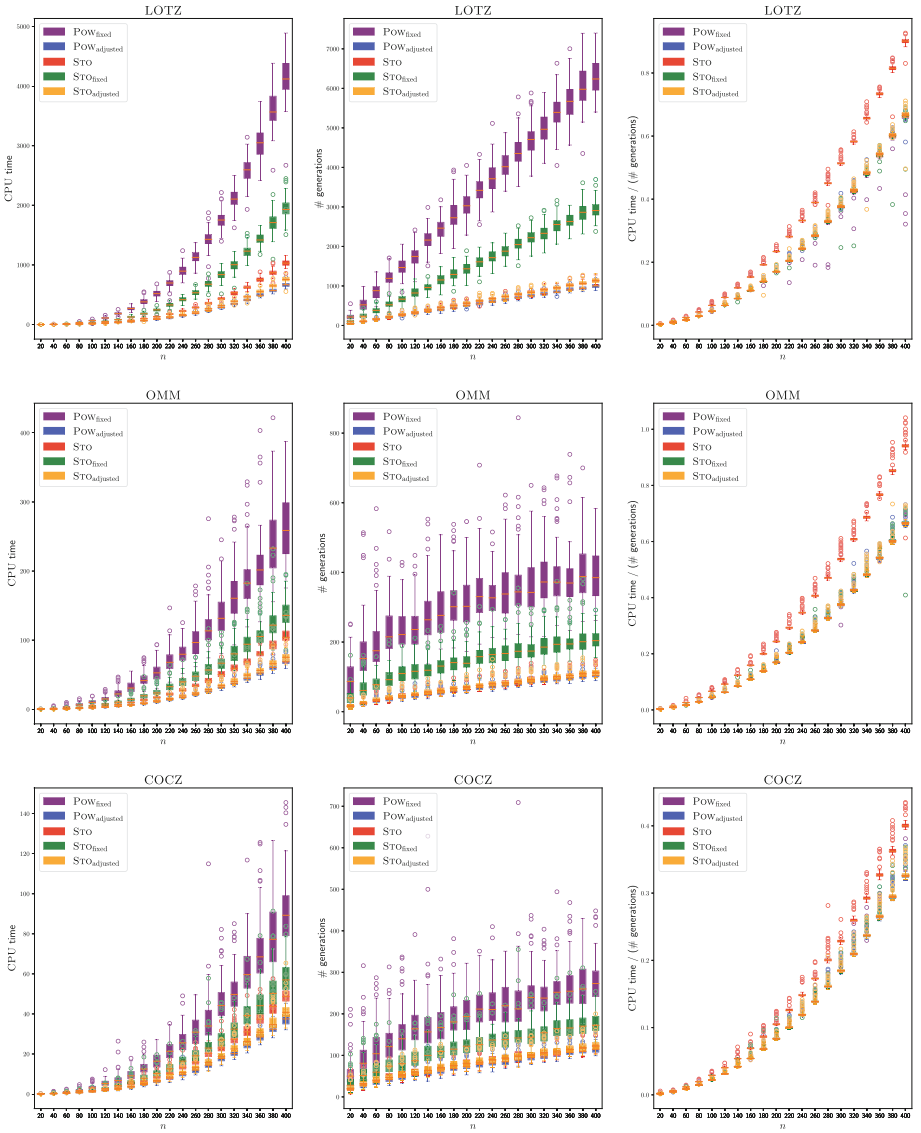


Fig. 2. Empirical results on LOTZ, OMM, COCZ for $n \in \{20, 40, \ldots, 400\}$. The CPU times are in seconds. The last column of plots shows the average CPU time spent per generation for each variant of NSGA-II.

Particularly, if we look at the case of LOTZ as seen in the top-left plot, this is the difference between solving a problem in more than one hour versus in 10 minutes. The best implementations are $\textsc{Pow}_{\text{adjusted}}$, $\textsc{Sto}_{\text{adjusted}}$, and $\textsc{Sto}$ as their differences are hard to distinguish in the figure. We therefore extract them for the case of LOTZ to show in Fig. 3. The middle plot of that figure confirms

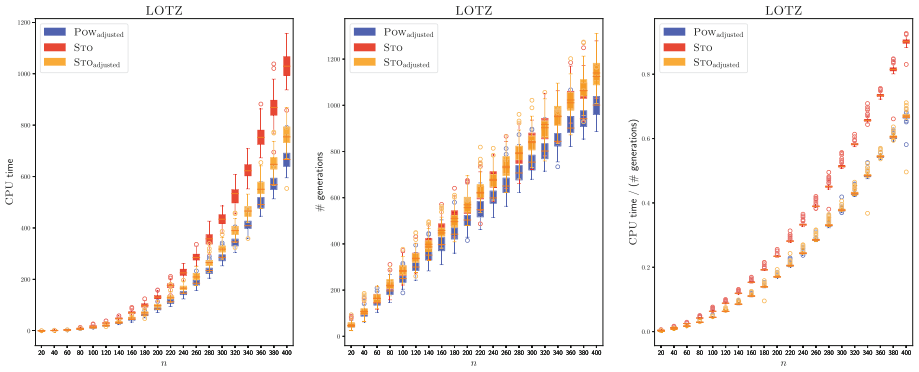


Fig. 3. Empirical results for $\text{POW}_{\text{adjusted}}$, $\text{STO}_{\text{adjusted}}$, STO on LOTZ for $n \in \{20, 40, \ldots, 400\}$. The CPU times are in seconds.

that $\text{STO}_{\text{adjusted}}$ and STO are exactly the same mechanism as they require the same number of generations to optimise the function. Nevertheless $\text{STO}_{\text{adjusted}}$ is more efficiently implemented as it results in shorter computational time in total, cf. the left plot. In all experiments, STO always spends the most CPU time per generation compared to all other selection mechanism as predicted in Sect. 4.2.

6 Conclusions

We have shown that stochastic tournament selection is asymptotically equivalent to power-law ranking selection with exponent 2. This yield a better understanding of stochastic tournament selection and it allows the direct transfer of runtime results between the two selection mechanisms for various algorithms studied in the literature [2,7]. We have investigated the efficient implementations of these mechanisms and found out that stochastic tournament selection can be implemented more efficiently if done similarly to power-law ranking with pre-computed selection probabilities. This improvement has been assessed by empirical results conducted on synthetic benchmark functions. The reason for this improvement is that the implementation reduces the cost of randomness (number of random variables) from a quadratic order of the population size to a linear one. Our experiments also showed that overall the power-law ranking selection optimises the studied function using the shortest CPU times, hence it is the most efficient mechanism. This is because numerically (and not asymptotically) the mechanism attributes larger probabilities to select top-ranked individuals, compared to stochastic tournament selection. An insight from our study is that specifically when crossover is involved the treatment of equally good individuals has to be done properly, i.e. they should receive the same selection probability, because this seemingly small detail can make a surprisingly difference in the performance, i.e. decide between the best versus the worst implementations.

References

1. Bian, C., Qian, C.: Running time analysis of the non-dominated sorting genetic algorithm II (NSGA-II) using binary or stochastic tournament selection. arXiv preprint arXiv:2203.11550 (2022)
2. Bian, C., Qian, C.: Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) PPSN 2022, pp. 428–441. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14721-0_30
3. Bian, C., Zhou, Y., Li, M., Qian, C.: Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. Proc. Int. Joint Conf. Artif. Intell. **2023**, 5513–5521 (2023)
4. Blickle, T., Thiele, L.: A comparison of selection schemes used in evolutionary algorithms. Evol. Comput. **4**(4), 361–394 (1996)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press (2009)
6. Corus, D., Lissovoi, A., Oliveto, P.S., Witt, C.: On steady-state evolutionary algorithms and selective pressure: why inverse rank-based allocation of reproductive trials is best. ACM Trans. Evolution. Learn. Optimiz. **1**(1), 2:1–2:38 (2021)
7. Covantes Osuna, E., Gao, W., Neumann, F., Sudholt, D.: Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. Theor. Comput. Sci. **832**, 123–142 (2020)
8. Dang, D.-C., Eremeev, A.V., Lehre, P.K.: Non-elitist evolutionary algorithms excel in fitness landscapes with sparse deceptive regions and dense valleys. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2021), pp. 1133–1141. ACM (2021)
9. Dang, D.-C., Eremeev, A.V., Lehre, P.K., Qin, X.: Fast non-elitist evolutionary algorithms with power-law ranking selection. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2022), pp. 1372–1380. ACM (2022)
10. Dang, D.-C., Eremeev, A.V., Qin, X.: Empirical evaluation of evolutionary algorithms with power-law ranking selection. In: Shi, Z., Torresen, J., Yang, S. (eds.) IIP 2024, Part I, pp. 217–232. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-57808-3_16
11. 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)
12. Devroye, L.: Non-Uniform Random Variate Generation. Springer, New York (1986). https://doi.org/10.1007/978-1-4613-8643-8
13. Doerr, B.: Does comma selection help to cope with local optima? Algorithmica **84**(6), 1659–1693 (2022)
14. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation - Recent Developments in Discrete Optimization. Springer, Cham (2020)
15. Doerr,B., Qu, Z.: From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2023, pp. 12408–12416. AAAI Press (2023)
16. Fortin, F., Grenier, S., Parizeau, M.: Generalizing the improved run-time complexity algorithm for non-dominated sorting. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013), pp. 615–622. ACM (2013)

17. Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., Gagné, C.: DEAP: evolutionary algorithms made easy. J. Mach. Learn. Res. **13**, 2171–2175 (2012)
18. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. Evol. Comput. **18**(3), 335–356 (2010)
19. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley (1989)
20. Grefenstette, J.: Efficient implementation of algorithms. In: Handbook of Evolutionary Computation, 1st edn., pp. E2.1:1–E2.1:6. IOP Publishing Ltd. (1997)
21. Jansen, T., Zarges, C.: Analysis of evolutionary algorithms: from computational complexity analysis to algorithm engineering. In: Proceedings of Foundations of Genetic Algorithms (FOGA 2011), pp. 1–14. ACM (2011)
22. Kneissl, C., Sudholt, D.: The cost of randomness in evolutionary algorithms: crossover can save random bits. In: Pérez Cáceres, L., Stützle, T. (eds.) EvoCOP 2023, EvoStar 2023, pp. 179–194. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30035-6_12
23. 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)
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
25. Lehre, P.K.: Fitness-levels for non-elitist populations. In: Proceedings the Genetic and Evolutionary Computation Conference (GECCO 2011), pp. 2075–2082. ACM (2011)
26. Motoki, T.: Calculating the expected loss of diversity of selection schemes. Evol. Comput. **10**(4), 397–422 (2002)
27. Neumann, F., Oliveto, P.S., Witt, C.: Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2009), pp. 835–842. ACM (2009)
28. Oliveto, P.S., Witt, C.: On the runtime analysis of the simple genetic algorithm. Theoret. Comput. Sci. **545**, 2–19 (2014)
29. Oliveto, P.S., Witt, C.: Improved time complexity analysis of the simple genetic algorithm. Theoret. Comput. Sci. **605**, 21–41 (2015)
30. Oliveto, P.S., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: a decade of results. Int. J. Autom. Comput. **4**(3), 281–293 (2007)
31. Rudolph, G., Ziegenhirt, J.: Computation time of evolutionary operators. In: Handbook of Evolutionary Computation, 1st edn., pp. E2.2:1–E2.2:4. IOP Publishing Ltd. (1997)
32. Vose, M.D.: A linear algorithm for generating random numbers with a given distribution. IEEE Trans. Softw. Eng. **17**(9), 972–975 (1991)
33. Walker, A.J.: New fast method for generating discrete random numbers with arbitrary frequency distributions. Electron. Lett. **10**(8), 127–128 (1974)

Level-Based Theorems for Runtime Analysis of Multi-objective Evolutionary Algorithms

Duc-Cuong Dang, Andre Opris(✉), and Dirk Sudholt

University of Passau, Passau, Germany
andre.opris@uni-passau.de

Abstract. Runtime analysis of multi-objective evolutionary algorithms (MOEAs) is a rapidly emerging field in which recent breakthroughs studied state-of-the-art MOEAs like NSGA-II and NSGA-III. These analyses typically bound the expected time to cover the Pareto front by analysing (1) the expected time to find a first Pareto-optimal search point and (2) the expected time to cover the whole Pareto front from there.

We support this development by providing a powerful general tool for bounding the expected time to reach a first Pareto-optimal search point. It is based on the well-known fitness-level method, a simple and versatile yet powerful analysis method, adapted to multiple objectives. The benefits are to simplify runtime analyses by removing repetitive arguments used across many runtime analyses, thus allowing for shorter and simpler proofs, and to make runtime analysis of MOEAs more accessible to other researchers. Our level-based theorems further provide additional results on stochastic domination and tail bounds in addition to bounds on expected hitting times. We identify sufficient conditions for NSGA-II and NSGA-III to reach the Pareto front, which may pave the way for runtime analyses of state-of-the-art MOEAs approximating the Pareto front with population sizes smaller than the Pareto front.

Keywords: Runtime analysis · analysis methods · evolutionary multi-objective optimisation · theory

1 Introduction

Many optimisation problems found in practice feature multiple objectives that are often conflicting. Evolutionary algorithms are well suited to dealing with multiple objectives owing to their innate ability to store trade-offs in their population. State-of-the-art multi-objective evolutionary algorithms (MOEAs) like the non-dominated sorting algorithm II (NSGA-II) [19] or its successor for many-objective optimisation, NSGA-III [18], have found thousands of applications and have amassed over 50,000 citations between them. Yet there is little theoretical underpinning and the reasons behind their success are not well understood.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 246–263, 2024.
https://doi.org/10.1007/978-3-031-70071-2_16

Recent breakthrough papers managed to analyse the runtime of these state-of-the-art algorithms for common pseudo-Boolean benchmark problems. Zheng, Liu, and Doerr [46] gave the first runtime analysis of NSGA-II (without crossover) on the LOTZ benchmark, showing that the algorithm efficiently covers the whole Pareto front if the population size is large enough. Several subsequent papers demonstrated the usefulness of using crossover in NSGA-II [3,12,14,26]. Further analyses illustrating the efficiency of NSGA-II considered multimodal problems [15,23,24], noisy optimisation [13] and combinatorial optimisation [5]. Other papers considered improvements to the crowding distance computation [44], limitations in many-objective optimisation [45] and lower bounds on the runtime [25]. NSGA-III was first analysed in the breakthrough paper by Wietheger and Doerr [41] on a 3-objective benchmark problem and was recently shown to be effective on many-objective problems for large enough populations.

All these papers have enhanced our understanding of the limits and capabilities of these algorithms. However, the theory of MOEAs is still in its infancy. In particular, while powerful tools have been developed for analysing single-objective evolutionary algorithms (e.g., the fitness-level method and various extensions of it [29,33,40]), such tools are lacking for multi-objective optimisation. Consequently, the analysis of MOEAs is often done from scratch and proofs frequently use the same repetitive arguments. This work aims to advance this line of research by contributing a powerful and versatile method to simplify proofs and to obtain more powerful statements.

The well-known fitness-level method, which originated in [34,37] and was popularised in [40], divides the search space into sets ("levels") $A_0, \dots, A_k$ of increasing fitness and maps the current population of an evolutionary algorithm to a level. Elitist algorithms can never worsen their current level. If we have a lower bound s_i on the probability of leaving a level A_i, we obtain an upper bound of $1/s_i$ on the expected waiting time for this to happen and summing over all levels yields an upper bound on the expected time to find the highest fitness level (e.g., the set of global optima).

Here we review, refine and adapt the method to MOEAs to simplify a crucial part of existing and future runtime analyses. In most recent runtime analyses reviewed above, the goal was to cover the whole Pareto front and the analyses followed a common pattern: dividing a run into two phases. Phase 1 describes the time to create the first Pareto-optimal search point. Phase 2 covers the time to cover the whole Pareto front. For Phase 1 it is often sufficient to consider Pareto dominance, whereas for Phase 2 the specific details of the algorithm matter, e. g. deciding which incomparable solutions should survive, or whether or not to keep solution copies [15]. For algorithms that are based on Pareto-dominance, e.g., (G)SEMO [27,32], NSGA-II [19], NSGA-III [18] and SMS-EMOA [2] the same arguments are used in Phase 1. By providing a general analysis method that can be easily plugged into proofs, we aim to reduce the use of repetitive arguments, allowing for shorter and simpler proofs. We also aim to make runtime analysis of MOEAs accessible to other researchers as the new method is versatile and easy

to use. Moreover, we include several statements that go beyond a bound on the expectation of the time to find the Pareto front, including stochastic domination and tail bounds. This enhances existing analyses at no additional cost.

Finally, we obtain more clarity on required parameter settings that guarantee to discover the Pareto front efficiently, and find that these can be much looser than requirements for covering the whole Pareto front. We also believe that future research effort will be devoted to studying smaller populations. The goal of covering the whole Pareto front is only achievable if the population is at least as large as the Pareto front. In practice, this is not always realistic or reasonable and smaller populations are preferred, provided they give a good approximation of the Pareto front. By researching minimum requirements on parameters (e.g., the population size) to reach the Pareto front, we hope to facilitate this development.

2 Preliminaries

2.1 The Fitness-Level Method

The fitness-level method, also known as method of f-based partitions, was first formalised by Wegener [40]. It partitions the search space into non-empty sets $A_0, \ldots, A_k$ called *levels*, in a way that the final set A_k contains desirable solutions and there is some progression towards reaching levels of higher index. In its original formulation, the levels are sorted according to fitness (maximising some function f): for all $i \in \{0, \ldots, k-1\}$, all $x \in A_i$ and all $y \in A_{i+1}$ we have $f(x) < f(y)$. Then A_k contains search points with the highest fitness, and it is often restricted to global optima.

For an elitist (1+1) evolutionary algorithm we consider the fitness level A_i that contains the current search point and say that the algorithm is "in A_i" or "on level i" if its current search point is in A_i. If we have a lower bound s_i on the probability of reaching a higher fitness-level set in one generation, and this lower bound holds for all populations on this fitness level, the expected time until a better fitness level is found is stochastically dominated by a geometric random variable with parameter s_i and the expected time until this happens is at most $1/s_i$. Owing to elitism, the current level can never decrease. By summing up these waiting times for all non-optimal fitness levels, starting with the initial one, we obtain an upper bound of

$$\sum_{j=0}^{k-1} \Pr\left(\text{start in } A_j\right) \sum_{i=j}^{k-1} \frac{1}{s_i} \leq \sum_{i=0}^{k-1} \frac{1}{s_i}.$$

The inequality is trivial and subsequent applications often used the bound on the right-hand side for the sake of simplicity and since often there is very little gain in considering the initialisation.

The fitness-level method found many applications beyond (1+1) evolutionary algorithms by appropriately redefining the notion of the current fitness level (e.g. by considering the best fitness level in the current population). It

was extended to parent populations [42], ant colony optimisation [28,35], a binary particle swarm optimiser [39], offspring populations [31] and island models [30,31].

The fitness-level method was also used to prove *lower* bounds on the expected running time of EAs, by exploiting additional information on transition probabilities between fitness levels [38]. This approach was very recently simplified in [22] and refined by combining fitness levels with drift analysis [29].

The fitness-level method turned out also to be applicable to non-elitist algorithms, assuming that the probability of decreasing the current level is small enough. In a breakthrough paper, Lehre [33] and later Dang and Lehre [10] introduced the fitness-level method for non-elitist populations for which the strong requirement on the progress is relaxed to a probabilistic condition and a lower bound on the population size (to ensure the stability of the process). The method was later extended into the so-called *level-based* method [7] which can be applied to a variety of algorithms, including estimation of distribution algorithms [9], and noisy settings [8], and led to the introduction of performance-guaranteed algorithms for combinatorial problems [6] and new operators [11]. An improved version of the level-based method was developed in [21].

An advantage of fitness-levels for elitist populations is that the time to reach the last level is stochastically dominated by sums of geometric random variables, with probability bounds s_i as parameters. This was used in Zhou, Luo, Lu, and Han [47] to derive tail bounds on the time to reach the last fitness level. Such tail bounds were later improved by Witt [43]. Doerr [20] advocated the description of search processes using stochastic domination. This allows for more informative performance guarantees, decouples the algorithmic task of finding domination statements from probability-theoretical derivations of desired probabilistic guarantees, and helps to find simpler and more natural proofs [20].

2.2 Multi-objective Benchmark Functions

In this paper we look at maximisation problems of an m-objective function $f(x) := (f_1(x), \dots, f_m(x))$ where $f_i : \{0,1\}^n \to \mathbb{N}_0$ for each $i \in [m]$. Suppose that the range of f_i is the integer set $\{0, 1, \dots, f_i^{\max}\}$ for all $i \in \{1, \dots, m\}$ and let $f^{\text{sum}} := \sum_{i=1}^{m} f_i^{\max}$ be the sum of all fitness ranges.

Definition 1. *Consider an m-objective function f.*

(1) Given two search points $x, y \in \{0,1\}^n$, x weakly dominates *y, denoted by $x \succeq y$ if $f_i(x) \geq f_i(y)$ for all $1 \leq i \leq m$; and x* (strictly) dominates *y, denoted $x \succ y$, if one inequality is strict; if neither $x \succeq y$ nor $y \succeq x$ then x and y are* incomparable.

(2) A set $S \subseteq \{0,1\}^n$ is a set of mutually incomparable solutions *with respect to f if all search points in S are incomparable; thus, any two search points in S have distinct fitness vectors.*

(3) Each solution that is not dominated by any other solution in $\{0,1\}^n$ is called Pareto-optimal. A set of these solutions that covers all possible

non-dominated fitness values and are mutually incomparable is called a Pareto(-optimal) *set of* f.

When $m = 2$, the function is called also *bi-objective*. We will illustrate our level-based method on the following standard benchmarks. The function m-LOTZ aims to maximise the number of leading ones (a prefix of only ones) and the number of trailing zeros (a suffix of only zeros) at the same time. The parameter m describes the number of objectives. For $m > 2$ the bit string is divided into $m/2$ blocks of size $2n/m$ each.

Definition 2 (Laumanns et al. [32]). *Let m be divisible by 2 and let the problem size be a multiple of $m/2$. The m-objective function m-*LOTZ *is defined by m-*LOTZ $: \{0,1\}^n \to \mathbb{N}_0^m$ *as m-*LOTZ$(x) = (f_1(x), f_2(x), \ldots, f_m(x))$ *with*

$$f_k(x) = \begin{cases} \sum_{i=1}^{2n/m} \prod_{j=1}^{i} x_{j+n(k-1)/m}, & \text{if } k \text{ is odd,} \\ \sum_{i=1}^{2n/m} \prod_{j=i}^{2n/m} (1 - x_{j+n(k-2)/m}), & \text{otherwise,} \end{cases}$$

for all $x = (x_1, \ldots, x_n) \in \{0,1\}^n$.

The function m-COCZ aims to maximise the number of ones and the number of zeros in parts of the bit string. More specifically, the second half of the bit string is divided into $m/2$ blocks of length n/m each. Moreover, each objective also adds the number of ones in the first half of the bit string. Hence this function models a common goal as well as conflicting goals.

Definition 3 (Laumanns et al. [32]). *Let m be divisible by 2 and let the problem size be a multiple of m. The m-objective function m-*COCZ *is defined by m-*COCZ $: \{0,1\}^n \to \mathbb{N}_0^m$ *as m-*COCZ$(x) = (f_1(x), \ldots, f_m(x))$ *with*

$$f_k(x) = \sum_{i=1}^{n/2} x_i + \begin{cases} \sum_{i=1}^{n/m} x_{i+n/2+(k-1)n/(2\,m)}, & \text{if } k \text{ is odd,} \\ \sum_{i=1}^{n/m} \left(1 - x_{i+n/2+(k-2)n/(2\,m)}\right), & \text{otherwise,} \end{cases}$$

for all $x = (x_1, \ldots, x_n) \in \{0,1\}^n$.

2.3 Multi-objective Evolutionary Algorithms

We start our review of MOEAs with the simple algorithm (G)SEMO (Algorithm 1). One solution is initialised uniformly at random, and in each generation a new offspring solution y is created by mutating a parent solution p selected uniformly at random from the current population P_t. SEMO flips one bit chosen uniformly at random, whereas GSEMO uses standard bit mutation. If y is not weakly dominated by any solutions of P_t then it is added to P_t, and the next population P_{t+1} is computed by removing all those individuals weakly dominated by y from P_t. The population size $|P_t|$ may vary over time.

NSGA-II [17,19] and NSGA-III [18] using standard bit mutation are shown in Algorithm 2. Both algorithms start with a randomly initialised population of

Algorithm 1: (G)SEMO Algorithm on $\{0,1\}^n$

```
Initialise P_0 := {x} where x ~ Unif({0,1}^n);
for t := 0 → ∞ do
    Sample p ~ Unif(P_t);
    Create y by mutating p using 1-bit flips (SEMO) or standard bit mutation
    (GSEMO);
    if ∄x ∈ P_t : x ≻ y then
        Create the next population P_{t+1} := P_t ∪ {y};
        Remove all x ∈ P_{t+1} where y ⪰ x;
```

μ solutions, and then create new μ offspring solutions. Then the 2μ solutions (the population $R_t = P_t \cup Q_t$) compete for survival in the next population P_{t+1}.

We use the most basic choices for the parent selection and mutation operators to generate the offspring: each offspring solution is created independently from each other by first picking a parent uniformly at random (with replacement) and then applying a mutation operator to the copied parent.

In survival selection, the parent and offspring populations P_t and Q_t are joined into R_t, and then partitioned into layers $F^1_{t+1}, F^2_{t+1}, \dots$ by the *non-dominated sorting algorithm* [19]. The layer F^1_{t+1} consists of all non-dominated points, and F^i_{t+1} for $i > 1$ only contains points that are the non-dominated solutions after removing $F^1_{t+1}, \dots, F^{i-1}_{t+1}$. Then a critical layer $F^{i^*}_{t+1}$ is computed such that $\sum_{i=1}^{i^*-1} |F^i_{t+1}| < \mu$ and $\sum_{i=1}^{i^*} |F^i_{t+1}| \geq \mu$. Then P_{t+1} consists of all solutions from the layers $F^1_{t+1}, \dots, F^{i^*-1}_{t+1}$ and of $r := \mu - \sum_{i=1}^{i^*-1} |F^i_{t+1}|$ remaining solutions from $F^{i^*}_{t+1}$. To determine those remaining r ones, NSGA-II uses the crowding distance while NSGA-III uses the distance to a predefined set of reference rays after a normalisation procedure.

The detailed normalisation procedure can be found in [4]. The following description and properties are enough for our purposes in this paper. For an m-objective function $f\colon \{0,1\}^n \to (\mathbb{N}_0)^m$, the normalised fitness vector $f^n(x) := (f^n_1(x), \dots, f^n_m(x))$ of search point x is computed as

$$f^n_j(x) = \frac{f_j(x) - y^{\min}_j}{y^{\text{nad}}_j - y^{\min}_j} \tag{1}$$

for each $j \in [m]$. The points $y^{\min} := (y^{\min}_1, \dots, y^{\min}_m)$ and $y^{\text{nad}} := (y^{\text{nad}}_1, \dots, y^{\text{nad}}_m)$ are referred to as the *ideal* and *nadir* points, respectively, of the objective space. Particularly, $y^{\min}_j$ is set to the minimum value in objective j either from the current population, or from all search points that the algorithm has determined so far. To simplify the analysis, we suppose that $y^{\min}_j$ is based only on the individuals from F^1_{t+1} of the current population. For the full normalisation procedure we refer to [4].

Algorithm 2: NSGA-II [19] and NSGA-III [18] on $\{0,1\}^n$

1 Initialise $P_0 \sim \text{Unif}((\{0,1\}^n)^\mu)$;
2 **for** $t := 0$ *to* ∞ **do**
3 Initialise $Q_t := \emptyset$;
4 **for** $i = 1$ *to* μ **do**
5 Sample p from P_t uniformly at random;
6 Create x by mutating p and add x to Q_t;
7 Let $R_t := P_t \cup Q_t$;
8 Partition R_t into layers $F^1_{t+1}, F^2_{t+1}, \ldots$ using non-dominated sorting [19];
9 Compute the critical layer $i^* \geq 1$ such that $\sum_{i=1}^{i^*-1} |F^i_{t+1}| < \mu$ and $\sum_{i=1}^{i^*} |F^i_{t+1}| \geq \mu$;
10 Set $Y_t := \bigcup_{i=1}^{i^*-1} F^i_{t+1}$;
11 Select a multiset $\tilde{F}^{i^*}_{t+1} \subset F^{i^*}_{t+1}$ of individuals such that $|Y_t \cup \tilde{F}^{i^*}_{t+1}| = \mu$: use crowding distance for NSGA-II and distances to reference points (Algorithm 3) for NSGA-III;
12 Create the next population $P_{t+1} := Y_t \cup \tilde{F}^{i^*}_{t+1}$;

After normalization, each individual of rank at most i^* is associated with the reference point $\text{rp}(x)$ such that the distance between $f^n(x)$ and the line through the origin and $\text{rp}(x)$ is minimized (see Algorithm 3). Here ties are broken deterministically, i.e. two individuals which have the same smallest distance to two reference points are assigned to the same reference point. Then one iterates through all the reference points where the reference point with the fewest associated individuals that are already selected for the next generation P_{t+1} is chosen. Ties are broken uniformly at random. A reference point is omitted if it only has associated individuals that are already selected for P_{t+1}. Then, among the not yet selected individuals of that reference point the one nearest to the chosen reference point is taken for P_{t+1} where ties are again broken uniformly at random. The selection terminates if the required number of individuals is reached (i.e. if $|Y_t| + |\tilde{F}^i_t| = \mu$). (Compare also with [41] or [36].)

To define the set of reference points $\mathcal{R}_p$ the original paper [18] suggests the method from Das and Denis [16] to define this set with a parameter $p \in \mathbb{N}$:

$$\mathcal{R}_p := \left\{ \left(\frac{a_1}{p}, \ldots, \frac{a_d}{p} \right) \mid (a_1, \ldots, a_d) \in \mathbb{N}_0^d, \sum\nolimits_{i=1}^{d} a_i = p \right\}. \tag{2}$$

In case of NSGA-II the selection in line 12 in Algorithm 2 is based on the crowding distance $\text{cDist}(x_i, M)$ of x_i with respect to M where $M := (x_1, x_2, \ldots, x_{|M|})$ is a multi-set of search points. At first we sort M to obtain $M = (x_{k_1}, \ldots, x_{k_{|M|}})$ with respect to each objective $k \in [m]$ separately. Then $\text{cDist}(x_i,\ \text{M}) := \sum_{k=1}^{d} \text{cDist}_k(x_i, M)$, is calculated where

$$\text{cDist}_k(x_{k_i}, M) := \begin{cases} \infty & \text{if } i \in \{1, |M|\}, \\ \frac{f_k(x_{k_{i-1}}) - f_k(x_{k_{i+1}})}{f_k(x_{k_1}) - f_k(x_{k_M})} & \text{otherwise.} \end{cases} \tag{3}$$

Algorithm 3: Distance to reference points [18] to compute $\tilde{F}^{i^*}_{t+1}$ from $F^{i^*}_{t+1}$ based on a set $\mathcal{R}_p$ of reference points.

Compute the normalised $f^n(x)$ for each $x \in Y_t \cup F^{i^*}_{t+1}$ [4];
Associate each $x \in Y_t \cup F^{i^*}_{t+1}$ to its reference point $\mathrm{rp}(x) \in \mathcal{R}_p$ based on the smallest distance to the reference rays;
For each $r \in \mathcal{R}_p$, initialise $\rho_r := |\{x \in Y_t \mid \mathrm{rp}(x) = r\}|$;
Initialise $\tilde{F}^{i^*}_{t+1} := \emptyset$ and $R' := \mathcal{R}_p$;
while *true* **do**
    Determine $r_{\min} \in R'$ such that $\rho_{r_{\min}}$ is minimal (break ties randomly);
    Determine $x_{r_{\min}} \in F^{i^*}_{t+1} \setminus \tilde{F}^{i^*}_{t+1}$ which is associated with $r_{\min}$ and minimises the distance between $f^n(x_{r_{\min}})$ and the ray of $r_{\min}$ (break ties randomly);
    if $x_{r_{\min}}$ *exists* **then**
        Update $\tilde{F}^{i^*}_{t+1} := \tilde{F}^{i^*}_{t+1} \cup \{x_{r_{\min}}\}$;
        Update $\rho_{r_{\min}} := \rho_{r_{\min}} + 1$;
        if $|Y_t| + |\tilde{F}^{i^*}_{t+1}| = \mu$ **then return** $\tilde{F}^{i^*}_{t+1}$
    else Update $R' := R' \setminus \{r\}$

The first and last ranked individuals are always assigned an infinite crowding distance. The remaining individuals are then assigned the differences between the values of f_k of those ranked directly above and below the search point and divided by the difference between f_k of the first and last ranked. NSGA-II then takes r solutions from $F^{i^*}_{t+1}$ with the largest crowding distances from $F^{i^*}_{t+1}$ to complete P_{t+1} where ties are broken uniformly at random.

3 General Level-Based Theorems for Elitist Algorithms

We first give a general level-based theorem that applies to single- and multi-objective settings, before considering multi-objective functions in Sect. 4. The following result combines several extensions of the fitness-level method from the literature, including offspring populations from [31], stochastic domination results from [20] and tail bounds from [43].

Definition 4. *A partition $A_0, \ldots, A_k$ of the search space such that all sets A_i are non-empty is called* level partition. *For a population P denote by $\ell(P) := \max\{i \mid P \cap A_i \neq \emptyset\}$ the index of the best level set.*

Definition 5. *An algorithm $\mathcal{A}$ evolving populations $P_0, P_1, \ldots$ is called* level-monotone *w. r. t. a level partition $A_0, \ldots, A_k$ if, for all times t, $\ell(P_{t+1}) \geq \ell(P_t)$.*

For a sequence of sets $A_0, \ldots, A_k$ we use the shorthands $A_{>i} := \bigcup_{j>i} A_j$ and $A_{<i} := \bigcup_{j<i} A_j$, and likewise for $A_{\geq i} := A_i \cup A_{>i}$ and $A_{\leq i} := A_i \cup A_{<i}$.

Theorem 6. *Consider a level partition $A_0, \ldots, A_k$ and a level-monotone algorithm $\mathcal{A}$ evolving populations $P_0, P_1, \ldots$ of cardinality μ. Assume that $\mathcal{A}$ creates λ offspring by using some independent parent selection where a parent in $A_{\ell(P_t)}$*

is chosen with probability at least $|A_{\ell(P_t)} \cap P_t|/|P_t|$. *For* $0 \leq i \leq k-1$ *let* $s_i \in (0,1]$ *be a lower bound on the probability of creating a search point in* $A_{>\ell(P_t)}$ *when selecting a parent in* $A_{\ell(P_t)}$. *Let* T *denote the random number of evaluations made until the last level* A_k *is reached. Then the following statements hold.*

(1)

$$\mathrm{E}[T] \leq \lambda k + \mu \sum_{i=0}^{k-1} \frac{1}{s_i}.$$

(For $\lambda = 1$ *the summand* λk *can be dropped.)*

(2) T *is stochastically dominated by a sum of independent random variables:*

$$\lambda \sum_{i=0}^{k-1} \mathrm{Geom}(1-(1-s_i/\mu)^\lambda).$$

(3) For every $\delta \geq 0$,

$$\Pr\left(T > \lambda k + \mu \sum_{i=0}^{k-1} \frac{1}{s_i} + \delta\lambda\right) \leq e^{-(\delta/4)\min\{\delta/s,h\}}$$

where $h \coloneqq \min\{1-(1-s_i/\mu)^\lambda \mid 0 \leq i \leq k-1\}$ *and* $s \coloneqq \sum_{i=0}^{k-1} \frac{1}{(1-(1-s_i/\mu)^\lambda)^2}$.

Proof. The probability of selecting a search point from the current best level i and creating an offspring in a better level is at least s_i/μ. The probability of this happening during λ offspring creations is at least $1-(1-s_i/\mu)^\lambda$. The expected waiting time for this to happen is stochastically dominated by a geometric random variable with parameter $1-(1-s_i/\mu)^\lambda$. Multiplying by λ to account for λ offspring being created in one generation proves the second claim.

The first statement follows from the second one and the fact that the expectation of a geometric random variable with parameter p is $1/p$. Using $1-(1-p)^\lambda \geq \frac{p\lambda}{1+p\lambda}$ [1, Lemma 10], the expected number of *generations* on level i is at most

$$\frac{1}{1-(1-s_i/\mu)^\lambda} \leq \frac{1+s_i\lambda/\mu}{s_i\lambda/\mu} = 1 + \frac{\mu}{s_i\lambda}.$$

Multiplying by λ proves the claimed bound on the expectation. In the special case $\lambda = 1$ we have $\frac{1}{1-(1-s_i/\mu)^\lambda} = \frac{\mu}{s_i}$ and obtain a bound of $\sum_{i=0}^{k-1} \frac{1}{s_i}$ evaluations.

The third statement follows directly from Witt's tail bound for fitness levels [43] applied to amplified success probabilities. Here one step of the algorithm in Theorem 1 of [43] corresponds to one generation of $\mathcal{A}$ and the probability of improving the current level i is at least $1-(1-s_i/\mu)^\lambda$. Multiplying the time bound with λ yields a tail bound on the number of evaluations used by $\mathcal{A}$:

$$\Pr\left(T > \lambda \sum_{i=0}^{k-1} \frac{1}{1-(1-s_i/\mu)^\lambda} + \delta\lambda\right) \leq e^{-(\delta/4)\min\{\delta/s,h\}}.$$

Noting $\lambda \sum_{i=0}^{k-1} \frac{1}{1-(1-s_i/\mu)^\lambda} \leq \lambda k + \mu \sum_{i=0}^{k-1} \frac{1}{s_i}$ (see above) yields the claim. □

4 Applying Level-Based Theorems to Multi-objective EAs

We now show how the fitness-level method can be applied to multi-objective evolutionary algorithms.

Definition 7. *A level partition $A_0, \ldots, A_k$ is called* level partition respecting Pareto-dominance *if for all $i \in \{0, \ldots, k\}$ all search points in A_i are not weakly dominated by any search point in $A_{<i}$.*

Note that any set A_i in a level partition respecting Pareto-dominance may still contain search points x and y such that one dominates the other. This is intentional as it allows for a coarse-grained partition of the search space. However, for the final set A_k such a coarse-graining may not be desired if we are looking for Pareto-optimal search points. If A_k only contains mutually incomparable search points, all search points in A_k are Pareto-optimal.

Lemma 8. *Consider a level partition $A_0, \ldots, A_k$ respecting Pareto-dominance. If all search points in A_k are mutually incomparable then A_k only contains Pareto optima.*

Proof. Assume for a contradiction that A_k contains a search point x that is not Pareto optimal. Then there exists a search point y strictly dominating x. By Definition 7 and Pareto-dominance, y cannot be in $A_{<k}$. So $y \in A_k$, contradicting the assumption that all points in A_k are mutually incomparable. □

Level partitions can be carefully crafted or be defined in a canonical fashion. The following lemma shows three different ways: using the sum of fitness values, focusing on one objective i and ignoring all other objectives and applying non-dominated sorting to the whole search space.

Lemma 9. *(1) For $j \in \{0, \ldots, f^{sum}\}$ let $A_j := \{x \in \{0,1\}^n \mid f_1(x) + \ldots + f_m(x) = j\}$. Then the non-empty A_j form a level partition respecting Pareto-dominance.*

(2) For a fixed dimension $i \in \{1, \ldots, m\}$ and $j \in \{0, \ldots, f_i^{\max}\}$ let $B_j^i := \{x \in \{0,1\}^n \mid f_i(x) = j\}$. Then the non-empty sets from $B_0^i, B_1^i, \ldots, B_{f_i^{\max}}^i$ form a level partition respecting Pareto-dominance.

(3) Let $F_1, \ldots, F_k$ be the layers obtained when applying non-dominated sorting to the whole search space $\{0,1\}^n$. For $j \in \{1, \ldots, k\}$ let $A_j := F_{k-j}$. Then $A_0, \ldots, A_{k-1}$ form a level partition respecting Pareto-dominance.

Proof. (1): Let $j < i$ and suppose that $x \in A_i$ and $y \in A_j$. Then $\sum_{i=1}^m f_i(x) > \sum_{i=1}^m f_i(y)$. Suppose that $x \preceq y$. Then $f_i(x) \leq f_i(y)$ for every $i \in \{1, \ldots, m\}$ and therefore $\sum_{i=1}^m f_i(x) \leq \sum_{i=1}^m f_i(y)$, a contradiction.
(2): Let $i \in \{1, \ldots, m\}$ and j, ℓ with $j < \ell$. Suppose that $x \in B_\ell^i$ and $y \in B_j^i$, which implies $f_i(y) < f_i(x)$. If $x \preceq y$ then $f_i(x) \leq f_i(y)$, a contradiction.
(3): By the non-dominated sorting the $F_1, \ldots, F_k$ form a partition of $\{0,1\}^n$. Suppose some $x \in A_s = F_{k-s}$ weakly dominates a $y \in A_\ell = F_{k-\ell}$ for $s < \ell$,

i.e. x is in a worse layer than y. If $f(x) = f(y)$ then $s = \ell$, a contradiction (as individuals with equal fitness are put in the same layer). If $f(x) \neq f(y)$ then x strictly dominates y and hence, x is in a better layer than y, a contradiction. □

Example 10. Let $f := m$-LOTZ. Then $f_i^{\max} = 2n/m$ for $i \in [m]$ and hence $f^{\text{sum}} = 2n$. Additionally, for $i \in [m/2]$ we have that $\{f_{2i-1}(x) + f_{2i}(x) \mid x \in \{0,1\}^n\} = \{0, \ldots, 2n/m\} \setminus \{2n/m - 1\}$ since every block has length $2n/m$. Note that there is no $x \in \{0,1\}^n$ with $f_{2i-1}(x) + f_{2i}(x) = 2n/m - 1$, because block i of x has the form $1^\star 010^\star$ if x has second largest possible fitness value $f_{2i-1}(x) + f_{2i}(x)$. This implies $\{\sum_{i=1}^m f_i(x) \mid x \in \{0,1\}^n\} = \{0, \ldots, n-m, n-m+2, n-m+4, \ldots, n\}$ since the number of blocks is $m/2$. Hence there are $n+1-m/2$ levels $A_0, \ldots, A_{n-m}, A_{n-m+2}, \ldots, A_n$. Since the LeadingOnes- or TrailingZeros-value in every block can be any value in $\{0, \ldots, 2n/m\}$ there are $2n/m + 1$ levels $B_0, \ldots, B_{2n/m}$.

Example 11. Let $g := m$-COCZ. We have $g_i^{\max} = n/2 + n/m$ for $i \in [m]$ and hence $g^{\text{sum}} = mn/2 + n$. For $x \in \{0,1\}^n$ let $|x|_1^1$ be the number of ones in the first half of x. Then for $i \in \{1, \ldots, m/2\}$ and a given $\ell \in \{0, \ldots, n/2\}$ we have $\{g_{2i-1}(x) + g_{2i}(x) \mid x \in \{0,1\}^n, |x|_1^1 = \ell\} = \{2\ell + n/m\}$ and therefore $\{\sum_{i=1}^m g_i(x) \mid x \in \{0,1\}^n, |x|_1^1 = \ell\} = \{m\ell + n/2\}$. The latter implies $\{\sum_{i=1}^m g_i(x) \mid x \in \{0,1\}^n\} = \{m\ell + n/2 \mid \ell \in \{0, \ldots, n/2\}\}$ and hence there are $n/2 + 1$ many levels $A_{n/2}, A_{n/2+m}, \ldots, A_{n/2+mn/2}$. For $i \in [m]$ we see that $\{g_i(x) \mid x \in \{0,1\}^n\} = \{0, \ldots, n/2 + n/m\}$ (since $|x|_1^1 \in \{0, \ldots, n/2\}$ and every block in the left half of x has size at most n/m) which shows that there are $n/2 + n/m + 1$ many levels $B_0^i, \ldots, B_{n/2+n/m}^i$.

The next goal for this section is to show that the multiobjective algorithms (G)SEMO, NSGA-II and NSGA-III are level-monotone for level partitions respecting Pareto-dominance. In case of NSGA-II we use a *stable* sorting algorithm to sort each objective when computing the crowding distance, i.e. sort the j-th objective and then transfer this result to the $(j+1)$-th one by keeping identical elements in their original order. The following lemma shows operations under which the current level is maintained.

Lemma 12. *Let $P \subset \{0,1\}^n$ and suppose that P' results from P either by adding a search point from $\{0,1\}^n$ or by removing a search point from P that is weakly dominated by another element from P. Then $\ell(P) \leq \ell(P')$ with respect to* any *level partition respecting Pareto-dominance.*

Proof. Clearly, adding a search point does not decrease the level. Suppose that x is the search point removed from P and $y \neq x$ is a search point weakly dominating x. If $x \notin A_{\ell(P)}$ then clearly $\ell(P') = \ell(P)$. If $x \in A_{\ell(P)}$, by Definition 7 we have that $y \notin A_{<\ell(P)}$ and thus we must have $y \in A_{\ell(P)}$. Thus, $\ell(P) = \ell(P')$. □

Now we show that well known EMO algorithms like (G)SEMO, NSGA-II and NSGA-III are level-monotone given certain algorithmic parameter settings.

Lemma 13. *Let S be a maximum-cardinality set of mutually incomparable solutions and let m be the number of objectives. The following properties hold with respect to* every *level partition $A_0, \dots, A_k$ that respects Pareto-dominance.*

(1) (G)SEMO is level-monotone.
(2) Suppose that $m = 2$ and $\mu \geq 2|S|$. For any iteration t of NSGA-II we have $\ell(P_t) \leq \ell(R_t) = \ell(P_{t+1})$, i.e. NSGA-II is level-monotone.
(3) Suppose that $\mu \geq |S|$ and there are $p \geq 2m^{3/2} f_{\max}$ divisions along each objective to define the set $\mathcal{R}_p$ of reference points. For any iteration t of NSGA-III we have $\ell(P_t) \leq \ell(R_t) = \ell(P_{t+1})$, i.e. NSGA-III is level-monotone.

Lemma 13 applies to *every* level partition respecting Pareto-dominance. Thus, researchers wanting to apply our tool only need to verify that their level partition respects Pareto-dominance and (for NSGA-II/III) choose algorithmic parameters meeting the requirements from the lemma.

Proof of Lemma 13. (1): During one iteration of (G)SEMO only individuals are removed which are weakly dominated by the offspring x. By Lemma 12 (G)SEMO is level-monotone.

(2)/(3): Let P_t be the current population. Then a set Q_t of μ offspring is generated and $R_t := Q_t \cup P_t$ is computed. Let $F^1_{t+1}, \dots, F^s_{t+1}$ be the partition of R_t computed by non-dominated sorting. Then $\ell(P_t) \leq \ell(R_t) = \ell(F^1_{t+1})$ where the latter equality is due to Lemma 12 and the fact that every individual in $F^2_{t+1}, \dots, F^s_{t+1}$ is strictly dominated by a $z \in F^1_{t+1}$. Then we argue for both algorithms that for every $y \in F^1_{t+1}$ there is $z \in P_{t+1}$ with $f(y) = f(z)$ as follows.

For NSGA-II we follow [46]: for every fitness vector there are at most two search points covering that vector with positive crowding distance. Hence, there can be at most $2|S| \geq 2|\{f(x) \mid x \in F^1_{t+1}\}|$ many individuals in F^1_{t+1} with positive crowding distance and if $\mu \geq 2|S|$ then every individual from F^1_{t+1} with positive crowding distance is taken into P_{t+1}. Note also that for every fitness vector covered by an individual $x \in F^1_t$ there is at least one individual $x' \in F^1_{t+1}$ covering the same vector and having a positive crowding distance. We refer to [46] for details.

For NSGA-III we follow [36,41]: if there are $p \geq 2m^{3/2} f_{\max}$ divisions along each objective then one can show that two individuals from F^1_{t+1} with two distinct fitness vectors are associated with two distinct reference points. Since there are $\mu \geq |S| \geq |\{f(x) \mid x \in F^1_{t+1}\}| := s$ distinct individuals, there are also s reference points to which at least one individual is associated. Since $\mu \geq s$, at least one $x' \in F^1_{t+1}$ which is associated to the same reference point as x survives with $f(x) = f(x')$. We refer to [36] for details. Hence, $\ell(F^1_{t+1}) = \ell(P_{t+1})$. Together, we see $\ell(P_t) \leq \ell(P_{t+1})$. □

The following lemma improves on Lemma 13 by stating much looser algorithmic requirements for NSGA-II and NSGA-III, however for *specific* level partitions that only consider an arbitrary but fixed objective i. This means that an appropriate choice of the level partition can yield much looser algorithmic requirements (e.g., smaller populations) for reaching the Pareto front. This is an

important stepping stone for runtime analyses using populations of much smaller sizes than that of the Pareto front. Compact populations were studied in [15].

Lemma 14. *Let $i \in \{1, \dots, m\}$ and let $B_0^i, \dots, B_{f_i^{\max}}^i$ be defined by $B_j^i := \{x \in \{0,1\}^n \mid f_i(x) = j\}$. Then the following holds.*

(1) If $\mu \geq 2m$, then $\ell(R_t) = \ell(P_{t+1})$ for every iteration t of NSGA-II, i. e. it is level-monotone w. r. t. the level partition of non-empty sets of B_j^i. In addition, if $m = 2$ the condition $\mu \geq 2$ is sufficient.
(2) If $m = 2$ and $\mu \geq |\mathcal{R}_p| \geq 2$ (i.e. at least $p \geq 1$ divisions along each objective are used), then $\ell(R_t) = \ell(P_{t+1})$ for every iteration t of NSGA-III, i. e. it is level-monotone w. r. t. the level partition of non-empty sets of B_j^i.

Proof. (1): Note that there are at most $2m$ many individuals in F_{t+1}^1 with infinite crowding distance. Since $\mu \geq 2m$ they are all taken into P_{t+1}. Hence, there is an individual with maximum f_i-value in F_{t+1}^1 with infinite crowding distance. This individual is taken into P_{t+1}. Hence, $\ell(P_t) \leq \ell(R_t) = \ell(P_{t+1})$. Now suppose $m = 2$. Since we use a stable sorting algorithm, the sorting $(x^1, x^2, \dots, x^K)$ of the individuals from F_{t+1}^1 with respect to the first objective induces the sorting $(x^K, \dots, x^1)$ with respect to the second objective. The reason is because all individuals in F_{t+1} are mutually incomparable, hence if $f_1(x^i) < f_1(x^j)$ for $i < j$ we must have $f_2(x^i) > f_2(x^j)$ and the other way round. Note that this does not hold for $m \geq 3$ in general. Hence, there are at most 2 individuals x^1, x^K with infinite crowding distance and they both will transfer to P_{t+1} owing to $\mu \geq 2$.

(2): If $|F_{t+1}^1| \leq \mu$ then the claim holds since there is an $x \in F_{t+1}^1$ with maximum f_i-value. So suppose that $|F_{t+1}^1| > \mu$ and fix $x \in F_{t+1}^1$ with maximum f_i-value. Since we deal with two objectives and all individuals with distinct fitness in F_{t+1}^1 are incomparable, x has also minimal (f_{3-i})-value among all individuals from F_{t+1}^1 (otherwise x dominates another individual in F_{t+1}^1). We also have $x_{3-i} = y_{3-i}^{\min}$ and therefore $f_{3-i}^n(x) = 0$. Thus, x is on the reference ray g coinciding with the $x_{(3-i)}$-axis and hence it is associated with the reference point v where $v_{(3-j)} = 1$ if $j = i$ and $v_j = 0$ otherwise. Hence, the individuals from F_{t+1}^1 with the smallest angle to the reference ray g are on g and hence have the same fitness value as x. (If $y \neq x$ and y is on g then either $x \prec y$ or $y \prec x$.) Since $\mu \geq |\mathcal{R}_p|$, for every reference point v at least one individual associated with v with the smallest angle to v survives. Hence, an individual x' with the same fitness value as x survives, which implies $\ell(P_t) \leq \ell(R_t) = \ell(P_{t+1})$. □

An immediate consequence of Lemma 14 is that for two objectives, NSGA-II and NSGA-III are able to find a Pareto-optimal search point even if μ is small.

Theorem 15. *Consider NSGA-II with $\mu \geq 2$ or NSGA-III with $\mu \geq |\mathcal{R}_p| \geq 2$ (i.e. at least $p \geq 1$ divisions along each objective are used) on $f :=$ 2-LOTZ. Let T denote the number of evaluations made until a Pareto-optimal solution is created. Then the following statements hold.*

(1) $E[T] \leq \mu n + e\mu n^2$.
(2) T is stochastically dominated by $\mu \sum_{i=0}^{n-1} Geom(1/(en+1))$.

(3) For every $\delta > 0$,

$$\Pr\left(T > \mu n + e\mu n^2 + \delta\mu\right) \leq e^{-(\delta/4)\min\{\delta/(n(en+1)^2),1/(en+1)\}}$$

which is $e^{-\Omega(n)}$ if $\delta = \Omega(n^2)$.

Proof. By Lemma 14 NSGA-II and NSGA-III are level-monotone w. r. t. the level partition defined by $B_j^1 := \{x \in \{0,1\}^n \mid f_1(x) = j\}$ for $j \in \{0,\dots,n\}$. Note that $B_n^1 = \{1^n\}$ consists only of one Pareto-optimal search point and therefore a Pareto-optimal search point is found if B_n^1 is reached. Let $\ell \in \{0,\dots,n-1\}$ be the current best level. Since one $x \in P_t$ in B_ℓ^1 is chosen uniformly at random, i.e. with probability $1/\mu$, we can apply Theorem 6 on $B_0^1,\dots,B_n^1$.

The current level is improved if a specific zero bit of an individual in B_ℓ^j is flipped while the remaining bits are kept unchanged. Hence, we may choose $s_i = 1/(en)$, and, by Theorem 6, $\mathrm{E}[T] \leq \mu n + \mu\sum_{i=0}^{n-1} en = \mu n + \mu en^2$ since μ offspring are created in one iteration. We also have that T is stochastically dominated by $\mu\sum_{i=0}^{n-1}\mathrm{Geom}(1-(1-1/(en\mu))^\mu)$. For $r := 1/(en\mu)$ we obtain

$$1-(1-r)^\mu \geq \frac{\mu r}{1+\mu r} = \frac{1}{1/(\mu r)+1} = \frac{1}{en+1},$$

and hence T is also stochastically dominated by $\mu\sum_{i=0}^{n-1}\mathrm{Geom}(1/(en+1))$. For the third statement, $s := \sum_{i=0}^{n-1}\frac{1}{(1-(1-1/(en\mu))^\mu)^2} \leq \sum_{i=0}^{n-1}\frac{1}{(1/(en+1))^2} = n(en+1)^2$ and $h := \min\{1-(1-s_i/\mu)^\mu \mid 0 \leq i \leq n-1\} = 1-(1-1/(en\mu))^\mu \geq 1/(en+1)$. Thus, for every $\delta > 0$

$$\Pr\left(T > \mu n + \mu en^2 + \delta\mu\right) \leq e^{-(\delta/4)\min\{\delta/(n(en+1)^2),1/(en+1)\}}.$$

□

Theorem 16. *Consider NSGA-II with $\mu \geq 2$ or NSGA-III with $\mu \geq |\mathcal{R}_p| \geq 2$ (i.e. at least $p \geq 1$ divisions along each objective are used) on $f :=$ 2-COCZ. Let T denote the random number of evaluations made until a Pareto-optimal solution is created. Then we get, denoting by $H(n)$ the n-th harmonic number,*

(1) $E[T] \leq \mu n + e\mu n H(n)$.
(2) T is stochastically dominated by $\mu\sum_{i=0}^{n-1} Geom(i/(en+i))$.
(3) For every $\delta > 0$

$$\Pr\left(T > \mu n + \mu enH(n) + \delta\mu\right) \leq e^{-\Omega(\delta\min\{\delta/n^2,1/n\})}.$$

Proof. We use the same level partition and the same initial arguments as in the proof of Theorem 15. The level is improved if a zero bit of an individual in B_i^1 is flipped while the remaining bits are kept unchanged. Hence, $s_i = (n-i)/(en)$, and by Theorem 6 we obtain $\mathrm{E}[T] \leq \mu n + \mu\sum_{i=0}^{n-1} en/(n-i) = \mu n + \mu\sum_{i=1}^{n} en/i = \mu n + \mu enH(n)$. We also have that T is stochastically dominated by $\mu\sum_{i=1}^{n}\mathrm{Geom}(1-(1-r_i)^\mu)$ where $r_i := i/(en\mu)$. Since $1-(1-r_i)^\mu \geq \frac{r_i\mu}{1+r_i\mu} = \frac{i}{en+i}$, T is also stochastically dominated by $\mu\sum_{i=1}^{n}\mathrm{Geom}(i/(en+i))$.

For the third statement, $s := \sum_{i=1}^{n} \frac{1}{(1-(1-i/(en\mu))^{\mu})^2} \leq \sum_{i=1}^{n} \frac{1}{(i/(en+i))^2} = \sum_{i=1}^{n} (en/i+1)^2 \leq \sum_{i=1}^{n} ((e+1)n/i)^2 \leq (e+1)^2 n^2 \sum_{i=1}^{\infty} 1/i^2 = (e+1)^2 n^2 \pi^2/6$ and $h = \min\{1-(1-r_i/\mu)^{\mu} \mid 0 \leq i \leq n-1\} = 1-(1-1/(en\mu))^{\mu} \geq 1/(en+1)$. For every $\delta > 0$

$$\Pr\left(T > \mu n + \mu enH(n) + \delta\mu\right) \leq e^{-(\delta/4)\min\{\delta/((e+1)^2 n^2 \pi^2/6), 1/(en+1)\}}.$$

□

5 Conclusions

We have shown that a general level-based theorem can be applied to all level partitions respecting Pareto-dominance. It immediately yields upper bounds on expected runtimes, stochastic domination statements as well as tail bounds. To apply this method, one only has to prove that the level partition respects Pareto-dominance (this is often an easy task and for straightforward partitions it is implied by Lemma 9), and the requirements for NSGA-II and NSGA-III guaranteeing level-monotonicity can be taken directly from Lemma 13. We also showed that when only seeking to find a Pareto-optimal solution, smaller population sizes are sufficient. We hope that Lemma 14 paves the way for analysing state-of-the-art MOEAs with population sizes smaller than the Pareto front.

References

1. Badkobeh, G., Lehre, P.K., Sudholt, D.: Black-box complexity of parallel search with distributed populations. In: Proceedings of the Foundations of Genetic Algorithms (FOGA 2015), pp. 3–15. ACM Press (2015)
2. Beume, N., Naujoks, B., Emmerich, M.T.M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**(3), 1653–1669 (2007)
3. Bian, C., Qian, C.: Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In: PPSN 2022. LNCS, vol. 13399, pp. 428–441. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14721-0_30
4. Blank, J., Deb, K., Roy, P.C.: Investigating the normalization procedure of NSGA-III. In: Deb, K., et al. (eds.) EMO 2019. LNCS, vol. 11411, pp. 229–240. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12598-1_19
5. Cerf, S., Doerr, B., Hebras, B., Kahane, Y., Wietheger, S.: The first proven performance guarantees for the non-dominated sorting genetic algorithm II (NSGA-II) on a combinatorial optimization problem. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5522–5530. ijcai.org (2023)
6. Corus, D., Lehre, P.K.: Theory driven design of efficient genetic algorithms for a classical graph problem. In: Amodeo, L., Talbi, E.-G., Yalaoui, F. (eds.) Recent Developments in Metaheuristics. ORSIS, vol. 62, pp. 125–140. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-58253-5_8
7. Corus, D., Dang, D.-C., Eremeev, A.V., Lehre, P.K.: Level-based analysis of genetic algorithms and other search processes. IEEE Trans. Evolution. Comput. **22**(5), 707–719 (2018)

8. Dang, D.-C., Lehre, P. K.: Efficient optimisation of noisy fitness functions with population-based evolutionary algorithms. In: Proceedings of the Foundations of Genetic Algorithms (FOGA 2015), pp. 62–68. ACM Press (2015)
9. Dang, D.-C., Lehre, P.K.: Simplified runtime analysis of estimation of distribution algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 15), pp. 513–518. ACM Press (2015)
10. Dang, D.-C., Lehre, P.K.: Runtime analysis of non-elitist populations: from classical optimisation to partial information. Algorithmica **75**(3), 428–461 (2016)
11. Dang, D.-C. , Eremeev, A.V., Lehre, P.K., Qin, X.: Fast non-elitist evolutionary algorithms with power-law ranking selection. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2022), pp. 1372–1380. ACM Press (2022)
12. Dang, D.-C., Opris, A., Salehi, B., Sudholt, D.: A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. Proc. AAAI Conf. Artif. Intell. **37**(10), 12390–12398 (2023)
13. Dang, D.-C., Opris, A., Salehi, B., Sudholt, D.: Analysing the robustness of NSGA-II under noise. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2023), pp. 642–651. ACM Press (2023)
14. Dang, D.-C., Opris, A., Sudholt, D.: Crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. Artif. Intell. **330**, 104098 (2024)
15. Dang, D.-C., Opris, A., Sudholt, D.: Illustrating the efficiency of popular evolutionary multi-objective algorithms using runtime analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2024) (2024). To appear
16. Das, I., Dennis, J.E.: Normal-boundary intersection: a new method for generating the pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optimiz. **8**(3), 631–657 (1998)
17. Deb, K.: NSGA-II source code in C, version 1.1.6 (2011). https://www.egr.msu.edu/~kdeb/codes/nsga2/nsga2-gnuplot-v1.1.6.tar.gz. Accessed 15 Aug 2022
18. 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)
19. 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)
20. Doerr, B.: Analyzing randomized search heuristics via stochastic domination. Theoret. Comput. Sci. **773**, 115–137 (2019)
21. Doerr, B., Kötzing, T.: Multiplicative up-drift. Algorithmica **83**(10), 3017–3058 (2021)
22. Doerr, B., Kötzing, T.: Lower bounds from fitness levels made easy. Algorithmica **86**(2), 367–395 (2024)
23. Doerr, B., Qu, Z.: A first runtime analysis of the NSGA-II on a multimodal problem. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) PPSN 2022. LNCS, vol. 13399, pp. 399–412. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14721-0_28
24. Doerr, B., Qu, Z.: A first runtime analysis of the NSGA-II on a multimodal problem. IEEE Trans. Evol. Comput. **27**(5), 1288–1297 (2023)
25. Doerr, B., Qu, Z.: From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2023, pp. 12408–12416. AAAI Press (2023)

26. Doerr, B., Qu, Z.: Runtime analysis for the NSGA-II: provable speed-ups from crossover. In: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2023, pp. 12399–12407. AAAI Press (2023)
27. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. Evol. Comput. **18**(3), 335–356 (2010)
28. Gutjahr, W.J., Sebastiani, G.: Runtime analysis of ant colony optimization with best-so-far reinforcement. Methodol. Comput. Appl. Probab. **10**(3), 409–433 (2008)
29. He, J., Zhou, Y.: Drift analysis with fitness levels for elitist evolutionary algorithms. In: Evolutionary Computation, pp. 1–25 (2024). To appear
30. Lässig, J., Sudholt, D.: General upper bounds on the running time of parallel evolutionary algorithms. Evol. Comput. **22**(3), 405–437 (2014)
31. Lässig, J., Sudholt, D.: Analysis of speedups in parallel evolutionary algorithms and (1+λ) EAs for combinatorial optimization. Theoret. Comput. Sci. **551**, 66–83 (2014)
32. 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)
33. Lehre, P.K.: Fitness-levels for non-elitist populations. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011), pp. 2075–2082. ACM Press (2011)
34. Mühlenbein H.: How genetic algorithms really work: mutation and hillclimbing. In: Parallel Problem Solving from Nature (PPSN II), pp. 15–26 (1992)
35. Neumann, F., Sudholt, D., Witt, C.: Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intell. **3**(1), 35–68 (2009)
36. Opris, A., Dang, D.-C., Neumann, F., Sudholt, D.: Runtime analyses of NSGA-III on many-objective problems. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2024) (2024). To appear. Preprint available at https://arxiv.org/abs/2404.11433
37. Rudolph, G.: Local performance measures: genetic algorithms. In: Bäck, T., Fogel, D.B., Michalewicz, Z., (eds.) Handbook of Evolutionary Computation, vol. B2, no. 4, pp. 20–27. IOP Publishing and Oxford University Press, Bristol, New York (1997)
38. Sudholt, D.: A new method for lower bounds on the running time of evolutionary algorithms. IEEE Trans. Evol. Comput. **17**(3), 418–435 (2013)
39. Sudholt, D., Witt, C.: Runtime analysis of a binary particle swarm optimizer. Theoret. Comput. Sci. **411**(21), 2084–2100 (2010)
40. Wegener, I.: Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In: Evolutionary Optimization, pp. 349–369. Kluwer (2002)
41. Wietheger, S., Doerr, B.: A mathematical runtime analysis of the non-dominated sorting genetic algorithm III (NSGA-III). In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI 2023), pp. 5657–5665. ijcai.org (2023)
42. Witt, C.: Runtime analysis of the (μ+1) EA on simple pseudo-Boolean functions. Evol. Comput. **14**(1), 65–86 (2006)
43. Witt, C.: Fitness levels with tail bounds for the analysis of randomized search heuristics. Inf. Process. Lett. **114**(1), 38–41 (2014)
44. Zheng, W., Doerr, B.: Better approximation guarantees for the NSGA-II by using the current crowding distance. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2022), pp. 611–619. ACM Press (2022)

45. Zheng, W., Doerr, B.: Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for many objectives. IEEE Trans. Evolution. Comput. (2023). To appear
46. Zheng, W., Liu, Y., Doerr, B.: A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II). In: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2022, pp. 10408–10416. AAAI Press (2022)
47. Zhou, D., Luo, D., Lu, R., Han, Z.: The use of tail inequalities on the probable computational time of randomized search heuristics. Theoret. Comput. Sci. **436**, 106–117 (2012)

Runtime Analysis for State-of-the-Art Multi-objective Evolutionary Algorithms on the Subset Selection Problem

Renzhong Deng[1], Weijie Zheng[1(✉)], Mingfeng Li[1], Jie Liu[1], and Benjamin Doerr[2]

[1] School of Computer Science and Technology, International Research Institute for Artificial Intelligence, Harbin Institute of Technology, Shenzhen, China
{dengrenzhong,limingfeng}@stu.hit.edu.cn, {zhengweijie,jieliu}@hit.edu.cn
[2] Laboratoire d'Informatique (LIX), École Polytechnique, CNRS, Institut Polytechnique de Paris, Palaiseau, France
doerr@lix.polytechnique.fr

Abstract. In the last few years, the mathematical runtime analysis of randomized search heuristics has made a huge step forward by developing the methods to analyze the most prominent multi-objective evolutionary algorithms (MOEAs) as opposed to previously only simplistic algorithms. These results confirmed that many previous results extend to state-of-the-art MOEAs, but also showed that algorithms like the NSGA-II can have unexpected difficulties on problems easily solved by simple MOEAs. We continue this line of research by analyzing how the NSGA-II and the SMS-EMOA (also with a recently proposed stochastic population update) solve the NP-hard subset selection problem. For these two state-of-the-art algorithms, we prove performance guarantees that agree with those previously shown for the POSS algorithm, a variant of the simplistic GSEMO, namely that they compute $(1 - e^{-\gamma})$-approximate solutions in expected time $O(k^2 n)$. Our experiments confirm these findings. This work is the first runtime analysis of state-of-the-art MOEAs for the subset selection problem, and also the first runtime analysis of SMS-EMOA on a combinatorial problem.

Keywords: Subset selection · Multi-objective optimization · Runtime analysis · Theory

1 Introduction

Many optimization problems have conflicting objectives. Multi-objective evolutionary algorithms (MOEAs) are widely used to solve such problems. For example, the NSGA-II was first proposed in 2002 by Deb et al. [8], and has received more than 50000 citations (according to Google scholar). The SMS-EMOA was first proposed in 2007 by Beume et al. [2], and now receives more than 2,000 citations (according to Google scholar). The runtime analysis of the simplistic

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 264–279, 2024.
https://doi.org/10.1007/978-3-031-70071-2_17

MOEAs (like SEMO and GSEMO) dates back to the early 2000s [14,17]. Only in the last few years, mathematical runtime analyses of the above prominent algorithms were conducted. Zheng et al. [33] (see [28] for the complete journal version) gave the first runtime analysis of the NSGA-II, and proved that the expected runtime of the NSGA-II to cover the full Pareto front is $O(n^2 \ln n)$ for OneMinMax and is $O(n^3)$ for LOTZ. We note that both complexities are the same as for the (G)SEMO [14,15,17]. The first runtime analysis of the SMS-EMOA was conducted by Bian et al. [4], where a runtime of $O(n^{k+1})$ for bi-objective OJZJ [12] with gap size k is proved. We also note that it is the same asymptotic complexity as the GSEMO [30] and the NSGA-II [10].

The above runtime results on the state-of-the-art MOEAs quickly attracted considerable attention. The effect of crossover [24] and heavy-tailed mutation [30] witnessed in the GSEMO was also proven for the NSGA-II [3,6,10,11]. For the combinatorial bi-objective minimum spanning tree problem, Cerf et al. [5] showed that the NSGA-II has the same performance as the GSEMO. Besides, the SMS-EMOA reaches the same runtime on DLTB as the GSEMO [32]. The above works confirm that many previous results for GSEMO extend to the NSGA-II and the SMS-EMOA.

However, Zheng and Doerr [29] proved that from three objectives on, the NSGA-II with reasonable population size needs at least exponential runtime to cover the full Pareto front of the m-objective OneMinMax problem, in contrast to the polynomial runtime of the GSEMO. That is, we should be careful when extending the results of the simplistic MOEAs to the state-of-the-art MOEAs. We note that these difficulties where not seen with the NSGA-III [20,26].

The subset selection problem is a classic NP-hard optimization problem. The aim is to select a subset of size no more than k from a total set of size n to optimize a given objective function f. Qian et al. [25] modelled the above constrained single-objective problem into a bi-objective one, and used the GSEMO as the multi-objective solver to obtain a set of solutions. The best feasible one is then selected as the output. They call this the Pareto optimization for subset selection, POSS, and proved that it can find a solution with $(1 - e^{-\gamma})$ approximation ratio, which is the same ratio as the greedy algorithm [7]. They also proved that the expected runtime to reach such an approximation is $2ek^2n$. Considering real-word scenarios and optimization efficiency in the subset selection problem, parallel, noisy and distributed POSS algorithms have been gradually proposed [21–23].

Our Contributions: In this work, we continue this line of research, and discuss whether the results above for the GSEMO extend to the NSGA-II and the SMS-EMOA. We will prove that both the NSGA-II and the SMS-EMOA are able to achieve the same approximation ratio. We will also prove that the expected runtime to reach such approximation is $8ek^2n$ for the NSGA-II, and $2ek^2n$ for the SMS-EMOA. Both are asymptotically same as for the GSEMO.

We will also discuss the impact of a recently proposed stochastic population update [4] for the SMS-EMOA, which allowed super-polynomial speed-ups for the OJZJ problem. We will prove that this variant computes $(1 - e^{-\gamma})$-

approximate solutions within $ek(4k+1)n$ iterations in expectation. Consequently, we were not able to prove a strong speed-up as done in [4] for OJZJ. Our experiments do not indicate such a speed-up either.

Extensive experiments are conducted on the sparse regression as an application of the subset selection. The experimental results show that when using the same function evaluation times $2ek^2n$, NSGA-II, SMS-EMOA, and SMS-EMOA with stochastic population update reach a similar approximation ratio to the POSS. We also show that the impact of stochastic population update is limited.

2 Preliminaries

The subset selection problem is to select a subset (with a size less than a predefined value) of a given set to minimize (or maxmize) a given objective function.

Definition 1 (Subset Selection). *Let $n, k \in \mathbb{N}$ and $U = \{X_1, \ldots, X_n\}$. The subset selection problem (minimizing $f : 2^U \to \mathbb{R}$) is*

$$\min_{S \subseteq U} f(S), \text{s.t. } |S| \leq k. \tag{1}$$

Since the subset selection is an NP-hard problem, in the following, we focus on how well a solution approximates the optimum.

2.1 Characteristics

We now list some definitions and characteristics of the subset selection problem that will be discussed and used in our proof.

Definition 2 (Submodularity Ratio [7]). *Let f be a non-negative set function. The submodularity ratio of f w.r.t. a set U and a parameter $k \geq 1$ is*

$$\gamma_{U,k}(f) = \min_{L \subseteq U, S: |S| \leq k, S \cap L = \emptyset} \frac{\sum_{x \in S}(f(L \cup \{x\}) - f(L))}{f(L \cup S) - f(L)}.$$

Definition 3 (Monotonity). *Let $n \in \mathbb{N}$ and $U = \{X_1, \ldots, X_n\}$. We call that $f : 2^U \to \mathbb{R}$* monotonically increases *if $f(S) \leq f(S')$ holds for all $S \subseteq S' \subseteq U$. Similarly, we call that f* monotonically decreases *if $f(S) \geq f(S')$ holds for all $S \subseteq S' \subseteq U$.*

From the above definitions, we obviously know that $\gamma_{U,k}(f)$ monotonically decreases w.r.t. U, see the following corollary.

Corollary 4. *Let f be a non-negative set function, $k \geq 1$, and $\gamma_{U,k}(f)$ as defined in Definition 2. Then $\gamma_{U,k}(f)$ monotonically decreases w.r.t. U.*

In the following, we will just write $\gamma_{U,k}$ for $\gamma_{U,k}(f)$ without ambiguity. For a non-negative and monotonically increasing set function, we have the following lemma that shows that one can always add an item so that the function improvement is proportional to the distance between the optimal function value and the current function value. Note that this lemma is from [25, Lemma 1], which is deduced from the proof in [7, Theorem 3.2].

Lemma 5 ([25]). *Let $f \geq 0$ be monotonically increasing. Let* OPT *be the optimal function value for the subset problem (1). Then for any $S \subseteq U$, there is one item $v \in U \setminus S$ such that*

$$f(S \cup \{v\}) - f(S) \geq \frac{\gamma_{S,k}}{k}(\mathrm{OPT} - f(S)).$$

2.2 Pareto Optimization

Different from designing algorithms to solve the above constrained single-objective problem, Qian et al. [25] first formulated (1) into maximizing the following bi-objective problem. In this bi-objective function, the first objective is f in (1) but penalizes it as $-\infty$ when $|S| \geq 2k$ when the constraint is violated too much. $-|S|$ is set as the second objective.

Definition 6 (Bi-objective Model [25]). *Let $n, k \in \mathbb{N}$ and $U = \{X_1, \ldots, X_n\}$. The bi-objective function $g = (g_1(S), g_2(S)) : 2^U \to \mathbb{R}^2$ is defined by*

$$g_1(S) = \begin{cases} -\infty, & \text{if } |S| \geq 2k, \\ f(S), & \text{else}, \end{cases}, g_2(S) = -|S|. \tag{2}$$

After formulating (1) into the bi-objective (2), Qian et al. [25] then used the simplistic GSEMO as the multi-objective solver. The GSEMO will provide a set of solutions when it is terminated. Among these solutions, the best feasible one that maximizes f will finally be selected. The above procedure is called *Pareto optimization for subset selection, POSS*, see Algorithm 1.

Algorithm 1. POSS

Input: The set $U = \{X_1, \ldots, X_n\}$, a given criterion f, an integer parameter $k \in [1..n]$, the number of iterations T and an isolation function $I : \{0,1\}^n \to \mathbb{R}$

Output: A subset of U with at most k items

1: Generate bi-objective g (Definition 6) from f
2: Let $\boldsymbol{s} = \{0\}^n$ and $P = \{\boldsymbol{s}\}$. Let $t = 0$
3: **while** $t < T$ **do**
4: Select $\boldsymbol{s}$ from P uniformly at random
5: Generate $\boldsymbol{s'}$ form $\boldsymbol{s}$ by flipping each bit of $\boldsymbol{s}$ with probability $1/n$
6: **if** $\nexists \boldsymbol{z} \in P$ such that $I(\boldsymbol{z}) = I(\boldsymbol{s'})$ and $((g_1(\boldsymbol{z}) < g_1(\boldsymbol{s'}) \wedge g_2(\boldsymbol{z}) \leq g_2(\boldsymbol{s'}))$ or $(g_1(\boldsymbol{z}) \leq g_1(\boldsymbol{s'}) \wedge g_2(\boldsymbol{z}) < g_2(\boldsymbol{s'}))$ **then**
7: $Q = \{\boldsymbol{z} \in P \mid I(\boldsymbol{z}) = I(\boldsymbol{s'}) \wedge g_1(\boldsymbol{s'}) \leq g_1(\boldsymbol{z}) \wedge g_2(\boldsymbol{s'}) \leq g_2(\boldsymbol{z})\}$
8: $P = (P \setminus Q) \cup \{\boldsymbol{s'}\}$
9: **end if**
10: $t = t + 1$
11: **end while**
12: **return** $\arg\min_{\boldsymbol{s} \in P, |\boldsymbol{s}| \leq k} f(\boldsymbol{s})$

Here we use a binary vector $\boldsymbol{s} \in \{0,1\}^n$ to represent the subset $S \subseteq V$, where $s_i = 1$ means that the i-th elements in V is in subset S and $s_i = 0$ otherwise. The isolation function I [27] is introduced in the POSS, and two solutions can be compared if and only if they have the same isolation function value. When I is set to a constant, it can be ignored since every solution has the same I value. In the following discussions, we will set I as a constant as in [25], and just omit it in the discussion of the NSGA-II and the SMS-EMOA in Sects. 3 and 4.

Here are some definitions related to the bi-objective optimization that will be used in this paper. For any bi-objective function $g = (g_1, g_2)$, and for any two solutions $\boldsymbol{x}, \boldsymbol{y}$, if $g_i(\boldsymbol{x}) \geq g_i(\boldsymbol{y})$ for $i = 1, 2$, then we say $\boldsymbol{x}$ *weakly dominates* $\boldsymbol{y}$, denoted as $\boldsymbol{x} \succeq \boldsymbol{y}$. If $\boldsymbol{x} \succeq \boldsymbol{y}$ and at least one of the above inequalities is strict, then we say $\boldsymbol{x}$ *dominates* $\boldsymbol{y}$, denoted as $\boldsymbol{x} \succ \boldsymbol{y}$.

Although it is mentioned but without any proof in [25] for (2), we extract the following lemma here with a formal proof as we believe it will be useful for future usage. In the following, we will use $[a..b]$ to represent a set of integers $\{a, \ldots, b\}$ with $a \leq b$.

Lemma 7. *Consider any set of solutions P such that $\boldsymbol{r} \not\preceq \boldsymbol{s}$ w.r.t. (2) for all $\boldsymbol{r}, \boldsymbol{s} \in P$ with $\boldsymbol{r} \neq \boldsymbol{s}$. Then $|P| \leq 2k$.*

3 NSGA-II Efficiently Approximates

As introduced in Sect. 1, the state-of-the-art MOEAs like the NSGA-II can have unexpected difficulties on problems easily solved by simple GSEMO. For the subset selection problem, the existing literature only considered the GSEMO as the multiobjective solver in the POSS, and showed the nice approximation ability and efficient runtime for reaching such an approximation. Whether the above difficulty will also be witnessed on the subset selection, for state-of-the-art MOEAs like NSGA-II and SMS-EMOA, is still unknown. This section will prove that the NSGA-II has the same approximation ability as the GSEMO, and needs the same asymptotical runtime as the GSEMO. The performance of SMS-EMOA will be discussed in the next section.

3.1 NSGA-II

The NSGA-II algorithm is first proposed in [8] and uses random initialization. Note that the GSEMO used in POSS [25] starts with an initial individual of 0^n. Different from the GSEMO, the NSGA-II starts with a set of solutions (population) with size N. Following this, we also consider the NSGA-II starts with a population with all 0^ns for the subset selection problem. This choice aligns with the previous works [21–23,25] analyzing randomized search heuristics for the subset selection problem, and 0^n is an obvious feasible solution, hence a natural starting point, in particular, for a problem where a solution with few, namely k, ones is asked for. We note that the MOEAs regarded in this work in particular optimize the single objectives. Consequently, by minimizing the second objective, they would also quickly find the solution 0^n. From this point on,

they would work with a population that contains at least one copy of 0^n, and all our proofs would apply to this situation equally well as to an initial population consisting of all-zeros strings only.

Other steps are the same as the original NGSA-II. In each loop, an offspring population Q_t with the same size N will be generated. By using fast-non-dominated-sort [8], the joint population of the parent population and the offspring population $R_t = P_t \cup Q_t$ is divided into $F_1, F_2, \ldots$, where no individual in F_i will be dominated by individuals in F_j with $j \geq i$. The parent population of the next iteration is determined by $F_1, \ldots, F_{i^*-1}$ and some individuals in F_{i^*}, where $\sum_{i=1}^{i^*-1} |F_i| < N$ and $\sum_{i=1}^{i^*} |F_i| \geq N$. Some individuals in F_{i^*} are determined by using crowding-distance-assignment [8]. The crowding distance of an individual in F_{i^*} is the sum of the crowding distances over all objectives. For each objective, after sorting the objective values of all individuals in F_{i^*} in ascending order, the individuals with largest and smallest objective values have an infinite crowding distance, and the crowding distance for other individuals is the difference in objective values between their left and right neighbors normalized by the largest and smallest objective values. $N - \sum_{i=1}^{i^*-1} |F_i|$ individuals in F_{i^*} with largest crowding distance will enter the next generation population. The loop ends when a certain termination condition is reached. For details, see Algorithm 2. Note that as discussed in Sect. 2 we will omit the isolation function in POSS as [25] set it as a constant and takes no effect.

Algorithm 2. NSGA-II

1: Generate the initial population $P_0 = \{x_1, x_2, \ldots, x_N\}$ with $x_i \in \{0\}^n, i = 1, 2, \ldots, N$
2: **for** $t = 0, 1, 2, \ldots,$ **do**
3: Generate the offspring population Q_t with size N
4: Use fast-non-dominated-sort() [8] to divide $R_t = P_t \cup Q_t$ into $F_1, F_2, \ldots$
5: Find $i^* \geq 1$ such that $\sum_{i=1}^{i^*-1} |F_i| < N$ and $\sum_{i=1}^{i^*} |F_i| \geq N$
6: Use crowding-distance-assignment() [8] to separately calculate the crowding distance of each individual in $F_1, \ldots, F_{i^*}$
7: Let $\tilde{F}_{i^*}$ be the $N - \sum_{i=1}^{i^*-1} |F_i|$ individuals in F_{i^*} with largest crowding distance, chosen at random in case of a tie
8: $P_{t+1} = (\bigcup_{i=1}^{i^*-1} F_i) \cup \tilde{F}_{i^*}$
9: **end for**

3.2 Performance Analysis

As discussed in Sect. 2, the subset selection is an NP-hard problem so that the optimum is difficult to obtain. Hence, we will evaluate the performance of the algorithm by the approximation ratio to show how well it approximates the optimum. On the other hand, we also concern how efficient the algorithm is to obtain such approximation ratio. That is, we will also discuss the runtime (the number of function evaluations or the number of iterations) as commonly used in the evolutionary theory [1,9,16,19,34].

Recall in Sect. 2 that we use a binary bitstring $\boldsymbol{s} \in \{0,1\}^n$ to denote the subset $S \subseteq U$ (item i in U is selected into S if $s_i = 1$). Let $j \in [0..k], \gamma_{\min} = \min_{\boldsymbol{s}:|\boldsymbol{s}|_1=k-1} \gamma_{S,k}$. Since all individuals in P_0 are $\{0\}^n$, let $J_{\max}$ denote the maximum value of j such that there exists a solution $\boldsymbol{s}$ in the population P with $|\boldsymbol{s}|_1 \leq j$ and $f(\boldsymbol{s}) \geq \left(1 - (1 - \frac{\gamma_{\min}}{k})^j\right) \cdot \mathrm{OPT}$, that is,

$$J_{\max} = \max \left\{ j \in [0..k] \mid \exists \boldsymbol{s} \in P : |\boldsymbol{s}|_1 \leq j \wedge f(\boldsymbol{s}) \geq \left(1 - \left(1 - \frac{\gamma_{\min}}{k}\right)^j\right) \cdot \mathrm{OPT} \right\}.$$

When $J_{\max} = k$, it means that $\exists \boldsymbol{s} \in P$ such that $|\boldsymbol{s}|_1 \leq k$ and

$$f(\boldsymbol{s}) \geq (1 - (1 - \gamma_{\min}/k)^k) \cdot \mathrm{OPT} \geq (1 - e^{-\gamma_{\min}}) \cdot \mathrm{OPT}\,.$$

That is, a feasible solution with approximation ratio of $(1 - e^{-\gamma_{\min}})$ is obtained for the original subset selection problem. Hence, in our following proofs, the key relies on whether it can increase the value of $J_{\max}$ to k, and relies on how many fitness evaluations the algorithm needs to reach $J_{\max} = k$ if it is reachable.

We first prove in the following lemma that if the population size is large enough, the value of $J_{\max}$ will not decrease. The key of the proof is that at least one solution corresponding to the $J_{\max}$ or the improved $J_{\max}$ will survive in the iterations when the population size $N \geq 8k$.

Lemma 8. *Consider using the NSGA-II algorithm with population size $N \geq 8k$ to optimize g defined in Definition 6. Then $J_{\max}$ will not decrease as the population evolves.*

The following theorem shows approximation ratio of the NSGA-II and the runtime to reach such an approximation. The key of the proof is to consider a solution starting from 0^n (i.e., $J_{\max} = 0$), and greedily flip a specific 0 bit each time to increase $J_{\max}$ from 0 to k and then achieve such an approximation ratio.

Theorem 9. *Consider optimizing g defined in Definition 6 via the NSGA-II with standard bit-wise mutation once to each parent. If the population size N is at least $8k$ and is of order $\Theta(k)$. then the expected runtime to find a solution $\boldsymbol{s}$ (which denote the subset S of U) with $|\boldsymbol{s}|_1 \leq k$ and $f(\boldsymbol{s}) \geq (1 - e^{-\gamma_{\min}}) \cdot \mathrm{OPT}$ is ekn iterations and $ekNn$ fitness evaluations, where $\gamma_{\min} = \min_{\boldsymbol{s}:|\boldsymbol{s}|_1=k-1} \gamma_{S,k}$.*

Theorem 9 shows that the NSGA-II can reach the same approximation ratio as the GSEMO in POSS [25], and that the runtime to reach such an approximation is $O(k^2 n)$, the same asymptotic complexity as the POSS [25].

4 SMS-EMOA Also Efficiently Approximates

The above section shows that the performance of the POSS on the subset selection problem extends to the NSGA-II successfully. This section will discuss the SMS-EMOA, another state-of-the-art MOEAs that is recently analyzed. We will show that the SMS-EMOA also achieves the same approximation ratio and

requires the same asymptotic runtime as the POSS. Besides, we will also analyze the impact of the stochastic population update strategy, which is proposed and shown to result in a super-polynomial speed-up for the bi-objective JUMP [4], but is also proven with a reduced speed-up for many-objective ones. Our theoretical result only shows the same performance as the original one, and the experiments in the next section do not indicate a considerable speed-up either.

4.1 SMS-EMOA

The SMS-EMOA algorithm is proposed in [2], and can be regarded as a variant of the steady-state NSGA-II. Steady-state NSGA-II here means that only one offspring individual is generated in each iteration instead of N offspring individuals in the original NSGA-II. The SMS-EMOA is such a variant but replaces the crowding distance by the hypervolume indicator in the survival selection. Formally speaking, after dividing the combined parent and offspring population into several fronts, an individual in F_{i^*} with the smallest $\Delta_r(\boldsymbol{z}, F_{i^*})$ will be removed. Here $\Delta_r(\boldsymbol{z}, F_{i^*}) = \mathrm{HV}_r(F_{i^*}) - \mathrm{HV}_r(F_{i^*} \setminus \{\boldsymbol{z}\})$, $\mathrm{HV}_r(S) = \mathcal{L}\left(\bigcup_{\boldsymbol{u}\in S} H_{\boldsymbol{u},r}\right)$ represents the hypervolume value of S w.r.t. the reference point r, and $\mathcal{L}$ represents the Lebesgue measure. The reference point r is usually dominated by all points in the solution space. Similar to the GSEMO and the NSGA-II in Sects. 2 and 3, the SMS-EMOA for the subset selection also initialize the set of the solutions with all $\{0\}^n$s. We also set the isolation function to be constant and ignore its effect as in [25]. See Algorithm 3 for more details.

Algorithm 3. SMS-EMOA

1: Generate the initial population $P_0 = \{x_1, x_2, \ldots, x_\mu\}$ with $x_i \in \{0\}^n, i = 1, 2, \ldots, \mu$
2: **for** $t = 0, 1, 2, \ldots,$ **do**
3: Select a solution $\boldsymbol{x}$ uniformly at random from P_t
4: Generate $\boldsymbol{x'}$ by flipping each bit of $\boldsymbol{x}$ independently with probability $1/n$
5: Use fast-non-dominated-sort() [8] to divide $R_t = P_t \cup \{\boldsymbol{x'}\}$ into $F_1, ..., F_{i^*}$
6: Calculate $\Delta_r(\boldsymbol{z}, F_{i^*})$ for all $\boldsymbol{z} \in F_{i^*}$ and find $D = \arg\min_{z\in F_{i^*}} \Delta_r(\boldsymbol{z}, F_{i^*})$
7: Uniformly at random pick $\boldsymbol{z'} \in D$ and $P_{t+1} = R_t \setminus \{\boldsymbol{z'}\}$
8: **end for**

Performance of SMS-EMOA. The first runtime analysis of the SMS-EMOA is only recently conducted by Bian et al. [4] on the bi-objective OJZJ benchmark in 2023, and the other runtime analysis is for the many-objective JUMP function in [31]. There is no theoretical analysis of the SMS-EMOA on the subset selection problem till now. Here we conduct such analysis.

Similar to the discussion in Sect. 3.2, here we discuss the approximation ratio and the runtime to reach such approximation as the performance of the algorithm.

With similar proof ideas, we analyze the performance of the SMS-EMOA by regarding whether $J_{\max}$ can increase to k (the approximation ratio of $1 - e^{-\gamma_{\min}}$ will be reached) and how efficiently the algorithm reaches $J_{\max} = k$. The following lemma shows that when the population size is at least the upper bound of the size of the non-dominated solutions (Lemma 7), $J_{\max}$ will not decrease. The key of the proof is the same as Lemma 8.

Lemma 10. *Consider using the SMS-EMOA algorithm with population size $\mu \geq 2k$ to optimize g defined in Definition 6. Then $J_{\max}$ will not decrease in the iterations.*

Now we establish the performance analysis of the SMS-EMOA. The key of the proof is the same as Theorem 9.

Theorem 11. *Consider optimizing g defined in Definition 6 via the SMS-EMOA. If the population size μ is at least $2k$ and is of order $\Theta(k)$, then the expected runtime to find a solution $\boldsymbol{s}$ (which denote the subset S of U) with $|\boldsymbol{s}|_1 \leq k$ and $f(\boldsymbol{s}) \geq (1 - e^{-\gamma_{\min}}) \cdot \mathrm{OPT}$ is $ek\mu n$ fitness evaluations, where $\gamma_{\min} = \min_{\boldsymbol{s}:|\boldsymbol{s}|_1=k-1} \gamma_{S,k}$.*

Theorem 11 shows that the SMS-EMOA can also reach the same approximation ratio within the same runtime of $O(k^2 n)$, as the GSEMO in POSS [25]. We note that it is the first runtime analysis of the SMS-EMOA on a combinatorial problem.

4.2 SMS-EMOA with Stochastic Population Update

Bian et al. [4] proposed a stochastic population update strategy for the SMS-EMOA, aiming to make some usage of the inferior solutions. The stochastic population update happens after the offspring population is generated. Compared to the original SMS-EMOA , this strategy only picks half of the solutions in the combined parent and offspring population for the survival selection. Thus, some inferior solutions but with useful information will have some chance of being kept. We denote its variant as SMS-EMOA-SPU, see Algorithm 4. Note that the only difference from in Algorithm 3 locates in Steps 6 and 7.

Performance of SMS-EMOA-SPU. With the stochastic population update strategy, Bian et al. [4] proved a super-polynomial speed-up compared to the original SMS-EMOA on the bi-objective OJZJ. However, a recent work [31] proved that the impact of this strategy will drastically reduce and even vanish when the number of objectives increases. Hence, whether or when this strategy helps for a specific problem is still unclear. Here, we consider the performance (as discussed in previous sections) of the SMS-EMOA with this strategy on the subset selection.

The process of using the SMS-EMOA-SPU to optimize the bi-objective sparse regression problem is similar to the SMS-EMOA. We consider whether $J_{\max}$ can increase to k and how efficiently the algorithm reaches $J_{\max} = k$. Similar to

Algorithm 4. SMS-EMOA-SPU

1: Generate the initial population $P_0 = \{x_1, x_2, \ldots, x_\mu\}$ with $x_i \in \{0\}^n, i = 1, 2, \ldots, \mu$
2: **for** $t = 0, 1, 2, \ldots,$ **do**
3: Select a solution $\boldsymbol{x}$ uniformly at random from P_t
4: Generate $\boldsymbol{x'}$ by flipping each bit of $\boldsymbol{x}$ independently with probability $1/n$
5: $R_t \leftarrow P_t \cup \{\boldsymbol{x'}\}$
6: $R'_t \leftarrow \lfloor(\mu + 1)/2\rfloor$ solutions uniformly and randomly selected from R_t without replacement
7: Use fast-non-dominated-sort() [8] to divide R'_t into $F_1, \ldots, F_{i^*}$
8: Calculate $\Delta_r(\boldsymbol{z}, F_{i^*})$ for all $\boldsymbol{z} \in F_{i^*}$ and find $D = \arg\min_{z \in F_{i^*}} \Delta_r(\boldsymbol{z}, F_{i^*})$
9: Uniformly at random pick $\boldsymbol{z'} \in D$ and $P_{t+1} = R_t \setminus \{\boldsymbol{z'}\}$
10: **end for**

Lemma 10, $J_{\max}$ will not decrease in the evolving process of the SMS-EMOA-SPU with population size more than $4k$.

Lemma 12. *Consider using the SMS-EMOA-SPU algorithm with population size $\mu \geq 4k+1$ to optimize g defined in Definition 6. Then $J_{\max}$ will not decrease in the iterations.*

Now we prove the performance of the SMS-EMOA with the stochastic population update strategy.

Theorem 13. *Consider optimizing g defined in Definition 6 via the SMS-EMOA-SPU. If the population size μ is at least $4k + 1$ and is of order $\Theta(k)$, then the expected runtime to find a solution $\boldsymbol{s}$ (which denotes the subset S of U) with $|\boldsymbol{s}|_1 \leq k$ and $f(\boldsymbol{s}) \geq (1 - e^{-\gamma_{\min}}) \cdot \mathrm{OPT}$ is $ek\mu n$ fitness evaluations, where $\gamma_{\min} = \min_{\boldsymbol{s}:|\boldsymbol{s}|_1=k-1} \gamma_{S,k}$.*

Comparing Theorem 13 with Theorem 11, we see that for obtaining an approximation ratio of $1 - e^{-\gamma_{\min}}$, both algorithms need the same asymptotic runtime. Noting that both results give the upper bounds, we cannot say that the stochastic population update cannot help the SMS-EMOA on the subset selection. Some lower bound is needed. However, the experiments in the next section show that the speed-up of this strategy is questionable, and we believe that we may not put effort for obtaining such lower bounds.

5 Experiments

The above sections theoretically show that the good performance of the POSS on the subset selection successfully extends to the state-of-the-art MOEAs, the NSGA-II and the SMS-EMOA. Although the same approximation ratio and runtime to reach such an approximation also extends to the SMS-EMOA with the stochastic population update, whether a good speed-up exists is still unknown. This section will use the sparse regression as an application of the subset selection problem to experimentally verify the theoretical results we obtained, and to experimentally check whether a practical speed-up can be achieved with the stochastic population update.

5.1 Sparse Regression as an Application

In this section, we will follow the same line as in [25] to analyze the performance of MOEAs on the subset selection in its application of sparse regression. Sparse regression problem [18] aims to obtain a linear model that depends on as small as possible number of features. How to select such features is a subset selection problem.

Definition 14 (Sparse Regression). *Let $n \in \mathbb{N}$ and $U = \{1, \ldots, n\}$ be set of feature indices. The sparse regression is to find a subset of feature indices with size at most k that minimizes the mean squared error, that is,*

$$\begin{aligned} &\min_{S \subseteq U} \mathrm{MSE}(S) \\ &\text{s.t. } |S| \leq k \end{aligned} \tag{3}$$

with $\mathrm{MSE}_S = \min_{\alpha \in \mathbb{R}^{|S|}} E\left[\left(\boldsymbol{Z} - \sum_{i \in S} \alpha_i \boldsymbol{X_i}\right)^2\right]$, *where* $\boldsymbol{X_i}, i = 1, \ldots, n$ *is the vector of observed values for i-th feature, and* $\boldsymbol{Z}$ *is the predictor.*

Following [25], in our experiments, we replace MSE_S by the squared multiple correlation

$$R^2_{\boldsymbol{Z},S} = (\mathrm{Var}(\boldsymbol{Z}) - \mathrm{MSE}_S)/\mathrm{Var}(\boldsymbol{Z}). \tag{4}$$

Here are our settings.

Datasets. Table 1 collects nine datasets[1] from [25] that are utilized for the experiments.

Algorithms. POSS, NSGA-II, SMS-EMOA, SMS-EMOA-SPU. The original POSS with the GSEMO as the multiobjective solver is conducted as the baseline of the performance. We do not consider other algorithms for comparison as our motivation for this work is to test whether the results of the POSS successfully extend to the NSGA-II and the SMS-EMOA. NSGA-II, SMS-EMOA, and SMS-EMOA-SPU are the algorithms that we have theoretically analyzed in previous sections.

Accuracy Baseline. Since the subset selection is NP-hard, we just use the exhaustive enumeration as the optimizer to obtain the (approximate) optimum. We denote, in our experiments, the best solution via the enumeration as OPT. For the first three relatively smaller datasets (housing, enuite2001, svmguide3), the obtained values are optimal. For other datasets, after running a long time without obtaining results, we evaluated the precision achieved by the algorithms under a given number of fitness evaluations. The maximum number of fitness evaluations is set to be $\lfloor 2ek^2n \rfloor$ which is the theoretical upper bound for POSS algorithm suggested in [25].

[1] The data sets are from http://archive.ics.uci.edu/ml/ and https://www.openml.org/ and http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. Some binary classification data are used for regression.

Table 1. The datasets.

Dataset	#inst	#feat	Dataset	#inst	#feat	Dataset	#inst	#feat
housing	506	13	ionosphere	351	34	coil2000	9000	86
eunite2001	367	16	sonar	208	60	mushrooms	8124	112
svmguide3	1284	21	triazines	186	60	clean1	476	166

Efficiency. For the first three datasets where the optimum is reachable, we use the number of fitness evaluations that the algorithm uses to reach the optimum as the measure for the efficiency. Note that it is widely used as the runtime in the evolutionary theory community [1,9,16,19,34].

Others. To assess each method on each data set, we repeated each experiment 100 times. The data set was randomly and evenly split into a training set and a test set.

5.2 Experimental Results and Analyses

Accuracy and Generalization Performance. For the first three datasets, all algorithms reach the optimum. Hence, Table 2 only records the training R^2 values for the remaining datasets. Note that merely regarding the training accuracy is also a subset selection problem. From Table 2, it is indicated that the NSGA-II, SMS-EMOA, and SMS-EMOA-SPU reach a similar approximation ratio to the POSS (using GSEMO) in optimizing sparse regression problems under a given number of fitness evaluations. Table 3 collects the generalization performance of the algorithms on ionosphere, sonar, and triazines datasets, as in [25]. Similarly, a t-test with a significance level of 0.05 indicates that there is no significant difference among the four algorithms. The reason for conducting statistical tests on the data is to eliminate the influence of randomness on the experimental results as much as possible.

Efficiency. Recall that for the first three datasets, all algorithms obtained the optimum. Here, we further collected the number of function evaluations when such an optimum is reached for the first time. These results will help to check the runtime we theoretically obtained in the previous sections. Recall that we proved the same asymptotic runtime of the NSGA-II, SMS-EMOA, and SMS-EMOA-SPU as the GSEMO, and that the constants for these three are $8, 2, 4$ and 2 for the GSEMO [25]. From Table 4 and Fig. 1, we saw similar rankings to the above constants. That is, NSGA-II needs the largest number of function evaluations, then SMS-EMOA-SPU. POSS and SMS-EMOA are similar, and need the smallest number of function evaluations. We also note that we do not see a speed-up when SMS-EMOA uses the stochastic population update.

Summary of Experimental Results. The experimental results verify our theoretical findings that the NSGA-II, SMS-EMOA, and SMS-EMOA-SPU approximate the optimum of the subset selection as well as the POSS. The stochastic population update does not speed up the SMS-EMOA on the subset selection.

Table 2. The training R^2 values (mean ± std.) of the four algorithms on 6 data sets for $k = 8$ with the same number of function evaluations which is set to be $\lfloor 2ek^2n \rfloor$. We performed Kolmogorov-Smirnov tests for normality on all experimental data, and the results indicate that the experimental data conform to a normal distribution. We demonstrated through a t-test with significance level 0.05 that four evolutionary algorithms achieve the same performance (without significant differences) across these datasets, with the exception of the mushrooms data set in which there is significant difference between the POSS and the SMS-EMOA.

Dataset	POSS	NSGA-II	SMS-EMOA	SMS-EMOA-SPU
ionosphere	0.5988 ± 0.0402	0.5984 ± 0.0403	0.5987 ± 0.0403	0.5988 ± 0.0404
sonar	0.5389 ± 0.0447	0.5396 ± 0.0438	0.5395 ± 0.0433	0.5392 ± 0.0437
triazines	0.4269 ± 0.0986	0.4255 ± 0.1053	0.4250 ± 0.1052	0.4280 ± 0.0982
mushrooms	0.9916 ± 0.0016	0.9915 ± 0.0016	0.9911 ± 0.0019	0.9916 ± 0.0016
clean1	0.4354 ± 0.0259	0.4331 ± 0.0289	0.4354 ± 0.0281	0.4349 ± 0.0282
coil2000	0.0614 ± 0.0460	0.0632 ± 0.0421	0.0632 ± 0.0421	0.0632 ± 0.0421

Table 3. The test R^2 value (mean ± std.) of the four algorithms on 3 data sets for $k = 8$. We also conducted a t-test on the experimental data with significance level 0.05. The results indicated that there is no significant difference in the generalization performance among the four algorithms (no significant differences were observed).

Dataset	POSS	NSGA-II	SMS-EMOA	SMS-EMOA-SPU
ionosphere	0.5029 ± 0.0516	0.5037 ± 0.0507	0.5006 ± 0.0533	0.5033 ± 0.0493
sonar	0.3169 ± 0.0546	0.3203 ± 0.0546	0.3235 ± 0.0517	0.3161 ± 0.0547
triazines	0.2264 ± 0.1142	0.2299 ± 0.1191	0.2178 ± 0.1201	0.2262 ± 0.1205

Table 4. The number of functions evaluations of reaching OPT (mean ± std.) of the four algorithms on 3 data sets for $k = 8$ and we eliminated the five best and five worst data points from the experiments.

Dataset	POSS	NSGA-II	SMS-EMOA	SMS-EMOA-SPU
eunite2001	917 ± 428	3000 ± 997	1368 ± 949	1702 ± 690
housing	522 ± 247	1924 ± 444	577 ± 220	942 ± 305
svmguide3	4379 ± 8921	7528 ± 13173	2433 ± 2268	3598 ± 4095

The results demonstrate that these more complex algorithms do not outperform simpler ones. It is evident that in practical scenarios, opting for these state-of-the-art algorithms is not advisable. The absence of notable speed-up resulting from the introduction of stochastic population updates can be attributed to the fact that, in the OJZJ problem, this approach enables the retention of solutions with inferior objective function values but closer proximity to certain elusive special Pareto optimal solutions. However, for the subset selection problem, the existence of such distinctive solutions is not guaranteed, thus hindering the acceleration observed in the OJZJ problem.

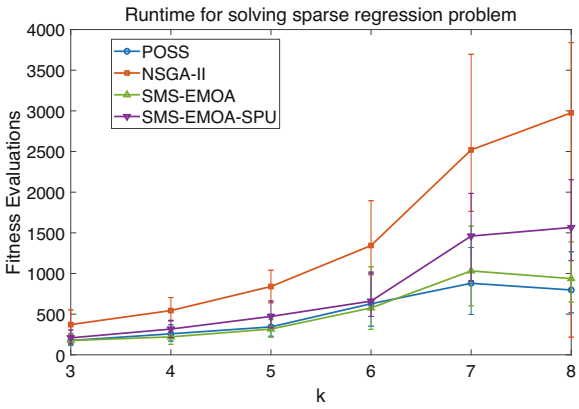


Fig. 1. The median number of fitness evaluations of POSS, NSGA-II, SMS-EMOA and SMS-EMOA-SPU for solving sparse regression problem on enuite2001 data set for $k \in \{3, 4, 5, 6, 7, 8\}$ in 100 independent runs, with error bars set to the first and third quartiles.

6 Conclusion

In this paper, we proved that on the subset selection problem, the approximation ability and the running time guarantee of $O(k^2 n)$ obtained by the previous simplistic GSEMO (i.e., POSS) can be extended to the state-of-the-art MOEAs such as NSGA-II, SMS-EMOA, and SMS-EMOA with the stochastic population update. Our experiments on the sparse regression problems verified the above findings and indicated that no speed-up is reached for the stochastic population update. We successfully conducted the first runtime analysis of state-of-the-art MOEAs on the subset selection problem and the first runtime analysis of SMS-EMOA on the combinatorial problem.

Our experiments revealed that the NSGA-II requires the highest number of function evaluations, which is likely due to its $(N + N)$-strategy. It would therefore be valuable to investigate a steady-state version of NSGA-II [13], i.e., the $(N + 1)$-version. The only difference between this steady-state NSGA-II and the SMS-EMOA would be the secondary selection criterion. Exploring how this steady-state NSGA-II performs in comparison to SMS-EMOA, particularly in terms of efficiency and effectiveness on subset selection problems, will provide deeper insights into the relative advantages of these algorithms.

Acknowledgements. This work was supported by National Natural Science Foundation of China (Grant No. 62350710797, 62306086), Science, Technology and Innovation Commission of Shenzhen Municipality (Grant No. GXWD20220818191018001), and Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019A1515110177). This research benefited from the support of the FMJH Program Gaspard Monge for optimization and operations research and their interactions with data science.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Auger, A., Doerr, B. (eds.): Theory of Randomized Search Heuristics. World Scientific Publishing (2011)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: multiobjective selection based on dominated hypervolume. Eur. J. Oper. Res. **181**, 1653–1669 (2007)
3. Bian, C., Qian, C.: Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) PPSN XVII, pp. 428–441. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14721-0_30
4. Bian, C., Zhou, Y., Li, M., Qian, C.: Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In: International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5513–5521. ijcai.org (2023)
5. Cerf, S., Doerr, B., Hebras, B., Kahane, J., Wietheger, S.: The first proven performance guarantees for the non-dominated sorting genetic algorithm II (NSGA-II) on a combinatorial optimization problem. In: International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5522–5530. ijcai.org (2023)
6. Dang, D.C., Opris, A., Salehi, B., Sudholt, D.: A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. In: Conference on Artificial Intelligence, AAAI 2023, pp. 12390–12398. AAAI Press (2023)
7. Das, A., Kempe, D.: Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In: International Conference on Machine Learning, ICML 2011, pp. 1057–1064. ACM (2011)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**, 182–197 (2002)
9. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation—Recent Developments in Discrete Optimization. Springer, Cham (2020)
10. Doerr, B., Qu, Z.: A first runtime analysis of the NSGA-II on a multimodal problem. IEEE Trans. Evol. Comput. **27**, 1288–1297 (2023)
11. Doerr, B., Qu, Z.: Runtime analysis for the NSGA-II: provable speed-ups from crossover. In: Conference on Artificial Intelligence, AAAI 2023, pp. 12399–12407. AAAI Press (2023)
12. Doerr, B., Zheng, W.: Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In: Conference on Artificial Intelligence, AAAI 2021, pp. 12293–12301. AAAI Press (2021)
13. Durillo, J.J., Nebro, A.J., Luna, F., Alba, E.: On the effect of the steady-state selection scheme in multi-objective genetic algorithms. In: Evolutionary Multi-criterion Optimization: 5th International Conference, EMO 2009, Nantes, 7–10 April 2009. Proceedings 5, pp. 183–197. Springer (2009)
14. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Congress on Evolutionary Computation, CEC 2003, pp. 1918–1925. IEEE (2003)
15. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. Evol. Comput. **18**, 335–356 (2010)

16. Jansen, T.: Analyzing Evolutionary Algorithms – The Computer Science Perspective. Springer, Heidelberg (2013)
17. Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. IEEE Trans. Evol. Comput. **8**, 170–182 (2004)
18. Miller, A.: Subset Selection in Regression. CRC Press (2002)
19. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity. Springer, Heidelberg (2010)
20. Opris, A., Dang, D.C., Neumann, F., Sudholt, D.: Runtime analyses of NSGA-III on many-objective problems. In: Genetic and Evolutionary Computation Conference, GECCO 2024. ACM (2024), to appear
21. Qian, C., Li, G., Feng, C., Tang, K.: Distributed pareto optimization for subset selection. In: Lang, J. (ed.) International Joint Conference on Artificial Intelligence, IJCAI 2018, pp. 1492–1498. ijcai.org (2018)
22. Qian, C., Shi, J., Yu, Y., Tang, K., Zhou, Z.: Parallel pareto optimization for subset selection. In: International Joint Conference on Artificial Intelligence, IJCAI 2016, pp. 1939–1945. IJCAI/AAAI Press (2016)
23. Qian, C., Shi, J., Yu, Y., Tang, K., Zhou, Z.: Optimizing ratio of monotone set functions. In: International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 2606–2612. ijcai.org (2017)
24. Qian, C., Yu, Y., Zhou, Z.: An analysis on recombination in multi-objective evolutionary optimization. Artif. Intell. **204**, 99–119 (2013)
25. Qian, C., Yu, Y., Zhou, Z.H.: Subset selection by pareto optimization. In: Advances in Neural Information Processing Systems, NIPS 2015, vol. 28. Curran Associates, Inc. (2015)
26. Wietheger, S., Doerr, B.: A mathematical runtime analysis of the Non-dominated Sorting Genetic Algorithm III (NSGA-III). In: International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5657–5665. ijcai.org (2023)
27. Yu, Y., Yao, X., Zhou, Z.H.: On the approximation ability of evolutionary optimization with application to minimum set cover. Artif. Intell. **180**, 20–33 (2012)
28. Zheng, W., Doerr, B.: Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). Artif. Intell. **325**, 104016 (2023)
29. Zheng, W., Doerr, B.: Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for many objectives. IEEE Trans. Evolution. Comput. (2023). In press. https://doi.org/10.1109/TEVC.2023.3320278
30. Zheng, W., Doerr, B.: Theoretical analyses of multiobjective evolutionary algorithms on multimodal objectives. Evol. Comput. **31**, 337–373 (2023)
31. Zheng, W., Doerr, B.: Runtime analysis of the SMS-EMOA for many-objective optimization. In: Conference on Artificial Intelligence, AAAI 2024, pp. 20874–20882. AAAI Press (2024)
32. Zheng, W., Li, M., Deng, R., Doerr, B.: How to use the metropolis algorithm for multi-objective optimization? In: Conference on Artificial Intelligence, AAAI 2024, pp. 20883–20891. AAAI Press (2024)
33. Zheng, W., Liu, Y., Doerr, B.: A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II). In: Conference on Artificial Intelligence, AAAI 2022, pp. 10408–10416. AAAI Press (2022)
34. Zhou, Z.H., Yu, Y., Qian, C.: Evolutionary Learning: Advances in Theories and Algorithms. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-5956-9

When Does the Time-Linkage Property Help Optimization by Evolutionary Algorithms?

Mingfeng Li[1], Weijie Zheng[1(✉)], Wen Xie[1], Ao Sun[1], and Xin Yao[2]

[1] School of Computer Science and Technology, International Research Institute for Artificial Intelligence, Harbin Institute of Technology, Shenzhen, China
zhengweijie@hit.edu.cn

[2] Lingnan University, Hong Kong SAR, China

Abstract. Recent theoretical works show that the time-linkage property challenges evolutionary algorithms to optimize. Here we consider three positive circumstances and give the first runtime analyses to show that the time-linkage property can also help the optimization of evolutionary algorithms. The problem is easier to optimize if the time-linkage property changes the optimal function value to an easy-to-reach one. We construct a time-linkage variant of the $\textsc{Cliff}_d$ problem with this feature and prove that conditional on an event that happens with $\Omega(1)$ probability, the $(1+1)$ EA reaches the optimum in expected $O(n \ln n)$ iterations. It is much better than the expected runtime of $\Theta(n^d)$ for the original $\textsc{Cliff}_d$. If the time-linkage property does not change the optimal function value but enlarges the optimal solution set, the problem is also possible to be easier to optimize. We construct another time-linkage variant of the $\textsc{Cliff}_d$ problem with this feature, and also prove an expected runtime of $O(n \ln n)$ (conditional on an event happening with $\Omega(1)$ probability), compared with the expected runtime of $\Omega(n^{d-2})$ for the corresponding problem without the time-linkage property. Even if the time-linkage property neither changes the optimal function value nor the optimal solution set, it is still possible to ease this problem if the intermediate solution, from which the optimum is easier to reach, is more prone to be maintained. We construct a time-linkage variant of the Jump problem, and proved that the expected runtime is reduced from $O(n^k)$ to $O(n^{k-1})$. Our experiments also verify the above theoretical findings.

Keywords: Time-linkage property · Evolutionary algorithms

1 Introduction

Many real-world optimization problems have the time-linkage property, that is, the objective function depends on both the current solutions and the historical ones [2]. However, as already pointed out in the first paper [2] that introduced this property into the evolutionary computation, the time-linkage property can

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 280–294, 2024.
https://doi.org/10.1007/978-3-031-70071-2_18

turn a problem into a deceptive one and then makes the problem hard to solve. The theoretical runtime analysis of the evolutionary algorithms on time-linkage problems recently started and also confirmed the search difficulties that this property brings [14,15,18]. OneMax is arguably the easiest benchmark analyzed in the evolutionary theory community, and the $(1+1)$ EA and other well-analyzed evolutionary algorithms reach the optimum in expected runtime of $O(n \ln n)$ [4,11,12]. Zheng, Chen, and Yao [15] constructed a time-linkage version of OneMax where the first dimension of the previous solution takes the time-linkage effect. They proved that for this benchmark, with probability $1 - o(1)$, randomized local search and the $(1+1)$ EA cannot find the optimum. They also proved that this challenge can be largely overcome by introducing a non-trivial parent population with its size large enough. Zheng et al. [18] proved that a $1-o(1)$ probability of not reaching the global optimum still exists for the $(1+\lambda)$ EA where the non-trivial offspring population is introduced, and they also proved that the non-elitism will help. Yang and Zhou [14] considered the mutiobjective optimization and constructed a time-linkage version of COCZ (a bi-objective counterpart of OneMax). They also proved the difficulty encountered by the GSEMO (the multiobjective counterpart of the $(1+1)$ EA). Till now, all existing theoretical works show that the time-linkage property turns an easy problem into a hard one.

Here, we will consider three positive circumstances and give the first runtime analyses to show that the time-linkage property can also help the optimization by evolutionary algorithms. We first consider the circumstance where the problem becomes easier as the time-linkage property changes the optimal function value to an easy-to-reach one. We take the $\textsc{Cliff}_d$ problem as an example. $\textsc{Cliff}_d$ is regarded as a not-easy-to-solve benchmark in the community as the $(1+1)$ EA needs expected $\Theta(n^d)$ runtime to reach its optimum due to the cliff region to jump [9]. We will construct $\textsc{Cliff}_{dTL}$, a time-linkage variant of $\textsc{Cliff}_d$ that changes the optimal function value and moves the original optimal solution from 1^n (after the cliff region) to solutions that can be reached easily without jumping. We will prove that for $\textsc{Cliff}_{dTL}$ the expected runtime for the $(1+1)$ EA is only $O(n \ln n)$, conditional on an event that happens with $\Omega(1)$ probability (Theorem 1).

We then consider the circumstance in which the problem becomes easier as the introduction of the time-linkage property enlarges the optimal solution set but keeps the optimal function value unchanged. We will still take the $\textsc{Cliff}_d$ problem (slightly modifying its constant component (from $1/2$ to 1) which requires expected runtime of $\Omega(n^{d-2})$ for the $(1+1)$ EA) as an example. The constructed time-linkage variant $\textsc{Cliff}'_{dTL}$ will have a larger optimal solution set that includes easy-to-reach solutions. We will also prove an expected runtime of $O(n \ln n)$ for the $(1+1)$ EA, conditional on an event happening with $\Omega(1)$ probability (Theorem 3).

The last circumstance in which the problem becomes easier that we will consider is that some easy-to-reach intermediate solution becomes prone to be maintained even if both the optimal function value and optimal solution set are

unchanged. Here, we take another not-easy-to-solve benchmark JUMP (which requires jumping across a gap) as an example. The constructed time-linkage variant JUMP_{kTL} makes a gap point near the optimum be possible to be maintained, which makes the optimum easier to be reached. We will prove that the expected runtime of the $(1+1)$ EA will be reduced from $O(n^k)$ to $O(n^{k-1})$ (Theorem 4).

Our experimental results also verify the above theoretical findings.

2 Preliminaries

Similar to $\text{ONEMAX}_{(0,1^n)}$, this work only considers the time-linkage effect of the first dimension of the last step on a pseudo-Boolean problem. Formally, we consider to maximize the time-linkage pseudo-Boolean problem $f : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$

$$f(x^{t-1}, x^t) = g(x_1^{t-1}, x^t)$$

for two consecutive $x^{t-1} = (x_1^{t-1}, \ldots, x_n^{t-1})$, $x^t = (x_1^t, \ldots, x_n^t)$ and the pseudo-Boolean problem $g : \{0,1\} \times \{0,1\}^n \to \mathbb{R}$.

The $(1+1)$ EA is the simplest evolutionary algorithm and widely analyzed in the evolutionary theory community. Hence, we will use the $(1+1)$ EA to demonstrate whether the time-linkage property will ease the optimization of the algorithm. Same as in [15], we take Algorithm 1 as the time-linkage $(1+1)$ EA to analyze. This time-linkage $(1+1)$ EA is the same as the original $(1+1)$ EA except that here the fitness evaluation requires the value of stored historical solutions.

Algorithm 1. The $(1+1)$ EA to maximize the time-linkage problem f for two consecutive time steps

1: Generate the random initial two generations $x^0 = (x_1^0, \ldots, x_n^0)$ and $x^1 = (x_1^1, \ldots, x_n^1)$
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Generate $\tilde{x}^t$ via independently flipping each bit of x^t with probability of $1/n$
4: **if** $f(x^t, \tilde{x}^t) \geq f(x^{t-1}, x^t)$ **then**
5: $(x^t, x^{t+1}) = (x^t, \tilde{x}^t)$
6: **else**
7: $(x^t, x^{t+1}) = (x^{t-1}, x^t)$
8: **end if**
9: **end for**

Besides, in this work, we will use $|x|_1$ to denote the number of ones in the bitstring x, and use $[a..b] := \{a, a+1, \ldots, b\}$ for $a \leq b$ and $a, b \in \mathbb{Z}$. The optimization goal is to reach an optimal solution for the first time.

3 Different But Easy-to-Reach Optimal Function Value

This paper will consider three positive circumstances in which the time-linkage property will help the optimization by the evolutionary algorithms. The optimal function value and optimal solution set may change after the time-linkage property is introduced. This section will consider the first circumstance in which the optimal function value becomes different. Other two circumstances in Sects. 4 and 5 consider the unchanged optimal function value.

3.1 CLIFF$_{dTL}$

The problem becomes easier as the time-linkage property changes the optimal function value to an easy-to-reach one. To better demonstrate it, we start from a benchmark that is not that easy to solve. Multimodal benchmarks can be somehow difficult to solve due to the existence of local optima. Here we choose the popular CLIFF$_d$ benchmark (Definition 1), extending a construction by [7], as an example. For this problem, the optimal solution set is $\{1^n\}$ and the optimal function value is $n - d + \frac{1}{2}$. Paixao et al. [9] proved that the expected runtime time of the $(1+1)$ EA on CLIFF$_d$ with $d \in [2..\frac{n}{2}]$ is $\Theta(n^d)$.

Definition 1 (Cliff$_d$ [9]). *For any $x = (x_1, \ldots, x_n)$, the CLIFF$_d$: $\{0,1\}^n \to \mathbb{R}$ is defined by*

$$\textsc{Cliff}_d(x) = \begin{cases} |x|_1, & \text{if } |x|_1 \leq n-d \\ |x|_1 - d + \frac{1}{2}, & \text{otherwise} \end{cases}$$

where $|x|_1 = \sum_{i=1}^{n} x_i$ counts the number of ones.

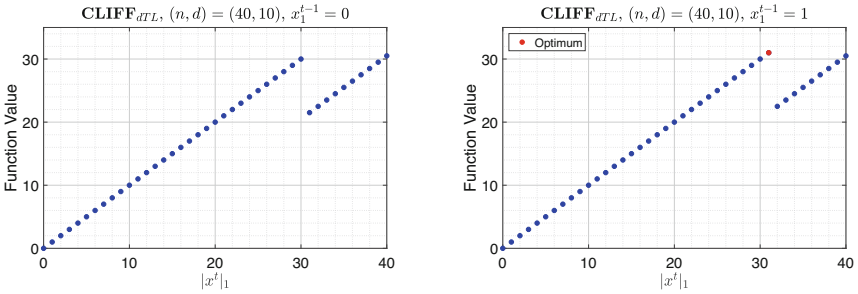
Now we construct the time-linkage variant of CLIFF$_d$. As discussed in Sect. 2, we will only consider the time-linkage effect of the value of the first dimension in the previous generation. The time-linkage takes effect if the previous first bit has the value of 1, and the optimal function value only exists in this situation. See Definition 2 for the formal description of our time-linkage CLIFF$_d$, called CLIFF$_{dTL}$.

Definition 2 (Cliff$_{dTL}$). *Let $n \in \mathbb{N}$, the CLIFF$_{dTL}$: $\{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$ is defined by*

$$\textsc{Cliff}_{dTL}(x^{t-1}, x^t) = \begin{cases} n - d + \frac{1}{2}, & |x^t|_1 = n \\ |x^t|_1 - d + \frac{1}{2}, & |x^t|_1 - x_1^{t-1} \in [n-d+1, n-1] \\ |x^t|_1, & \text{else} \end{cases}$$

where $x^{t-1} = (x_1^{t-1}, ..., x_n^{t-1}) \in \{0,1\}^n$ and $x^t = (x_1^t, ..., x_n^t) \in \{0,1\}^n$ are two consecutive solutions.

For a better illustration, we plot the function values of the CLIFF$_{dTL}$ in Fig. 1. If the first bit of the last generation $x_1^{t-1} = 0$, it is identical to the original CLIFF$_d$, but is different for $x_1^{t-1} = 1$. The optimal function value is 31, witnessed when x^t with $|x^t|_1 = 31$ and $x_1^{t-1} = 1$.



(a) $x_1^{t-1}=0$ (b) $x_1^{t-1}=1$

Fig. 1. CLIFF$_{dTL}$ with $(n,d)=(40,10)$

We form its optimal function value and optimal solution set in the following lemma.

Lemma 1. *The optimal function value of* CLIFF$_{dTL}$ *is* $n-d+1$*, and the optimal solution set is* $\{(x^{t-1},x^t) \mid (x_1^{t-1},|x^t|_1)=(1,n-d+1)\}$.

Compared with the CLIFF$_d$, we see that the optimal function value changes from $n-d+1/2$ to $n-d+1$, and the optimal set changes from $\{1^n\}$ to $\{(x^{t-1},x^t) \mid (x_1^{t-1},|x^t|_1)=(1,n-d+1)\}$, which seems easier to be reached.

3.2 Runtime Analysis of the (1 + 1) EA

Here, we conduct the runtime analysis of the $(1+1)$ EA optimizing CLIFF$_{dTL}$ and will show that the expected runtime is reduced from $\Theta(n^d)$ (for the original CLIFF$_d$) to $O(n\ln n)$ conditional on an event (that happens with $\Omega(1)$ probability).

We first list two lemmas that will be used in our proofs.

Lemma 2 (Cheronff Inequality [3]). *Let* $X_1,\ldots,X_n$ *be independent random variables taking values in* $[0,1]$*. Let* $X=\sum_{i=1}^n X_i$*. Let* $\delta\geq 0$*. Then*

$$\Pr[X\geq(1+\delta)E[X]]\leq\exp\left(-\frac{\min\{\delta^2,\delta\}E[X]}{3}\right).$$

Lemma 3 ([9]). *For all* $0\leq i<j\leq n$*. Let* $\textsc{mut}(i,j)$ *denote the probability that a global mutation of a search point with* i *ones creates an offspring with* j *ones. Then*

$$\frac{\textsc{mut}(i,j)}{\sum_{k=j}^n \textsc{mut}(i,k)}\geq\frac{1}{2}.$$

Our main runtime result is shown in the following theorem.

Theorem 1. *Let $d \in [3..\frac{n}{4}]$. Conditional on an event that happens with $\Omega(1)$ probability, the expected runtime for the $(1+1)$ EA to optimize $\textsc{Cliff}_{dTL}$ is $O(n \ln n)$.*

Proof. We assume that the initial individual x^1 has less than $n-d$ number of ones. Let $X = \sum_{i=1}^{n} x_i^1$, then $E[X] = \frac{n}{2}$. From Lemma 2, we have

$$\Pr\left[\sum_{i=1}^{n} x_i^1 \geq \frac{3n}{4}\right] = \Pr\left[X \geq \left(1+\frac{1}{2}\right)E[X]\right] \leq \exp\left(-\frac{n}{24}\right).$$

Hence, we know the probability of $|x^1|_1 < n-d$ is at least

$$1 - \exp\left(-\frac{n}{24}\right). \tag{1}$$

Now we divide the whole process into two phases. The first phase ends when an individual with at least $n-d$ number of ones is reached for the first time. Then the second phase starts, and ends when the optimum is reached.

We first consider the expected number of iterations to reach a solution with at least $n-d$ number of ones. From Definition 2, we know that for any solution x^t with $|x^t|_1 \leq n-d$, $\textsc{Cliff}_{dTL}\ (x^{t-1}, x^t) = |x^t|_1$. Hence, before an individual with at least $n-d$ number of ones is reached, the process is identical to the process of the original (1+1) EA (without considering the time-linkage property) optimizing OneMax. That is, for any solution x^t with $|x^t|_1 = i, i < n-d$, the probability of generating an offspring with better fitness is

$$\Pr\left[|\tilde{x}^t|_1 \geq i+1 \mid |x^t|_1 = i\right] \geq \frac{n-i}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{n-i}{en},$$

and the expected number of iterations to reach a search solution with at least $n-d$ number of ones is at most

$$\sum_{i=0}^{n-d-1} \frac{en}{n-i} = en \sum_{i=d+1}^{n} \frac{1}{i} \leq en \ln n. \tag{2}$$

Note if a solution with exactly $n-d$ number of ones is generated during the phase, it will be accepted. From Lemma 3, we know that with a probability at least

$$\frac{1}{2}, \tag{3}$$

such a search solution indeed has $n-d$ number of ones. In the following, we consider the process (the second phase) after a search solution with $n-d$ number of ones is reached for the first time.

If the second phase starts from x^t with $(x_1^t, |x^t|_1) = (1, n-d)$, we regard it as a success if the offspring is 1^n or has $(x_1^t, |\tilde{x}^t|_1) = (1, n-d+1)$ and as a failure if $(\tilde{x}_1^t, |\tilde{x}^t|_1) = (0, n-d)$. Note that only when a success or a failure happens can x^t enter into the next population. Hence, for other cases, x^t will keep its

current status, that is, $(x_1^t, |x^t|_1) = (1, n-d)$. Besides, once a failure happens $((\tilde{x}_1^t, |\tilde{x}^t|_1) = (0, n-d))$, its offspring in the next generation survives only when the previous considered success event happens or it returns to our starting status x^t with $(x_1^t, |x^t|_1) = (1, n-d)$. Therefore, let T_1 denote the number of iterations to witness a success, let T_2 denote the total number of iterations that are spent in the failure event and returning back before a success, and then we have

$$E[T_1] = \frac{1}{p_1} + E[T_2],$$

where p_1 denotes the probability of a success event from x^t with $(x_1^t, |x^t|_1) = (1, n-d)$. It is not difficult to see that

$$p_1 > \left(\frac{n-1}{n}\right)^{n-1} \frac{d}{n} > \frac{d}{en}.$$

Now we calculate $E[T_2]$. Let p_2 denote the probability of a failure event from x^t with $(x_1^t, |x^t|_1) = (1, n-d)$. Then we know that

$$\frac{d}{en^2} < \frac{1}{n}\frac{d}{n}\left(1 - \frac{1}{n}\right)^{n-2} < p_2 < \frac{d}{n^2}.$$

Note that once a success event happens, the failure event will no longer occur. Hence, we know the expected number of failures before a success is

$$\frac{p_1 + p_2}{p_1} - 1. \tag{4}$$

Let p_3 denote the probability to generate an offspring $\tilde{x}^t$ with $(\tilde{x}_1^t, |\tilde{x}^t|_1) = (1, n-d)$ or $\tilde{x}^t = 1^n$ from x^t with $(x_1^t, |x^t|_1) = (0, n-d)$. Then we know

$$p_3 > \frac{1}{n}\frac{n-d}{n}\left(1 - \frac{1}{n}\right)^{n-2} > \frac{n-d}{en^2}.$$

Note that only the above kinds of offspring will survive. Hence, the expected number of iterations of returning back or witness a success is $\frac{1}{p_3}$. We pessimistically consider that $\tilde{x}^t$ with $(\tilde{x}_1^t, |\tilde{x}^t|_1) = (1, n-d)$ is reached (otherwise, the success is already reached). Then we know

$$E[T_2] = \left(\frac{p_1 + p_2}{p_1} - 1\right)\frac{1}{p_3} = \frac{p_2}{p_1 p_3} < \frac{\frac{d}{n^2}}{\frac{d}{en}\frac{n-d}{en^2}} = \frac{e^2 n}{n-d}.$$

Hence, we have

$$E[T_1] = \frac{1}{p_1} + E[T_2] < \frac{en}{d} + \frac{e^2 n}{n-d} = en\left(\frac{1}{d} + \frac{e}{n-d}\right). \tag{5}$$

If the second phase starts from a solution x^t with $(x_1^t, |x^t|_1) = (0, n-d)$, then from the above analysis we know within $\frac{1}{p_3} < \frac{en^2}{n-d}$ expected number of iterations,

it moves to the case of x^t with $(x_1^t, |x^t|_1) = (1, n-d)$ that we discussed before, or reaches a success. That is, we only need additional $\frac{en^2}{n-d}$ expected number of iterations compared to $E[T_1]$ to witness a success.

Recall that a success event requires to generate $\tilde{x}^t$ with $|\tilde{x}^t|_1 = n$ or $(x_1^t, |\tilde{x}^t|_1) = (1, n-d+1)$. Now we calculate the probability that we witness $(x_1^t, |\tilde{x}^t|_1) = (1, n-d+1)$ (the optimum) instead of 1^n. From the above analysis, we know that we possibly witness a success, from two cases, that is, from x^t with $(x_1^t, |x^t|_1) = (1, n-d)$ and from x^t with $(x_1^t, |x^t|_1) = (0, n-d)$. For the former case, both 1^n and the optimum can be reached as a success, but for the latter case, only the success of $\tilde{x}^t = 1^n$ can happen. Let p_4 denote the probability of generating 1^n from x^t with $(x_1^t, |x^t|_1) = (1, n-d)$, and we know that

$$p_4 < \frac{1}{n^d}.$$

Together with p_1, we know that for one success witnessed from $(x_1^t, |x^t|_1) = (1, n-d)$, we reach the optimum with probability

$$1 - \frac{p_4}{p_1} > 1 - \frac{e}{dn^{d-1}}. \tag{6}$$

Let p_5 denote the probability of generating 1^n (the only success case) from x^t with $(x_1^t, |x^t|_1) = (0, n-d)$, and we know that

$$p_5 = p_4 < \frac{1}{n^d}.$$

Together with p_3, we know that for one time of leaving the failure status via returning back or witness a success from x^t with $(x_1^t, |x^t|_1) = (0, n-d)$, the probability for leaving the failure status via returning back is

$$1 - \frac{p_5}{p_3} > 1 - \frac{e}{(n-d)n^{d-2}}. \tag{7}$$

Here, we consider the following artificial process. It is identical to the original process except that it will not count it as a success when 1^n is reached from x^t with $(x_1^t, |x^t|_1) = (0, n-d)$ (once it happens the process will directly return back to $(x_1^t, |x^t|_1) = (1, n-d)$), and ends when an optimum is reached for the first time. It is not difficult to see that before 1^n is reached for the first time, two processes are identical. Now we calculate the probability of reaching an optimum (ending the process) before 1^n for the artificial process. To this aim, we first estimate the total number of times of entering the failures, denoted as T_3. From (4), we know that

$$E[T_3] = \frac{p_2}{p_1} < \frac{e}{n}.$$

From Markov inequality, we know

$$\Pr[T_3 \geq e] \leq \frac{1}{n}.$$

Hence,

$$\Pr[T_3 < e] \geq 1 - \frac{1}{n}. \tag{8}$$

Then from (6) (7) (8), we know that such probability is at least

$$\left(1 - \frac{e}{dn^{d-1}}\right)\left(1 - \frac{1}{n}\right)\left(1 - \frac{e}{(n-d)n^{d-2}}\right)^{e}. \tag{9}$$

Therefore, from (1) (3) (9) and (2) (5), we know that conditional on an event that happens with probability at least

$$\frac{1}{2}\left(1 - \exp\left(-\frac{n}{24}\right)\right)\left(1 - \frac{e}{dn^{d-1}}\right)\left(1 - \frac{1}{n}\right)\left(1 - \frac{e}{(n-d)n^{d-2}}\right)^{e} = \Omega(1),$$

the optimum is reached in expected

$$en\ln n + en\left(\frac{1}{d} + \frac{e+n}{n-d}\right) = en\left(\ln n + \frac{1}{d} + \frac{n+e}{n-d}\right) = O(n\ln n)$$

number of iterations.

The above theorem shows that changing the optimal function value to an easy-to-reach one, $\textsc{Cliff}_{dTL}$ is significantly easier to optimize by the $(1+1)$ EA compared to the expected runtime of $\Theta(n^d)$ for the original $\textsc{Cliff}_d$. As this message is clearly conveyed, we do not intend to derive tighter estimates of conditional probabilities, which might need more complicated calculations. Instead, we conducted the experiments in Sect. 6, and the results indicate that such conditional probabilities are actually very close to 1.

4 Same Optimal Function Value but Larger Optimal Solution Set

Section 3 discussed the positive circumstance of the time-linkage property changing the optimal function value to an easy-to-reach one. In this section, we consider one positive circumstance that the optimal function value is unchanged but the optimal solution set becomes larger.

4.1 $\textsc{Cliff}'_{dTL}$

Similar in Sect. 3, we resort to $\textsc{Cliff}_d$ function as a basic hard problem, and construct a time-linkage variant with same optimal function value but with a larger optimal solution set. We consider the following $\textsc{Cliff}'_d$ as the example.

Definition 3 ($\textsc{Cliff}'_d$). *For any $x = (x_1, \ldots, x_n)$, the $\textsc{Cliff}'_d : \{0,1\}^n \to \mathbb{R}$ is defined by*

$$\textsc{Cliff}'_d(x) = \begin{cases} |x|_1, & \text{if } |x|_1 \leq n-d \\ |x|_1 - d + 1, & \text{otherwise} \end{cases}$$

where $|x|_1 = \sum_{i=1}^{n} x_i$ counts the number of ones.

For $\textsc{Cliff}'_d$, the optimal function value is $n-d+1$ and the optimal solution set is $\{1^n\}$.

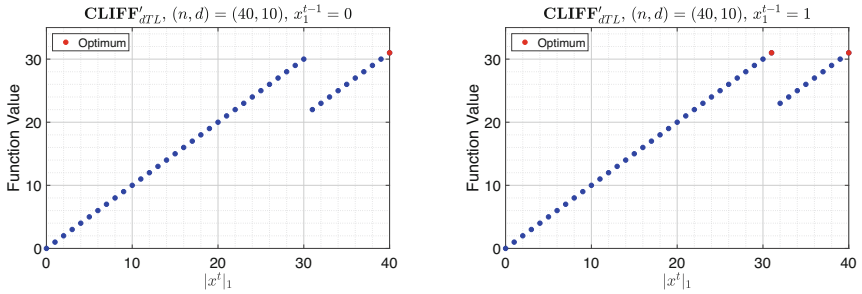
One may note that above $\textsc{Cliff}'_d$ changes the constant component in the original $\textsc{Cliff}_d$ from $1/2$ to 1. With this modification, we now construct a time-linkage variant with the same optimal function value of $n-d+1$ but with a larger optimal solution set. Formally, the time-linkage variant, called $\textsc{Cliff}'_{dTL}$, is defined as follows.

Definition 4 ($\textsc{Cliff}'_{dTL}$). *Let $n \in \mathbb{N}$. $\textsc{Cliff}'_{dTL} : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$ is defined by*

$$\textsc{Cliff}'_{dTL}(x^{t-1}, x^t) = \begin{cases} n-d+1, & |x^t|_1 = n \\ |x^t|_1 - d + 1, & |x^t|_1 - x_1^{t-1} \in [n-d+1, n-1] \\ |x^t|_1, & \textit{else} \end{cases}$$

where $x^{t-1} = (x_1^{t-1}, ..., x_n^{t-1}) \in \{0,1\}^n$ and $x^t = (x_1^t, ..., x_n^t) \in \{0,1\}^n$ are two consecutive solutions.

Figure 2 gives an example plot of $\textsc{Cliff}'_{dTL}$. It is identical to $\textsc{Cliff}'_d$ if $x_1^{t-1} = 0$ but has the extra x^t with $|x^t|_1 = 31$ as the optima, which are easier to reach than 1^n (the optimum of the original problem) if $x_1^{t-1} = 1$.



(a) $x_1^{t-1} = 0$ (b) $x_1^{t-1} = 1$

Fig. 2. $\textsc{Cliff}'_{dTL}$ with $(n, d) = (40, 10)$

The optimal function value and optimal solution set of $\textsc{Cliff}'_{dTL}$ is shown in the following lemma.

Lemma 4. *The optimal function value of $\textsc{Cliff}'_{dTL}$ is $n-d+1$, and the optimal solution set is $\{(x^{t-1}, x^t) \mid (x_1^{t-1}, |x^t|_1) = (1, n-d+1)$ or $(x_1^{t-1}, |x^t|_1) = (*, n)\}$.*

Compared with $\textsc{Cliff}'_d$, we see that the optimal function value is still $n-d+1$ but the optimal solution set becomes larger.

4.2 Runtime Analysis of the (1 + 1) EA

Before we give the analysis of the (1+1) EA optimizing $\textsc{Cliff}'_{dTL}$, the following theorem shows that the expected running time of the (1 + 1) EA on $\textsc{Cliff}'_d$ is $\Omega(n^{d-2})$. Due to space limit, we omit the proof here.

Theorem 2. *Let $4 \leq d \leq \frac{n}{4}$, conditional on an event that happens with $\Omega(1)$ probability, the expected runtime of the (1 + 1) EA on $\textsc{Cliff}'_d$ is $\Omega(n^{d-2})$.*

Now we have our main theorem of this section.

Theorem 3. *Let $\max\{2\ln n + 1, e^2 + 1\} \leq d \leq \frac{n}{4}$, conditional on an event that happens with $\Omega(1)$ probability, the expected runtime of the (1 + 1) EA on $\textsc{Cliff}'_{dTL}$ is $O(n \ln n)$.*

From Theorems 2 and 3, we could see that the expected runtime of $\Omega(n^{d-2})$ is significantly reduced to $O(n \ln n)$ conditional on an event that happens with $\Omega(1)$ probability. The reason is that the time-linkage property enlarges the optimal solution set. Similar to the discussion at the end of Sect. 4, our experiments in Sect. 6 will indicate that the above conditional probability is quite close to 1 in the actual runs.

5 Same Optimal Function Value and Optimal Solution Set

Now, we discuss another positive circumstance in which the optimal function value is unchanged after the time-linkage property is introduced. Different from Sect. 4 that the optimal solution set changes, this section will show that even if the optimal solution set is unchanged, the time-linkage property helps when the intermediate good solution is more prone to be maintained.

5.1 $\textsc{Jump}_{kTL}$

Similar to the previous two sections, we will consider a multimodal benchmark as a hard problem. Here, we restore another well-analyzed Jump benchmark [6] (Definition 5) as an example, which has inspired many interesting results [1,5, 8,10,13,16,17].

Definition 5 (Jump). *For any $x = (x_1, \ldots, x_n)$, the $\textsc{Jump} : \{0,1\}^n \to \mathbb{R}$ is defined by*

$$\textsc{Jump}(x) = \begin{cases} n - |x|_1, & \text{if } |x|_1 \in [n-k+1..n-1] \\ k + |x|_1, & \text{else} \end{cases}$$

where $|x|_1 = \sum_{i=1}^{n} x_i$ counts the number of ones.

For Jump, the optimal function value is $n + k$ and the optimal solution set is 1^n. The expected runtime of the (1 + 1) EA is $O(n^d)$.

We construct the following time-linkage variant, called $\textsc{Jump}_{kTL}$, where the gap size is smaller when the previous first bit has the value of 1.

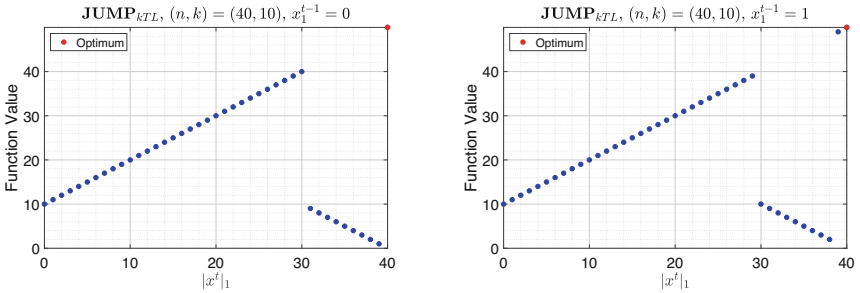
Definition 6. *Let* $n \in \mathbb{N}, k \in [1..n]$. *The* $\textsc{Jump}_{kTL}$ *function* $f : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$ *is defined by*

$$\textsc{Jump}_{kTL}(x^{t-1}, x^t) = \begin{cases} n - |x^t|_1, & |x^t|_1 + x_1^{t-1} \in [n-k+1..n-1] \\ k + |x^t|_1, & else \end{cases}$$

where $x^{t-1} = (x_1^{t-1}, ..., x_n^{t-1}) \in \{0,1\}^n$ *and* $x^t = (x_1^t, ..., x_n^t) \in \{0,1\}^n$ *are two consecutive solutions.*

It is not difficult to see that the optimal function value and optimal solution set are the same as the ones for the original Jump.

Figure 3 gives an example plot of $\textsc{Jump}_{kTL}$. It is identical to Jump if $x_1^{t-1} = 0$ but one gap point ($|x^t|_1 = n - 1 = 39$) will have larger fitness value and become prone to be maintained if $x_1^{t-1} = 1$.



(a) $x_1^{t-1} = 0$ (b) $x_1^{t-1} = 1$

Fig. 3. $\textsc{Jump}_{kTL}$ with $(n,k) = (40,10)$

5.2 Runtime Analysis of the (1 + 1) EA

We first extract the situation starting from x^t with $\textsc{Jump}_{kTL}$ $(x^{t-1}, x^t) \leq k$ into the following lemma.

Lemma 5. *Consider using the* $(1+1)$ *EA to optimize* $\textsc{Jump}_{kTL}$. *Assume that it starts from* x^t *with* $\textsc{Jump}_{kTL}$ $(x^{t-1}, x^t) \leq k$, *then within* $en \ln n$ *expected number of iterations a solution with fitness larger than* k *is reached for the first time.*

Now we present our main theorem of this section.

Theorem 4. *Let* $0 < k \leq \frac{n}{2}$, *the expected running time of the* $(1+1)$ *EA on* $\textsc{Jump}_{kTL}$ *is* $O(n^{k-1})$.

Theorem 4 shows that the performance improvement can happen even if the time-linkage property neither changes the optimal function value nor the optimal solution set.

6 Experiments

Here we conduct the experiments to see how the (1+1) EA actually performs for $\textsc{Cliff}_{dTL}$, $\textsc{Cliff}'_{dTL}$, and $\textsc{Jump}_{kTL}$, and also to see the conditional probabilities stated in Theorems 1 and 3.

Firstly for $\textsc{Cliff}_{dTL}$ and $\textsc{Cliff}'_{dTL}$, we set $n \in \{40, 50, 60, 70, 80\}$ and $d = 10$ to ensure that the inequality $\max\{2\ln n + 1, e^2 + 1\} \leq d \leq \frac{n}{4}$ (in Theorem 3) holds. The $(1 + 1)$ EA algorithm starts with a solution x^1 with $|x^1|_1 < n - d$. Experiments were conducted with 100 independent runs and terminated once an optimal solution is reached for the first time.

Figure 4 plots the mean (with standard deviations) number of iterations for the $(1 + 1)$ EA to reach an optimum of $\textsc{Cliff}_{dTL}$ and $\textsc{Cliff}'_{dTL}$ for the first time. For comparison, the curves of n^d and n^{d-2} are plotted as rough indicators of the runtime for $\textsc{Cliff}_d$ ($\Theta(n^d)$ in [9]) and $\textsc{Cliff}'_d$ ($\Omega(n^{d-2})$ in Theorem 2), respectively. Additionally, the curve of $4n \ln n$ is included to illustrate the runtime performance of the (1+1) EA optimizing the time-linkage problem. We can easily see that it is quite easier for the (1+1) EA to solve two time-linkage $\textsc{Cliff}_d$ than the original $\textsc{Cliff}_d$ ($\Omega(n^d)$ number of iterations) and $\textsc{Cliff}'_d$ ($\Omega(n^{d-2})$ number of iterations). Moreover, we also logged intermediate data to validate the $\Omega(1)$ probability of the event in Theorems 1 and 3, showing the probability is very closed to 1.

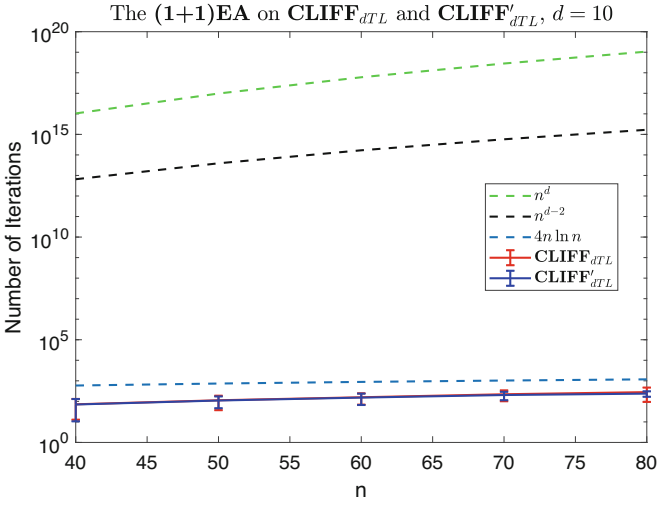


Fig. 4. The mean (with standard deviations) number of iterations of the $(1 + 1)$ EA on $\textsc{Cliff}_{dTL}$ and $\textsc{Cliff}'_{dTL}$ with $n \in \{40, 50, 60, 70, 80\}$ and $d = 10$ in 100 independent runs.

For the $\textsc{Jump}_{kTL}$ problem, we set $n \in \{20, 25, 30, 35, 40\}$ and $d = 5$ (where the algorithm can reach the optimum in a reasonable time), also for 100 independent runs. Same settings are conducted for optimizing the Jump problem for

comparison. Table 1 collected the mean (with standard deviations) numbers of iterations for the $(1+1)$ EA to reach an optimum of $\textsc{Jump}_{kTL}$ and $\textsc{Jump}$ for the first time. The Mann-Whitney U test showed clear improvements for all tested problem sizes, which verifies that the time-linkage problem $\textsc{Jump}_{kTL}$ is easier for the $(1+1)$ EA to optimize compared to the original $\textsc{Jump}$ problem.

Table 1. The number of iterations of the (1+1) EA on $\textsc{Jump}$ and $\textsc{Jump}_{kTL}$ (mean (std)) with $n \in \{20, 25, 30, 35, 40\}$ and $d = 5$ in 100 independent runs. We also conducted a Mann-Whitney U test with a confidence level of 0.99, indicating significant difference in the runtime of the $(1+1)$ EA on $\textsc{Jump}$ and $\textsc{Jump}_{kTL}$ for each value of n.

	$\textsc{Jump}$	$\textsc{Jump}_{kTL}$
n = 20	6,513,484 (6,048,982)	**72,458** (81,162)
n = 25	23,799,940 (25,009,767)	**187,023** (182,741)
n = 30	48,140,360 (44,635,768)	**443,183** (420,484)
n = 35	127,194,355 (144,156,985)	**665,242** (574,238)
n = 40	214,915,647 (209,601,405)	**1,291,213** (1,192,885)

7 Conclusion

Although all existing theoretical works show that the time-linkage property brings negative results, in this paper, we construct several time-linkage problems showing that the time-linkage property may help the optimization by evolutionary algorithms. For the $\textsc{Cliff}_d$ problem, the time-linkage property changes the optimal value to an easy-to-reach one, making it significantly easier to optimize. By enlarging the optimal solution set but keeping the optimal value unchanged, the $\textsc{Cliff}'_d$ problem with time-linkage property is proven to be easier for the $(1+1)$ EA to reach the optima. In this last case, we took the $\textsc{Jump}$ problem as an example and showed that the time-linkage property can also help even if both the optimal function value and optimal solution set are unchanged.

In summary, we conducted the first runtime analyses showing that the time-linkage property can also help optimization by evolutionary algorithms in three circumstances.

Acknowledgments. This work was supported by National Natural Science Foundation of China (Grant No. 62350710797, 62306086, 62250710682), Science, Technology and Innovation Commission of Shenzhen Municipality (Grant No. GXWD20220818191018001), and Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019A1515110177).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bian, C., Zhou, Y., Li, M., Qian, C.: Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In: International Joint Conference on Artificial Intelligence, IJCAI 2023, pp. 5513–5521. ijcai.org (2023)
2. Bosman, P.A.: Learning, anticipation and time-deception in evolutionary online dynamic optimization. In: Genetic and Evolutionary Computation Conference Workshop, GECCO 2005 Workshop, pp. 39–47. ACM (2005)
3. Doerr, B.: Probabilistic tools for the analysis of randomized optimization heuristics. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 1–87. Springer, Cham (2020). https://arxiv.org/abs/1801.06733
4. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. In: Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 1449–1456. ACM (2010)
5. 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)
6. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoret. Comput. Sci. **276**, 51–81 (2002)
7. 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)
8. Jansen, T., Wegener, I.: The analysis of evolutionary algorithms - a proof that crossover really can help. Algorithmica **34**, 47–66 (2002)
9. Paixao, T., Pérez Heredia, J., Sudholt, D., Trubenova, B.: First steps towards a runtime comparison of natural and artificial evolution. In: Genetic and Evolutionary Computation Conference, GECCO 2015, pp. 1455–1462. ACM (2015)
10. Rajabi, A., Witt, C.: Self-adjusting evolutionary algorithms for multimodal optimization. Algorithmica **84**, 1694–1723 (2022)
11. Sudholt, D.: Crossover speeds up building-block assembly. In: Genetic and Evolutionary Computation Conference, GECCO 2012, pp. 689–702. ACM (2012)
12. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Comb. Probab. Comput. **22**, 294–318 (2013)
13. Witt, C.: How majority-vote crossover and estimation-of-distribution algorithms cope with fitness valleys. Theoret. Comput. Sci. **940**, 18–42 (2023)
14. Yang, T., Zhou, Y.: Analysis of multi-objective evolutionary algorithms on fitness function with time-linkage property. IEEE Trans. Evol. Comput. **28**, 837–843 (2024)
15. Zheng, W., Chen, H., Yao, X.: Analysis of evolutionary algorithms on fitness function with time-linkage property. IEEE Trans. Evol. Comput. **25**, 696–709 (2021)
16. Zheng, W., Doerr, B.: Theoretical analyses of multiobjective evolutionary algorithms on multimodal objectives. Evol. Comput. **31**, 337–373 (2023)
17. Zheng, W., Doerr, B.: Runtime analysis of the SMS-EMOA for many-objective optimization. In: Conference on Artificial Intelligence, AAAI 2024, pp. 20874–20882. AAAI Press (2024)
18. Zheng, W., Zhang, Q., Chen, H., Yao, X.: When non-elitism meets time-linkage problems. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 741–749. ACM (2021)

A First Running Time Analysis of the Strength Pareto Evolutionary Algorithm 2 (SPEA2)

Shengjie Ren[1,2], Chao Bian[1,2], Miqing Li[3], and Chao Qian[1,2(✉)]

[1] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
201300036@smail.nju.edu.cn, bianc@lamda.nju.edu.cn, qianc@lamda.nju.edu.cn
[2] School of Artificial Intelligence, Nanjing University, Nanjing, China
[3] School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K.
m.li.8@bham.ac.uk

Abstract. Evolutionary algorithms (EAs) have emerged as a predominant approach for addressing multi-objective optimization problems. However, the theoretical foundation of multi-objective EAs (MOEAs), particularly the fundamental aspects like running time analysis, remains largely underexplored. Existing theoretical studies mainly focus on basic MOEAs, with little attention given to practical MOEAs. In this paper, we present a running time analysis of strength Pareto evolutionary algorithm 2 (SPEA2) for the first time. Specifically, we prove that the expected running time of SPEA2 for solving three commonly used multi-objective problems, i.e., mOneMinMax, mLeadingOnesTrailingZeroes, and m-OneJumpZeroJump, is $O(\mu n \cdot \min\{m \log n, n\})$, $O(\mu n^2)$, and $O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$, respectively. Here m denotes the number of objectives, and the population size μ is required to be at least $(2n/m+1)^{m/2}$, $(2n/m+1)^{m-1}$ and $(2n/m-2k+3)^{m/2}$, respectively. The proofs are accomplished through general theorems which are also applicable for analyzing the expected running time of other MOEAs on these problems, and thus can be helpful for future theoretical analysis of MOEAs.

1 Introduction

Multi-objective optimization requires optimizing several objectives at the same time, and it has been seen in many real-world scenarios. Since the objectives of a multi-objective optimization problem (MOP) are usually conflicting, there does not exist a single optimal solution, but instead a set of solutions which represent different optimal trade-offs between these objectives, called Pareto optimal solutions. The objective vectors of all the Pareto optimal solutions are called the Pareto front. The goal of multi-objective optimization is to find the Pareto front or a good approximation of it.

Evolutionary algorithms (EAs) [1,23] are a large class of randomized heuristic optimization algorithms inspired by natural evolution. They maintain a set of solutions (called a population), and iteratively improve it by generating new solutions and replacing inferior ones. The population-based nature makes EAs particularly effective in tackling MOPs, leading to their widespread application across various real-world domains [9,13,31,43,49]. Notably, there have been developed a multitude of well-established multi-objective EAs (MOEAs), including non-dominated

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 295–312, 2024.
https://doi.org/10.1007/978-3-031-70071-2_19

sorting genetic algorithm II (NSGA-II) [15], strength Pareto evolutionary algorithm 2 (SPEA2) [50], $\mathcal{S}$ metric selection evolutionary multi-objective optimization algorithm (SMS-EMOA) [2], and multi-objective evolutionary algorithm based on decomposition (MOEA/D) [44].

In contrast to the wide applications of MOEAs, the theoretical foundation of MOEAs, especially the essential aspect, running time analysis, is still underdeveloped, which is mainly due to the sophisticated behaviors of EAs and the hardness of MOPs. Early theoretical research primarily concentrates on analyzing the expected running time of GSEMO [26] and SEMO [30] for solving a variety of multi-objective synthetic and combinatorial optimization problems [4,18,26–28,33,34,38]. Note that GSEMO is a simple MOEA which employs the bit-wise mutation operator to generate an offspring solution in each iteration and keeps the non-dominated solutions generated-so-far in the population, and SEMO is a variant of GSEMO which employs one-bit mutation instead of bit-wise mutation. Furthermore, based on GSEMO and SEMO, the effectiveness of some parent selection strategies [10,24,25,30], mutation operator [22], crossover operator [38], and selection hyper-heuristics [37], has also been studied.

Recently, researchers have begun to examine practical MOEAs. The expected running time of $(\mu + 1)$ SIBEA, a simple MOEA employing the hypervolume indicator for population update, has been analyzed across various synthetic problems [7,17,35], contributing to the theoretical understanding of indicator-based MOEAs. Subsequently, attention has turned to well-established algorithms in the evolutionary multi-objective optimization field. Huang *et al.* [29] investigated MOEA/D, assessing the effectiveness of different decomposition methods by comparing running time for solving many-objective synthetic problems. Additionally, Zheng *et al.* [46] conducted the first analysis of the expected running time of NSGA-II by considering the bi-objective OneMinMax and LeadingOnesTrailingZeroes problems. Bian *et al.* [6] analyzed the running time of SMS-EMOA [2] for solving the bi-objective OneJumpZeroJump problem, and showed that a stochastic population update method can bring exponential acceleration. Moreover, Wietheger and Doerr [40] demonstrated that NSGA-III [14] exhibits superior performance over NSGA-II in solving the tri-objective problem 3OneMinMax. In a very recent study, Lu *et al.* [32] analyzed interactive MOEAs (iMOEAs), pinpointing scenarios where iMOEAs may work or fail. Some other works about well-established MOEAs include [3,5,8,11,12,19–21,36,39,45,47,48].

However, the running time analysis of SPEA2, one of the most popular MOEAs [50], has not been touched. SPEA2 employs *enhanced fitness assignment* and *density estimation* mechanisms. The former improves the selection pressure towards the Pareto-optimal front, while the latter helps maintain a diverse solution set by considering the density of surrounding solutions. In this paper, we analyze the expected running time of SPEA2 for solving three multi-objective problems, i.e., mOneMinMax, mLeadingOnes-TrailingZeroes, and m-OneJumpZeroJump, which have been commonly used in the theory community of MOEAs [19,30,36,46–48]. Note that $m \geq 2$ denotes the number of objectives. Specifically, we prove that the expected number of fitness evaluations of SPEA2 for finding the Pareto front of the three problems is $O(\mu n \cdot \min\{m \log n, n\})$, $O(\mu n^2)$, and $O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$, respectively, where the population size μ is required to be at least $(2n/m + 1)^{m/2}$, $(2n/m + 1)^{m-1}$ and

$(2n/m - 2k + 3)^{m/2}$, correspondingly. The proofs are accomplished through general theorems which can also be applied for analyzing the expected running time of other MOEAs.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries. The general theorems on the running time of MOEAs are provided in Sect. 3, and these theorems are applied to SPEA2 in Sect. 4 and other MOEAs in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Preliminaries

In this section, we first introduce multi-objective optimization. Then, we introduce the analyzed algorithm, SPEA2. Finally, we present the mOneMinMax, mLeadingOnes-Trailingzeroes and mOneJumpZeroJump problems considered in this paper.

2.1 Multi-objective Optimization

Multi-objective optimization seeks to optimize two or more objective functions concurrently, as presented in Definition 1. In this paper, we focus on maximization (though minimization can be similarly defined) and pseudo-Boolean functions whose solution space $\mathcal{X} = \{0,1\}^n$. Given that the objectives of a practical MOP typically conflict with each other, a canonical complete order within the solution space $\mathcal{X}$ does not exist. Instead, we employ the *domination* relationship presented in Definition 2 to compare solutions. A solution is deemed *Pareto optimal* if no other solution in $\mathcal{X}$ dominates it, and the collection of objective vectors of all the Pareto optimal solutions is called the *Pareto front*. The goal of multi-objective optimization is to identify the Pareto front or its good approximation.

Definition 1 (Multi-objective Optimization). *Given a feasible solution space $\mathcal{X}$ and objective functions $f_1, f_2, \ldots, f_m$, multi-objective optimization can be formulated as*

$$\max_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{f}(\boldsymbol{x}) = \max_{\boldsymbol{x}\in\mathcal{X}} \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})\big).$$

Definition 2 (Domination). *Let $\boldsymbol{f} = (f_1, f_2, \ldots, f_m) : \mathcal{X} \to \mathbb{R}^m$ be the objective vector. For two solutions $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathcal{X}$:*

- $\boldsymbol{x}$ weakly dominates $\boldsymbol{y}$ *(denoted as $\boldsymbol{x} \succeq \boldsymbol{y}$) if for any $1 \le i \le m$, $f_i(\boldsymbol{x}) \ge f_i(\boldsymbol{y})$;*
- $\boldsymbol{x}$ dominates $\boldsymbol{y}$ *(denoted as $\boldsymbol{x} \succ \boldsymbol{y}$) if $\boldsymbol{x} \succeq \boldsymbol{y}$ and $f_i(\boldsymbol{x}) > f_i(\boldsymbol{y})$ for some i;*
- $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *are* incomparable *if neither $\boldsymbol{x} \succeq \boldsymbol{y}$ nor $\boldsymbol{y} \succeq \boldsymbol{x}$.*

2.2 SPEA2

The SPEA2 algorithm [50], as presented in Algorithm 1, is a well-established MOEA which employs a regular population P and an archive A. It starts from an initial population of μ solutions and an empty archive A (lines 1–2). In each generation, it selects the non-dominated solutions in $P \cup A$ and adds them into an empty set A' (line 4). If the size of A' is larger than $\bar{\mu}$, SPEA2 uses the following truncation operator to remove

Algorithm 1 SPEA2 [50]

Input: objective functions $f_1, f_2..., f_m$, population size μ, archive size $\bar{\mu}$
Output: $\bar{\mu}$ solutions from $\{0,1\}^n$

1: $P \leftarrow \mu$ solutions uniformly and randomly selected from $\{0,1\}^n$ with replacement;
2: $A = \emptyset$;
3: **while** criterion is not met **do**
4: $A' \leftarrow$ non-dominated solutions in $P \cup A$;
5: **if** $|A'| > \bar{\mu}$ **then**
6: reduce A' by means of the truncation operator
7: **else if** $|A'| < \bar{\mu}$ **then**
8: fill A' with dominated individuals in P and A
9: **end if**
10: $A \leftarrow A'$;
11: let $P' = \emptyset, i = 0$;
12: **while** $i < \mu$ **do**
13: select a solution from A uniformly at random;
14: generate $\boldsymbol{x}'$ by flipping each bit of $\boldsymbol{x}$ independently with probability $1/n$;
15: $P' = P' \cup \{\boldsymbol{x}'\}, i = i + 1$
16: **end while**
17: $P \leftarrow P'$
18: **end while**
19: **return** A

the redundant solutions (lines 5–6). Let $\sigma_{\boldsymbol{x}}^k$ denote the distance of $\boldsymbol{x}$ to its k-th nearest neighbor in A'. We use $\boldsymbol{x} \leq_d \boldsymbol{y}$ to denote that $\boldsymbol{x}$ has a smaller distance to its neighbour compared with $\boldsymbol{y}$. That is, $\boldsymbol{x} \leq_d \boldsymbol{y} \iff (\forall 0 < k < |A'| : \sigma_{\boldsymbol{x}}^k = \sigma_{\boldsymbol{y}}^k) \vee (\exists 0 < k < |A'| : [(\forall 0 < l < k : \sigma_{\boldsymbol{x}}^l = \sigma_{\boldsymbol{y}}^l) \wedge \sigma_{\boldsymbol{x}}^k < \sigma_{\boldsymbol{y}}^k])$. The truncation operator iteratively removes an individual $\boldsymbol{x} \in A'$ such that $\boldsymbol{x} \leq_d \boldsymbol{y}$ for all $\boldsymbol{y} \in A'$, until $|A'| = \bar{\mu}$ (breaking a tie randomly). Note that once a solution is removed from A', the σ value will be updated. If the size of A' is smaller than $\bar{\mu}$, then the dominated solutions in $P \cup A$ are selected to fill the remaining slots according to their fitness (lines 7–8). The fitness of a solution is calculated as follows. First, let $S(\boldsymbol{x}) = |\{\boldsymbol{y} \in A \cup P \mid \boldsymbol{x} \succ \boldsymbol{y}\}|$ denote the strength of a solution $\boldsymbol{x}$, i.e., the number of solutions dominated by $\boldsymbol{x}$, and let $R(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in P \cup A, \boldsymbol{y} \succ \boldsymbol{x}} S(\boldsymbol{y})$. We can see that a solution with smaller R value is preferred, and $R(\boldsymbol{x}) = 0$ implies that $\boldsymbol{x}$ is non-dominated. Then, the fitness of a solution $\boldsymbol{x}$ is calculated as $F(\boldsymbol{x}) = R(x) + 1/(\sigma_{\boldsymbol{x}}^k + 2)$. After calculating the fitness of the dominated solutions in $P \cup A$, the solutions with the smallest fitness are selected into A' such that the size of A' equals to $\bar{\mu}$. After the modification of A' finishes, the archive A is set to A' (line 10). Then, the population P of size μ is formed by mutating the solutions selected from A (lines 11–17).

2.3 Benchmark Problems

Now we introduce three multi-objective problems, i.e., mOneMinMax, mLeading-Ones-Trailingzeroes, and mOneJumpZeroJump, studied in this paper, where $m \geq 2$ is a positive even number and denotes the number of objectives.

The mOneMinMax problem presented in Definition 3 divides a solution into $m/2$ blocks, and in each block, the number of 0-bits and the number of 1-bits require to be maximized simultaneously. The Pareto front is $F^* = \{(i_1, 2n/m - i_1, \cdots, i_{m/2}, 2n/m - i_{m/2}) \mid i_1, \cdots, i_{m/2} \in [0..2n/m]\}$, whose size is $(2n/m + 1)^{m/2}$, and the Pareto optimal solution corresponding to $(i_1, 2n/m - i_1, \cdots, i_{m/2}, 2n/m - i_{m/2})$ is the solution with i_j 1-bits and $(2n/m - i_j)$ 0-bits in the j-th block. We can see that any solution $\boldsymbol{x} \in \{0,1\}^n$ is Pareto optimal.

Definition 3 *(mOneMinMax [47]) Suppose m is a positive even number, and n is a multiple of $m/2$. The mOneMinMax problem of size n is to find n-bits binary strings which maximize*

$$\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))$$

with

$$f_k(\boldsymbol{x}) = \begin{cases} \sum_{i=1}^{2n/m} \boldsymbol{x}_{i+n(k-1)/m}, & \text{if } k \text{ is odd,} \\ \sum_{i=1}^{2n/m} (1 - \boldsymbol{x}_{i+n(k-2)/m}), & \text{else.} \end{cases}$$

The mLeadingOnesTrailingZeroes problem presented in Definition 4 also divides a solution into $m/2$ blocks, and in each block, the number of leading 1-bits and the number of trailing 0-bits require to be maximized simultaneously. The Pareto front is $F^* = \{(i_1, 2n/m - i_1, \cdots, i_{m/2}, 2n/m - i_{m/2}) \mid i_1, \cdots, i_{m/2} \in [0..2n/m]\}$, whose size is $(2n/m + 1)^{m/2}$, and the Pareto optimal solution corresponding to $(i_1, 2n/m - i_1, \cdots, i_{m/2}, 2n/m - i_{m/2})$ is the solution with i_j leading 1-bits and $(2n/m - i_j)$ trailing 0-bits in the j-th block.

Definition 4. *(mLeadingOnesTrailingZeroes [30]) Suppose m is a positive even number, and n is a multiple of $m/2$. The mLeadingOnesTrailingZeroes problem of size n is to find n-bits binary strings which maximize*

$$\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))$$

with

$$f_k(\boldsymbol{x}) = \begin{cases} \sum_{i=1}^{2n/m} \Pi_{j=1}^{i} \boldsymbol{x}_{j+n(k-1)/m}, & \text{if } k \text{ is odd,} \\ \sum_{i=1}^{2n/m} \Pi_{j=i}^{2n/m} (1 - \boldsymbol{x}_{j+n(k-2)/m}), & \text{else.} \end{cases}$$

Before introducing the mOneJumpZeroJump problem, we first introduce the single-objective Jump problem, which aims at maximizing the number of 1-bits of a solution except for a valley around the solution with all 1-bits. Formally, the Jump problem of size n aims to find an n-bits binary string which maximizes

$$\text{Jump}_{n,k}(\boldsymbol{x}) = \begin{cases} k + |\boldsymbol{x}|_1, & \text{if } |\boldsymbol{x}|_1 \leq n - k \text{ or } |\boldsymbol{x}|_1 = n, \\ n - |\boldsymbol{x}|_1, & \text{else,} \end{cases}$$

where $2 \leq k \leq n - 1$ is a parameter and $|\boldsymbol{x}|_1$ denotes the number of 1-bits in $\boldsymbol{x}$. The mOneJumpZeroJump problem presented in Definition 5 also divides a solution

into $m/2$ blocks, and in each block, it tries to optimize a Jump problem as well as a counterpart of Jump problem with the roles of 1-bits and 0-bits exchanged. The Pareto front is $F^* = \{(i_1, 2n/m + 2k - i_1, \cdots, i_{m/2}, 2n/m + 2k - i_{m/2}) \mid i_1, \cdots i_{m/2} \in [2k..2n/m] \cup \{k, 2n/m + k\}\}$ whose size is $(2n/m - 2k + 3)^{m/2}$, and the Pareto optimal solution corresponding to $(i_1, 2n/m + 2k - i_1, \cdots, i_{m/2}, 2n/m + 2k - i_{m/2})$ is the solution with $(i_j - k)$ 1-bits and $(2n/m - i_j + k)$ 0-bits in the j-th block.

Definition 5. *(mOneJumpZeroJump [48]) Suppose m is a positive even number, and n is a multiple of $m/2$. The mOneJumpZeroJump$_{n,k}$ problem of size n is to find n-bits binary strings which maximize*

$$\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))$$

with

$$f_i(\boldsymbol{x}) = \begin{cases} \textit{Jump}_{2n/m,k}(\boldsymbol{x}_{[n(i-1)/m+1..n(i+1)/m]}), & \textit{if } i \textit{ is odd}, \\ \textit{Jump}_{2n/m,k}(\bar{\boldsymbol{x}}_{[n(i-2)/m+1..ni/m]}), & \textit{else}, \end{cases}$$

where $\bar{\boldsymbol{x}} = (1 - x_1, \cdots, 1 - x_n)$.

3 General Theorems for Running Time Analysis of MOEAs

In this section, we present Theorems 1, 2 and 3 that can be used to derive expected running time of general MOEAs for solving the mOneMinMax, mLeadingOnesTrailingZeroes and mOneJumpZeroJump problems, respectively. These results will be applied in Sect. 4 to derive the running time bounds of SPEA2 and applied in Sect. 5 to derive the running time bounds of other MOEAs. In our analysis, we will use the general concept of a MOEA *preserving the non-dominated set*, which is defined as follows:

Definition 6. *If a non-dominated solution $\boldsymbol{x}$ appears in the combined population of parent and offspring, then there will always be a solution $\boldsymbol{y}$ in the next generation such that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{y})$.*

For different MOEAs, the proportion between the parent and offspring populations varies. We assume that the parent population size is μ and the offspring population size is $c\mu$, where $c \in [1/\mu, O(1)]$. For algorithms like SEMO, GSEMO, and SMS-EMOA, only one solution is generated in each iteration, thus $c = 1/\mu$, and for algorithms like NSGA-II and NSGA-III, $c = 1$. Note that the running time of EAs is measured by the number of fitness evaluations, which is the most time-consuming step in the evolutionary process.

3.1 On the mOneMinMax Problem

We prove in Theorem 1 that the expected number of fitness evaluations for any MOEA solving mOneMinMax is $O(\mu n \cdot \min\{m \log n, n\})$, if the algorithm uses uniform selection and bit-wise mutation or one-bit mutation to generate offspring solutions, and can preserve the non-dominated set. The proof idea is as follows. First, we show that the

probability of not finding a specific Pareto front point in $O(m\mu n \log n)$ and $O(\mu n^2)$ fitness evaluations is at most n^{-m} and e^{-n}, respectively. Then, we use the union bound to show that the probability of finding all Pareto front points in $O(\mu n \cdot \min\{m \log n, n\})$ fitness evaluations is $1 - o(1)$. Note that this proof idea is inspired by Theorem 5.2 in [36], which analyzes NSGA-III solving mOneMinMax, and uses the union bound to derive the probability of finding the whole Pareto front within T generations, after deriving the probability of not finding a specific Pareto front point within T generations. Our proof differs by considering more general scenarios, such as varying proportions of parent and offspring populations and a larger number m of objectives.

Theorem 1. *For any MOEA solving mOneMinMax, if the algorithm preserves the non-dominated set with a maximum population size of μ, employs uniform selection to select parent solutions, and employs bit-wise mutation or one-bit mutation to generate offspring solutions, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu n \cdot \min\{m \log n, n\})$.*

Proof. For the mOneMinMax problem, all the solutions are Pareto optimal, and thus are non-dominated. Since the MOEA preserves the non-dominated set, any objective vector will be preserved in the population once it is found.

First, we prove that for any objective vector $\boldsymbol{v}$, the probability of not finding it in $O(m\mu n \log n)$ fitness evaluations is at most n^{-m}. For any solution $\boldsymbol{x}$, we partition it into $m/2$ blocks, where the i-th ($i \in [1..m/2]$) block $B_i = [(i-1)(2n/m + 1) \dots i(2n/m)]$. When there is a bit flip in block $\boldsymbol{x}_{B_i}$, one of $f_{2i-1}(\boldsymbol{x})$ and $f_{2i}(\boldsymbol{x})$ will increase by 1, and the other one will decrease by 1. Therefore, $\boldsymbol{x}$ needs to flip at least $\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{v}\|_1 /2$ bits to obtain $\boldsymbol{v}$. Let $d_{\boldsymbol{v}} = \min_{\boldsymbol{x} \in P} \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{v}\|_1 /2$ denote the minimum number of bits that require to be flipped to obtain $\boldsymbol{v}$ for all the solutions in the population P. Since $d_{\boldsymbol{v}} = \min_{\boldsymbol{x} \in P} \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{v}\|_1 /2 \leq \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{v}\|_1 /2 \leq \|2\boldsymbol{v}\|_1 /2 = n$, and $d_{\boldsymbol{v}} = 0$ when there exists an $\boldsymbol{x} \in P$ such that $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{v}$, we have $0 \leq d_{\boldsymbol{v}} \leq n$. Since all the objective vectors will be preserved, $d_{\boldsymbol{v}}$ cannot increase. Let the random variable $X_k, k \in [1..n]$ denote the number of generations with $d_{\boldsymbol{v}} = k$. Let $X := \sum_{k=1}^{n} X_k$. When $d_{\boldsymbol{v}} = k$, the probability of selecting an individual $\boldsymbol{y}$ from the population such that $\|\boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{v}\|_1 /2 = k$ is at least $1/\mu$. Since flipping any one of the k bits corresponding to $d_{\boldsymbol{v}}$ will reduce $d_{\boldsymbol{v}}$ by 1, the probability that $d_{\boldsymbol{v}}$ decreases is at least $(k/n) \cdot (1 - 1/n)^{n-1} \geq k/(en)$ when using bit-wise mutation and at least k/n when using one-bit mutation. In the following, we consider bit-wise mutation, while the analysis for one-bit mutation holds analogously. In each generation, the probability that $d_{\boldsymbol{v}}$ decreases is at least

$$1 - \left(1 - \frac{k}{e\mu n}\right)^{c\mu} \geq 1 - e^{-\frac{ck}{en}} \geq \frac{ck}{ck + en} \geq \frac{ck}{(c+e)n}, \tag{1}$$

where $c\mu$ is the size of the offspring population, the second inequality holds by $1 + a \leq e^a$ for any $a \in \mathbb{R}$, and the third inequality holds by $k \leq n/m$. Let $p_k := ck/(c+e)n$. Hence, $X_1, \dots, X_n$ stochastically dominates independent geometric random variables $Y_1, \dots, Y_n$, where the success probability of Y_k is p_k. That is for any $\lambda \geq 0$, we have $\mathrm{P}(X_k \geq \lambda) \leq \mathrm{P}(Y_k \geq \lambda)$. Moreover, for $Y := \sum_{k=1}^{n} Y_k$, X stochastically dominates

Y. By Theorem 16 in [16] we have:

$$\mathrm{P}\Big(X \geq \frac{e+c}{c}(1+m)n\log n\Big) \leq \mathrm{P}\Big(Y \geq \frac{e+c}{c}(1+m)n\log n\Big) \leq n^{-m}. \quad (2)$$

In each generation, the algorithm produces $c\mu$ offspring solutions, thus the probability that the population contains no solution $\boldsymbol{x}$ with $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{v}$ after $c\mu \cdot (e+c)(1+m)n(\log n)/c = (e+c)(1+m)\mu n \log n$ fitness evaluations is at most n^{-m}.

Next, we prove that for any objective vector $\boldsymbol{v}$, the probability of not finding it in $O(\mu n^2)$ fitness evaluations is at most e^{-n}. Let $p'_k := ck/(ck+en)$. By Eq. (1), we have $X_1, \ldots, X_n$ stochastically dominates independent geometric random variables $Z_1, \ldots, Z_n$, where the success probability of Z_k is p'_k. Applying Theorem 1 in [42] to the variable $Z := \sum_{k=1}^{n} Z_k$, we have:

$$\mathrm{P}\Big(X \geq \frac{5en^2}{c}\Big) \leq \mathrm{P}\Big(Z \geq \frac{5en^2}{c}\Big) =\leq e^{-\frac{1}{4}\min\{\frac{\delta^2}{s}, \frac{c\delta}{c+en}\}} \leq e^{-n}, \quad (3)$$

where $\delta = 5en^2/c - en\log n/c - n$, $E[Z] = en\log n/c + n$ and $s = \sum_{k=1}^{n} 1/p'^2_k = n + 2en\log n/c + (e\pi n)^2/6c$. Thus, the probability that the population contains no solution $\boldsymbol{x}$ with $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{v}$ after $c\mu \cdot 5en^2/c = 5e\mu n^2$ fitness evaluations is at most e^{-n}.

Finally, we consider finding the whole Pareto front. Recall that for the mOneMinMax problem, the size of the Pareto front is $(2n/m+1)^{m/2}$. Then, by applying the union bound, the probability of finding the whole Pareto front in $(e+c)(1+m)\mu n\log n$ number of fitness evaluations is at least $1 - (2n/m+1)^{m/2} \cdot n^{-m} \geq 1 - (n+1)^{m/2} \cdot n^{-m} = 1 - O(n^{-m/2}) = 1 - o(1)$, and the probability of finding the whole Pareto front in $5e\mu n^2$ number of fitness evaluations is at least $1 - (2n/m+1)^{m/2} \cdot e^{-n} \geq 1 - 3^{n/2} \cdot e^{-n} = 1 - o(1)$. The inequality holds because the derivative of $(2n/m+1)^{m/2}$ with respect to m is always positive when $m \leq n$, implying that when $m = n$, $(2n/m+1)^{m/2}$ takes its maximum value, which is $3^{n/2}$. Combining the two parts, the whole Pareto front can be found in $\min\{(e+c)(1+m)\mu n\log n, 5e\mu n^2\}$ number of fitness evaluations with probability $1 - o(1)$. Let each $\min\{(e+c)(1+m)\mu n\log n, 5e\mu n^2\}$ fitness evaluations to find Pareto front be an independent trial with success probability of $1 - o(1)$. Then, the expected number of evaluations to find the Pareto front is at most $(1-o(1))^{-1}\min\{(e+c)(1+m)\mu n\log n, 5e\mu n^2\} = O(\mu n \cdot \min\{m\log n, n\})$. □

Hence, if the MOEA preserves the non-dominated set, the expected number of fitness evaluations for solving mOneMinMax is $O(\mu n \cdot \min\{m\log n, n\})$. Specifically, when m is a constant, $O(\min\{m\log n, n\}) = O(m\log n)$, leading to an expected number of fitness evaluations of $O(\mu n\log n)$. Conversely, if the number m of objectives is large, e.g., $m = n/4$, the expected number of fitness evaluations is $O(\mu n^2)$.

3.2 On the mLeadingOnesTrailingZeroes Problem

We prove in Theorem 2 that the expected number of fitness evaluations for any MOEA solving mLeadingOnesTrailingZeroes is $O(\mu n^2)$. The proof idea is to divide the optimization procedure into two phases, where the first phase aims at finding a point in the

Pareto front. Let $w_i(\boldsymbol{x}) := f_{2i-1}(\boldsymbol{x}) + f_{2i}(\boldsymbol{x})$ denote the sum of leading ones and trailing zeroes of block $\boldsymbol{x}_{B_i}$. Let $W_{\max}$ denote $\max_{\boldsymbol{x}\in P} \sum_{i=1}^{m/2} w_i(\boldsymbol{x})$. We prove that the expected number of fitness evaluations for $W_{\max}$ to reach n (i.e., a Pareto front point is found) is at most $O(\mu n^2)$. The second phase aims at finding the whole Pareto front extended from the Pareto front point found in the first phase. Similar to the analysis on mOneMinMax, we prove that the probability of not finding a specific point in the Pareto front within $6e\mu n^2$ number of fitness evaluations is at most e^{-n} by applying Theorem 1 in [42]; by the union bound, the probability of finding the whole Pareto front within $O(\mu n^2)$ number of fitness evaluations is at least $1 - (2n/m + 1)^{m/2} \cdot e^{-n} = 1 - o(1)$. Since the proof is similar to that of Theorem 4.3 in [36], and also resembles that of Theorem 1, we omit the detailed proof here.

Theorem 2. *For any MOEA solving mLeadingOnesTrailingZeroes, if the algorithm preserves the non-dominated set with a maximum population size of μ, employs uniform selection to select parent solutions, and employs bit-wise mutation or one-bit mutation to generate offspring solutions, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu n^2)$.*

Theorem 2 shows that for any MOEA preserving the non-dominated set, the expected number of fitness evaluations for solving the mLeadingOnesTrailingZeroes problem is $O(\mu n^2)$. Notably, although this result seems unrelated to the number m of objectives, the Pareto front size of mLeadingOnesTrailingZeroes (i.e., $(2n/m+1)^{m/2}$) amplifies with increasing m. Since the population size μ needs to be larger than $(2n/m + 1)^{m/2}$ to preserve the non-dominated set, the running time ultimately rises with the number m of objectives.

3.3 On the mOneJumpZeroJump Problem

We prove in Theorem 3 that the expected number of fitness evaluations for any MOEA solving mOneJumpZeroJump is $O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$. By Definition 5, if a solution $\boldsymbol{x}$ is a Pareto optimal, then for any block B_i, $|\boldsymbol{x}_{B_i}|_1 \in \{0, n'\} \cup [k..n'-k]$, where $n' = 2n/m$ denotes the size of each block. We call $\boldsymbol{x}$ an *internal* Pareto optimum if for any block B_i, $|\boldsymbol{x}_{B_i}|_1 \in [k..n'-k]$, and we call $\boldsymbol{x}$ an *extreme* Pareto optimum if there exists a block B_i such that $|\boldsymbol{x}_{B_i}|_1 \in \{0, n'\}$. The proof idea is to divide the optimization procedure into three phases, where the first phase aims at finding an internal Pareto front point, the second phase finishes after finding the whole internal Pareto front whose analysis is similar to that of finding the Pareto front of mOneMinMax, and the third phase focuses on finding the extreme Pareto front points from the edge of the internal Pareto front. Due to space limitation, we omit some details of the proof, which are provided in the supplementary material.

Theorem 3. *For any MOEA solving mOneJumpZeroJump$_{n,k}$, if the algorithm preserves the non-dominated set with a maximum population size of μ, employs uniform selection to select parent solutions, and employs bit-wise mutation to generate offspring solutions, the expected number of fitness evaluations for finding the Pareto front is $O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$.*

Proof. We divide the optimization procedure into three phases. The first phase starts after initialization and finishes until an internal point in the Pareto front is found; the second phase starts after the first phase and finishes until all internal points in the Pareto front are found; the third phase starts after the second phase and finishes when all extreme points in the Pareto front are found.

The first phase is to find an internal Pareto front point. If the initial population contains an internal Pareto optimum, the first phase is completed. We consider that the initial population does not contain an internal Pareto optimum. Then, for any solution $\boldsymbol{x}$ in the population P, there exists a block $B_i = [(i-1)n'+1..in']$, $i \in [1..m/2]$ such that $|\boldsymbol{x}_{B_i}|_1 \in [0..k-1] \cup [n'-k+1..n']$. Without loss of generality, assume that $|\boldsymbol{x}_{B_i}|_1 \in [0..k-1]$. Then the probability of generating a solution $\boldsymbol{y}$ with $|\boldsymbol{y}_{B_i}|_1 \in [k, n'-k]$ from $\boldsymbol{x}$ is at least $\binom{n'-|\boldsymbol{x}_{B_i}|_1}{k-|\boldsymbol{x}_{B_i}|_1}(1-\frac{1}{n})^{n-(k-|\boldsymbol{x}_{B_i}|_1)}(\frac{1}{n})^{k-|\boldsymbol{x}_{B_i}|_1} \geq \frac{1}{e}(\frac{1}{mk})^k$, where the inequality holds by $k \leq n'/2$. The same bound also applies to $|\boldsymbol{x}_{B_i}|_1 \in [n'-k+1..n']$. Let $J(\boldsymbol{x})$ denote the number of blocks B_i such that $|\boldsymbol{x}_{B_i}|_1 \in [k..n'-k]$, and let $J_{\max} := \max_{\boldsymbol{x}\in P} J(\boldsymbol{x})$. For a solution $\boldsymbol{x}^*$ with $J(\boldsymbol{x}^*) = J_{\max}$, it will not be dominated by any solution $\boldsymbol{y}$ with $J(\boldsymbol{y}) < J_{\max}$. Thus, $J_{\max}$ will not decrease. When $J_{\max}$ reaches $m/2$, an internal Pareto front point is found. The probability of selecting the solution $\boldsymbol{x}^*$ is $1/\mu$, and the probability of increasing $J_{\max}$ by 1 through bit-wise mutation is at least $\frac{1}{e}(\frac{1}{mk})^k$ as analysed above. Therefore, in each generation, the probability of increasing $J_{\max}$ by 1 is at least $1-(1-\frac{1}{e\mu}(\frac{1}{mk})^k)^{c\mu} \geq 1-e^{-\frac{c}{e}(1/mk)^k} \geq c/(c+e(mk)^k)$, where the inequalities hold by $1+a \leq e^a$ for any $a \in \mathbb{R}$. In each generation, the population produces $c\mu$ offspring solutions, thus the expected number of fitness evaluations of the first phase is at most $(m/2) \cdot c\mu \cdot (c+e(mk)^k)/c = O(\mu m^{k+1}k^k)$.

The second phase is to find all internal Pareto front points. The analysis of the second phase is similar to that of mOneMinMax. After finding an internal Pareto front point, we first prove that the probability of not finding a specific internal Pareto front point within $O(\mu n^2)$ fitness evaluations is at most e^{-n}. For any internal Pareto optimum $\boldsymbol{x}$, $f_{2i-1}(\boldsymbol{x}) + f_{2i}(\boldsymbol{x}) = 2k + n'$, and the value range is $[2k, n']$. Hence, the size of the internal Pareto front is $(n'-2k+1)^{m/2}$. By the union bound, the probability of finding the whole internal Pareto front within $O(\mu n^2)$ fitness evaluations is at least $1-(n'-2k+1)^{m/2} \cdot e^{-n} = 1-o(1)$.

The third phase is to find all extreme Pareto front points. Note that for any Pareto optimum $\boldsymbol{x}$, if there exists a block B_i such that $|\boldsymbol{x}_{B_i}|_1 \in \{0, n'\}$, then we call $\boldsymbol{x}$ an extreme Pareto optimum and $\boldsymbol{x}_{B_i}$ an extreme block. Now, we use two bounds to show the running time of the third phase.

First, we show that the expected number of fitness evaluations for finding all the extreme points in the Pareto front is $O(3^{m/2} \cdot \mu n^k)$. We divide the optimization procedure of the third phase into $m/2$ levels, and each solution in the i-th level contains i extreme blocks. We first consider the running time of finding all the extreme Pareto front points at level 1. After the second phase, all internal Pareto front points have been found. Hence, for any block B_i, there exist at least $\sigma_1 := (n'-2k+1)^{m/2-1}$ solutions $\boldsymbol{x}$ such that $|\boldsymbol{x}_{B_i}|_1 = k$. The probability of selecting such a solution as parent is at least σ_1/μ, and the probability of flipping k 1-bits to obtain $\boldsymbol{y}$ such that $|\boldsymbol{y}_{B_i}|_1 = 0$ is $(1/n^k) \cdot (1-1/n)^{n-k} \geq 1/(en^k)$. The same result also applies for $|\boldsymbol{x}_{B_i}|_1 = n'-k$ to generate a solution $\boldsymbol{y}$ such that $|\boldsymbol{y}_{B_i}|_1 = n'$. Therefore, in each generation, the probabil-

ity of generating an extreme block is at least $1-(1-\sigma_1/(e\mu n^k))^{c\mu} \geq 1-e^{-c\sigma_1/(en^k)} \geq c\sigma_1/(c\sigma_1+en^k)$, where the inequalities holds by $1+a \leq e^a$ for any $a \in \mathbb{R}$. Thus, the expected number of fitness evaluations for generating an extreme block is at most $c\mu \cdot (c\sigma_1+en^k)/c\sigma_1 = O(\mu n^k/\sigma_1)$. Then, we consider finding all the solutions at level 1 where B_i is an extreme block. This optimization process is similar to the second phase, and the difference is no need to consider the block B_i. Therefore, the expected number of fitness evaluations of this process is less than the second phase, which is $O(\mu n^2)$. Because there can be $m/2$ positions with extreme blocks, and each extreme block has 2 possible forms $0^{n'}$ and $1^{n'}$, the expected number of evaluations for finding all level 1 solutions is $O(m/2 \cdot 2 \cdot \mu n^k/\sigma_1) + O(m/2 \cdot 2 \cdot \mu n^2) = O(m\mu(n^k/\sigma_1 + n^2))$. For other levels, the proof idea is similar, and the expected number of fitness evaluations for finding all the Pareto front points at level i is $O(\binom{m/2}{i} \cdot 2^i \mu(n^k/\sigma_i + n^2))$, where $\sigma_i := (n'-2k+1)^{m/2-i}$. Sum up the time from level 1 to level $m/2$, we have $O(\sum_{i=1}^{m/2} \binom{m/2}{i} \cdot 2^i \mu(n^k/\sigma_i + n^2)) = O((2+1/(n'-2k+1))^{m/2}\mu n^k + 3^{m/2}\mu n^2)$. Since $k \leq n'/2$, the inequality $1/(n'-2k+1) \leq 1$ holds, implying that the expected number of fitness evaluations for finding the extreme Pareto front is at most $O(3^{m/2}n^k)$.

Then we show that after the second phase, all the extreme Pareto front points can be found in $O(m\mu n^{k+1})$ number of evaluations with probability $1-e^{-\Omega(mn)}$. After the second phase, all internal Pareto front points have been found. Assume that a Pareto front point $\boldsymbol{v}$ containing d extreme values (the objective value is k or $n'+k$) has not been found, then there exists a solution $\boldsymbol{x} \in P$ such that except for the d extreme values of $\boldsymbol{v}$, the remaining objective values are all equal to $\boldsymbol{v}$. Then, $\boldsymbol{v}$ could be extended from $\boldsymbol{x}$ by generating $d/2$ extreme blocks. Let the random variable $X_k, 1 \leq k \leq m/2$, denote the number of generations to generate the k-th extreme block. Let $X := \sum_{k=1}^{m/2} X_k$. By the analysis of extreme blocks, the probability of generating an extreme block in each generation is at least $c/(c+en^k)$. Similar to the analysis of Eq. (3), by Theorem 1 in [42], we have $P(X \geq mn^{k+1}/c) \leq e^{-\Omega(mn)}$. Thus, the probability that the population contains no solution $\boldsymbol{x}$ with $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{v}$ in $c\mu \cdot (mn^{k+1}/c) = m\mu n^{k+1}$ number of fitness evaluations is at most $e^{-\Omega(mn)}$. By the union bound, the probability of finding all the extreme Pareto front points within $m\mu n^{k+1}$ number of fitness evaluations is at most $1-((n'-2k+3)^{m/2}-(n'-2k+1)^{m/2}) \cdot e^{-\Omega(mn)} \geq 1-n^{m/2}e^{-\Omega(mn)} = 1-o(1)$. Finally, combining the two bounds, the third phase needs $O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$ expected number of fitness evaluations.

Combining the three phases, the upper bound on the expected number of fitness evaluations for finding the whole Pareto front is $O(\mu m^{k+1}k^k + \mu n^2 + \mu n^k \cdot \min\{mn, 3^{m/2}\}) = O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$. □

Theorem 3 shows that for any MOEA preserving the non-dominated set, the expected number of fitness evaluations for solving the mOneJumpZeroJump problem is $O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$. Note that when m is a constant, the expected number of fitness evaluations is $O(\mu n^k)$. Conversely, if the number m of objectives is large, e.g., $m = n/4$, the expected number of fitness evaluations is $O(\mu n^{k+2})$.

4 Application to Running Time Analysis of SPEA2

In this section, we first show that when the archive size of SPEA2 is large enough, SPEA2 can preserve the non-dominated set. Then, we apply the results in the previous section, i.e., Theorems 1, 2 and 3, to derive the running time of SPEA2.

4.1 Large Archive Preserves Non-Dominated Solutions

Throughout the process of SPEA2, the next population P consists of the mutated individuals from the archive A. That is, the archive A in SPEA2 actually plays the role of population, while the population P acts as offspring. We now show that when the archive size $\bar{\mu}$ is always at least the maximum cardinality of the set of non-dominated solutions having different objective vectors for any m-objective function $\boldsymbol{f}$, SPEA2 will preserve the non-dominated set on $\boldsymbol{f}$.

Lemma 4. *For SPEA2 solving $\boldsymbol{f}$, let S denote the non-dominated set of solutions in the union of the population P and archive A. If the size $\bar{\mu}$ of archive is always no less than $|\boldsymbol{f}(S)|$, then SPEA2 preserves the non-dominated set.*

Proof. Consider that all the non-dominated solutions in $P \cup A$ have been added to A' (line 4 of Algorithm 1). If the size of A' is at most $\bar{\mu}$, then all the non-dominated solutions will survive to the next generation. Thus, the lemma holds. If the size of A' is larger than $\bar{\mu}$ (i.e., $|S| > \bar{\mu}$), SPEA2 will use the truncation operator to remove the redundant solutions one by one, until $|A'| = \bar{\mu}$. Let $D_{\boldsymbol{z}} = \{\boldsymbol{y} \in A' \mid \boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{f}(\boldsymbol{y})\}$. Since $\bar{\mu}$ is at least $|\boldsymbol{f}(S)|$ and $|S| > \bar{\mu}$, there exists $\boldsymbol{z} \in A'$ such that $|D_{\boldsymbol{z}}| > 1$. Note that $\sigma^1_{\boldsymbol{x}} > 0$ if $|D_{\boldsymbol{x}}| = 1$, and $\sigma^1_{\boldsymbol{x}} = 0$ if $|D_{\boldsymbol{x}}| > 1$. Hence, if $|D_{\boldsymbol{x}}| = 1$, then $\sigma^1_{\boldsymbol{x}} > 0 = \sigma^1_{\boldsymbol{z}}$, implying that $\boldsymbol{z} \preceq_d \boldsymbol{x}$. Thus, $\boldsymbol{z}$ will be removed before $\boldsymbol{x}$, which implies that the non-duplicated solutions (regarding the objective vectors) will not be removed from A'. Therefore, the lemma holds. □

By Lemma 4, we can set the size of the archive A accordingly, so that SPEA2 preserves the non-dominated set, which then allows us to apply Theorems 1 to 3 to derive the expected running time of SPEA2.

4.2 Running Time of SPEA2 on Benchmark Problems

Let S denote the set of non-dominated solutions in the union of the population P and archive A. In the following, we will give the maximum values of $|\boldsymbol{f}(S)|$ when SPEA2 solves mOneMinMax, mLeadingOnesTrailingZeroes and mOneJumpZeroJump, respectively, so as to determine the size of the archive. Then, by applying Lemma 4, we provide the expected number of fitness evaluations for SPEA2 finding the Pareto front.

Theorem 5. *For SPEA2 solving mOneMinMax, if the size $\bar{\mu}$ of the archive is at least $(2n/m + 1)^{m/2}$, uniform selection is employed to select parent solutions, and bit-wise mutation or one-bit mutation is employed to generate offspring solutions, the expected number of fitness evaluations for finding the Pareto front is $O(\mu n \cdot \min\{m \log n, n\})$.*

Proof. Because any solution is Pareto optimal, $|\boldsymbol{f}(S)|$ is always at most the size of the Pareto front. By Definition 3, the size of the Pareto front is $(2n/m+1)^{m/2}$, implying that $|\boldsymbol{f}(S)| \leq (2n/m+1)^{m/2}$. According to Lemma 4 and Theorem 1, if the size $\bar{\mu}$ of the archive in SPEA2 is at least $(2n/m+1)^{m/2}$, the whole Pareto front can be found in $O(\bar{\mu}n \cdot \min\{m \log n, n\})$ expected number of fitness evaluations. □

Theorem 6. *For SPEA2 solving mLeadingOnesTrailingZeroes, if the size $\bar{\mu}$ of the archive is at least $(2n/m+1)^{m-1}$, uniform selection is employed to select parent solutions, and bit-wise mutation or one-bit mutation is employed to generate offspring solutions, the expected number of fitness evaluations for finding the Pareto front is $O(\mu n^2)$.*

Proof. By Lemma 4.2 in [36], the maximum cardinality of the set of non-dominated solutions having different objective vectors is at most $(2n/m+1)^{m-1}$ for mLeadingOnes-Trailingzeroes, implying $|\boldsymbol{f}(S)| \leq (2n/m+1)^{m-1}$. According to Lemma 4 and Theorem 2, if the size $\bar{\mu}$ of the archive in SPEA2 is no less than $(2n/m+1)^{m-1}$, the whole Pareto front can be found in $O(\bar{\mu}n^2)$ expected number of fitness evaluations. □

Theorem 7. *For SPEA2 solving mOneJumpZeroJump, if the size $\bar{\mu}$ of the archive is at least $(2n/m-2k+3)^{m/2}$, uniform selection is employed to select parent solutions, and bit-wise mutation is employed to generate offspring solutions, the expected number of fitness evaluations for finding the Pareto front is $O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$.*

Proof. By Lemma 3 in [48], if $k \leq n'/2$, the maximum cardinality of the set of non-dominated solutions having different objective vectors is at most $(2n/m+1)^{m/2}$ for mOneJump-ZeroJump, implying $|\boldsymbol{f}(S)| \leq (2n/m-2k+3)^{m/2}$. According to Lemma 4 and Theorem 3, if the archive size $\bar{\mu} \geq (2n/m-2k+3)^{m/2}$, the Pareto front can be found in $O(\mu n^k \cdot \min\{mn, 3^{m/2}\})$ expected fitness evaluations. □

5 Application to Other Algorithms

In this section, we show that Theorems 1, 2 and 3 can be applied to other MOEAs (including SEMO, NSGA-II, SMS-EMOA, etc.) to derive their expected running time, which are consistent with previous results.

mOneMinMax: For the bi-objective OneMinMax problem (i.e., mOneMinMax with $m = 2$), Giel and Lehre proved that the expected number of fitness evaluations for SEMO finding the Pareto front is $O(\mu n^2) = O(n^3)$, where the equality holds by $\mu \leq n+1$. Zheng *et al.* [46] proved that for a population size μ exceeding 4 times the Pareto front size (i.e., $\mu \geq 4(n+1)$), NSGA-II preserves the non-dominated set. the Pareto front can be found in $O(\mu n \log n) = O(n^2 \log n)$ expected number of fitness evaluations. Similarly, Bian *et al.* [6] and Nguyen *et al.* [35] proved that when the population size is greater than the Pareto front size (i.e., $n+1$), both SMS-EMOA and $(\mu+1)$SIBEA preserve the non-dominated set on OneMinMax, resulting in an expected number of fitness evaluations of $O(\mu n \log n)$. These results are all consistent with Theorem 1. For mOneMinMax with large constant m, Andre *et al.* [36] proved that NSGA-III preserves the non-dominated set, if it employs a set of reference points

$\mathcal{R}_p$ with $p \geq 4n\sqrt{m}$. With a population size of $\mu \geq (2n/m+1)^{m/2}$, the expected number of fitness evaluations is $O(\mu n \log n)$, which is consistent with Theorem 1.

mLeadingOnesTrailingZeroes: For the bi-objective LeadingOnesTrailingZeroes problem (i.e., mLeadingOnesTrailingZeroes with $m = 2$), Laumanns [30] and Giel [26] proved that the expected number of fitness evaluations for SEMO and GSEMO finding the Pareto front is $O(n^3)$. Brockhoff [7] proved that when the population size is no less than the Pareto front size (i.e., $n + 1$), $(\mu + 1)$SIBEA preserves the non-dominated set on LeadingOnesTrailingZeroes, the Pareto front can be found in $O(\mu n^2)$ expected number of fitness evaluations. Zheng *et al.* [46] proved that for the population size at least $4(n + 1)$, NSGA-II preserves the non-dominated set and the expected number of fitness evaluations is also $O(\mu n^2) = O(n^3)$. These results are all consistent with Theorem 2. For $m \geq 4$, Laumanns [30] proved that the expected running time of SEMO is $O(n^{m+1})$, which is consistent with the result in Theorem 2, because the maximal population size before finding the Pareto front is at most $(2n/m + 1)^{m-1} \leq n^{m-1}$. Andre *et al.* [36] proved that NSGA-III preserves the non-dominated set on mLeadingOnesTrailingZeroes, if it employs a set of reference points $\mathcal{R}_p$ with $p \geq 4n\sqrt{m}$. With a population size of $\mu \geq (2n/m + 1)^{m-1}$, the expected number of fitness evaluations is $O(\mu n^2)$, which is consistent with Theorem 2.

mOneJumpZeroJump: For the bi-objective OneJumpZeroJump problem (i.e., m-OneJumpZeroJump with $m = 2$), Doerr *et al.* [22] proved that the expected number of fitness evaluations for GSEMO finding the Pareto front is $O(\mu n^k) = O((n - 2k)n^k)$, where the equality holds by $\mu \leq n - 2k + 3$. When the population size is at least $4(n-2k+3)$ for NSGA-II [19] and $(n-2k+3)$ for SMS-EMOA [6], both algorithms preserve the non-dominated set on mOneJumpZeroJump and can find the Pareto front within $O(\mu n^k)$ expected number of fitness evaluations. These results are all consistent with Theorem 3. For $m \leq \log n$, Zheng and Doerr [48] proved that when the population size is at least $(2n/m - 2k + 3)^{m/2}$, SMS-EMOA preserves the non-dominated set and can find the Pareto front within $O(M\mu n^k)$ expected number of fitness evaluations, where $M = (2n/m - 2k + 3)^{m/2}$, while Theorem 3 shows that any MOEA that can preserve the non-dominated set can find the Pareto front in $O(3^{m/2}\mu n^k)$ expected number of fitness evaluations, implying a tighter bound.

6 Conclusion and Discussion

In this paper, we give general theorems for analyzing the expected running time of MOEAs on the three common benchmark problemes, mOneMinMax, mLeadingOnes-Trailingzeroes and mOneJumpZeroJump. These theorems show that if the population of MOEAs preserves the objective vectors of non-dominated solutions, their running time can be upper bounded. We apply them to derive the expected running time of SPEA2 for the first time, and also to analyze other MOEAs, deriving results consistent with previously known ones. We hope these theorems can be useful for theoretical analysis of MOEAs in the future. The theorems also suggest that when the population size is large enough, the analytical behavior of MOEAs may tend to be similar. Thus, it would be interesting to analyze the approximation performance of MOEAs with small populations, to better understand their differences.

In a parallel theoretical work, Wietheger and Doerr [41] proved near-tight running time guarantees for GSEMO on mOneMinMax, mCountingOnesCountingZeroes, mLeadingOnesTrailingZeroes, and mOneJumpZeroJump, and also transferred similar bounds to SEMO, SMS-EMOA, and NSGA-III. Their proof techniques are superior and the bounds are tighter, as they not only considered the probability p of not finding any Pareto front point within T generations but also refined this process by considering each block individually. They considered the probability p' of not achieving the optimization goal for each block within the given time. Using the union bound, they showed $p \leq m'p'$, where m' is the number of blocks. Let M be the size of the Pareto front. Then, the probability of finding the entire Pareto front within T generations is at least $1 - Mp \geq 1 - Mm'p'$. Their analysis of the optimization of blocks using the union bound allowed them to obtain tighter time bounds.

Acknowledgments. The authors want to thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Science and Technology Major Project (2022ZD0116600) and National Science Foundation of China (62276124). Chao Qian is the corresponding author. The supplementary is available at arXiv.

References

1. Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, Oxford (1996)
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. Bian, C., Qian, C.: Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In: Proceedings of the 17th International Conference on Parallel Problem Solving from Nature (PPSN 2022), Dortmund, Germany, pp. 428–441 (2022)
4. Bian, C., Qian, C., Tang, K.: A general approach to running time analysis of multi-objective evolutionary algorithms. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018), Stockholm, Sweden, pp. 1405–1411 (2018)
5. Bian, C., Ren, S., Li, M.Q., Qian, C.: An archive can bring provable speed-ups in multi-objective evolutionary algorithms. In: Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI 2024), Jeju Island, South Korea (2024, to appear)
6. Bian, C., Zhou, Y., Li, M., Qian, C.: Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In: Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023), Macao, SAR, China, pp. 2191–2197 (2023)
7. Brockhoff, D., Friedrich, T., Neumann, F.: Analyzing hypervolume indicator based algorithms. In: Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN 2008), Dortmund, Germany, pp. 651–660 (2008)
8. Cerf, S., Doerr, B., Hebras, B., Kahane, Y., Wietheger, S.: The first proven performance guarantees for the non-dominated sorting genetic algorithm II (NSGA-II) on a combinatorial optimization problem. In: Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023), Macao, SAR, China, pp. 5522–5530 (2023)
9. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-objective Problems. Springer, New York (2007). https://doi.org/10.1007/978-0-387-36797-2

10. Covantes Osuna, E., Gao, W., Neumann, F., Sudholt, D.: Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. Theoret. Comput. Sci. **832**, 123–142 (2020)
11. Dang, D.C., Opris, A., Salehi, B., Sudholt, D.: Analysing the robustness of NSGA-II under noise. In: Proceedings of the 25th ACM Conference on Genetic and Evolutionary Computation (GECCO 2023). Lisbon, Portugal, pp. 642–651 (2023)
12. Dang, D.C., Opris, A., Salehi, B., Sudholt, D.: A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023), Washington, DC, pp. 12390–12398 (2023)
13. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
14. 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)
15. 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)
16. Doerr, B.: Analyzing randomized search heuristics via stochastic domination. Theoret. Comput. Sci. **773**, 115–137 (2019)
17. Doerr, B., Gao, W., Neumann, F.: Runtime analysis of evolutionary diversity maximization for OneMinMax. In: Proceedings of the 18th ACM Conference on Genetic and Evolutionary Computation (GECCO 2016), Denver, CO, pp. 557–564 (2016)
18. Doerr, B., Kodric, B., Voigt, M.: Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC 2013), pp. 432–439 (2013)
19. Doerr, B., Qu, Z.: A first runtime analysis of the NSGA-II on a multimodal problem. IEEE Trans. Evol. Comput. **27**, 1288–1297 (2023)
20. Doerr, B., Qu, Z.: From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023), Washington, DC, pp. 12408–12416 (2023)
21. Doerr, B., Qu, Z.: Runtime analysis for the NSGA-II: provable speed-ups from crossover. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023), Washington, DC, pp. 12399–12407 (2023)
22. Doerr, B., Zheng, W.: Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021), Virtual, pp. 12293–12301 (2021)
23. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin (2015). https://doi.org/10.1007/978-3-662-44874-8
24. Friedrich, T., Hebbinghaus, N., Neumann, F.: Plateaus can be harder in multi-objective optimization. Theoret. Comput. Sci. **411**(6), 854–864 (2010)
25. Friedrich, T., Horoba, C., Neumann, F.: Illustration of fairness in evolutionary multi-objective optimization. Theoret. Comput. Sci. **412**(17), 1546–1556 (2011)
26. Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC 2003), vol. 3, pp. 1918–1925 (2003)
27. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimization. In: Proceedings of the 8th ACM Conference on Genetic and Evolutionary Computation (GECCO 2006), Seattle, WA, pp. 651–658 (2006)
28. Horoba, C.: Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem. In: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms (FOGA 2009), Orlando, FL, pp. 113–120 (2009)

29. Huang, Z., Zhou, Y., Luo, C., Lin, Q.: A runtime analysis of typical decomposition approaches in MOEA/D framework for many-objective optimization problems. In: Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021), Virtual, pp. 1682–1688 (2021)
30. 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)
31. Liang, J., et al.: An evolutionary multiobjective method based on dominance and decomposition for feature selection in classification. SCIENCE CHINA Inf. Sci. **67**(2), 120101 (2024)
32. Lu, T., Bian, C., Qian, C.: Towards running time analysis of interactive multi-objective evolutionary algorithms. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI 2024), Vancouver, Canada (2024, in press)
33. Neumann, F.: Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. Eur. J. Oper. Res. **181**(3), 1620–1629 (2007)
34. Neumann, F., Theile, M.: How crossover speeds up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem. In: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN 2010), Krakov, Poland, pp. 667–676 (2010)
35. Nguyen, A.Q., Sutton, A.M., Neumann, F.: Population size matters: rigorous runtime results for maximizing the hypervolume indicator. Theoret. Comput. Sci. **561**, 24–36 (2015)
36. Opris., A., Dang., D.C., Sudholt, D.: Runtime analyses of NSGA-III on many-objective problems. CORR abs/2404.11433 (2024)
37. Qian, C., Tang, K., Zhou, Z.H.: Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization. In: Proceedings of the 14th International Conference on Parallel Problem Solving from Nature (PPSN 2016), Edinburgh, Scotland, pp. 835–846 (2016)
38. Qian, C., Yu, Y., Zhou, Z.: An analysis on recombination in multi-objective evolutionary optimization. In: Proceedings of the 13th ACM Conference on Genetic and Evolutionary Computation (GECCO 2011), Dublin, Ireland, pp. 2051–2058 (2011)
39. Ren, S., Qiu, Z., Bian, C., Li, M.Q., Qian, C.: Maintaining diversity provably helps in evolutionary multimodal optimization. In: Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI 2024), Jeju Island, South Korea (2024, to appear)
40. Wietheger, S., Doerr, B.: A mathematical runtime analysis of the non-dominated sorting genetic algorithm III (NSGA-III). In: Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023), Macao, SAR, China, pp. 5657–5665 (2023)
41. Wietheger, S., Doerr, B.: Near-tight runtime guarantees for many-objective evolutionary algorithms. CORR abs/2404.12746 (2024)
42. Witt, C.: Fitness levels with tail bounds for the analysis of randomized search heuristics. Inf. Process. Lett. **114**(1–2), 38–41 (2014)
43. Yang, P., Zhang, L., Liu, H., Li, G.: Reducing idleness in financial cloud via multi-objective evolutionary reinforcement learning based load balancer. Sci. China Inf. Sci. **67**(2), 120102–(2024)
44. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007)
45. Zheng, W., Doerr, B.: Better approximation guarantees for the NSGA-II by using the current crowding distance. In: Proceedings of the 24th ACM Conference on Genetic and Evolutionary Computation (GECCO 2022), Boston, MA, pp. 611–619 (2022)
46. Zheng, W., Doerr, B.: Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). Artif. Intell. **325**, 104016 (2023)
47. Zheng, W., Doerr, B.: Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for many objectives. IEEE Trans. Evol. Comput. (2023, in press)

48. Zheng, W., Doerr, B.: Runtime analysis of the SMS-EMOA for many-objective optimization. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI 2024), Vancouver, Canada, pp. 20874–20882 (2024)
49. Zhou, Z.H., Yu, Y., Qian, C.: Evolutionary Learning: Advances in Theories and Algorithms. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-5956-9
50. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength pareto evolutionary algorithm. TIK report **103** (2001)

(Evolutionary) Machine Learning and Neuroevolution

Population-Based Algorithms Built on Weighted Automata

Gijs Schröder, Inge Wortel, and Johannes Textor(✉)

Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands
{gijs.schroeder,inge.wortel,johannes.textor}@ru.nl

Abstract. Many algorithms in natural computing and computational biology are population-based: genetic algorithms evolve candidate solutions for optimization problems; artificial immune systems and learning classifier systems maintain populations of rules. Using such algorithms at very large population sizes (e.g., millions or billions) is computationally expensive. Here, we develop a methodology for implementing population-based models using weighted finite state machines (WFSMs) with exact rational weights. For populations that can be represented as weighted sets of strings, WFSMs can reduce memory use and runtime of population-based algorithms by orders of magnitude. We demonstrate the generality of our approach by constructing an immune-inspired anomaly detector for string data and an evolutionary algorithm that solves Boolean satisfiability problems. The WFSM approach allows repurposing of advanced algorithms developed for natural language processing, and should be applicable to other population-based algorithms such as learning classifier systems.

Keywords: Simulation tools · Regular languages · Weighted automata

1 Introduction

Many algorithms in computer science perform a search, optimization, or learning procedure by maintaining and evolving a set of points in some kind of search space. Such *population-based* algorithms [4] encompass nature-inspired metaheuristics like evolutionary algorithms (EAs) [18], ant colony optimization [3], particle swarm optimization [14], and artificial immune systems [33], but are not only used in natural computing: various more classical algorithms such as the approximate Bayesian computation method for parameter inference [28] or simulated annealing can also be viewed as population-based.

This paper focuses on populations that can be represented as sets of strings with associated weights, such as, for instance, EAs with real-valued fitness functions. We ask a simple question: how can we efficiently implement such algorithms when the population sizes are very large—say, in the millions? While small populations are fine in some cases (even a simple 1+1-EA successfully solves many problems), large populations can be required to tackle complex, high-dimensional problems in many areas, such as anomaly detection by immune-inspired classifier systems [29].

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 315–332, 2024.
https://doi.org/10.1007/978-3-031-70071-2_20

An obvious approach to deal with large populations is parallelization, which is typically straightforward and achieves linear runtime improvement. Here we use a different approach that has been much less explored: *population compression*. While being more difficult to implement, it can, in some cases, achieve superlinear or even exponential gains [20]. To our knowledge, the idea of population compression first appeared in the field of artificial immune systems (AIS) [11], where it was used to build the first polynomial-time immune-inspired classification algorithms, a feat that had previously been considered unachievable as it was thought to be an NP-hard problem [32]. In these algorithms, population compression is achieved by representing the "detectors" comprising the population—which are simply strings—as a finite state machine (FSM). For large string sets with regular structure, FSMs can be exponentially smaller than an explicit list of the population. Recently, the FSM approach was used to build a computational model of T cell populations in the human immune system that contained tens of millions of "detectors" and processed the entire human proteome as input [35].

It is not straightforward to use FSM-based population compression methods in other population-based algorithms as existing methods are still relatively limited. Crucially, FSMs represent populations as sets without multiplicity. This means that all individuals in the population are qualitatively equal, limiting the types of operations that can be performed on them. For instance, in an EA, we may wish to sample new candidates for reproduction in a fitness-dependent manner; this kind of operation is not supported by existing FSM-based approaches.

In this paper, we will extend population compression methods developed for AIS in a manner that makes them useable for more general population-based algorithms. Specifically, we offer the following contributions:

1. We present a population compression approach based on *weighted* FSMs (WFSMs) with rational edge weights (Sect. 3). This approach allows us to augment the compressed population with important information such as multiplicity or fitness.
2. We use WFSMs to implement a weighted version of positive selection, a key AIS classification algorithm, and show that the addition of weights can lead to higher accuracy and robustness compared to the unweighted baseline (Sect. 4.2).
3. We use WFSMs to implement an EA that solves the Boolean satisfiability problem (Sect. 5.2).

We have implemented our WFSM-based algorithms in C++, making heavy use of the library OpenFST [1]. An implementation is available for download at Zenodo [27].

2 Background

We consider strings x over some finite alphabet Σ. E.g., $x = 0010 \in \{0,1\}^4$ is a string consisting of 4 binary characters and $x_3 = 1$ is its third character.

The symbol ℓ is always the length of some string. We write $|S|$ to denote the cardinality of a set S.

A *population* is a set of strings—e.g., the genotypes in an EA. A *weighted population* consists of a population S and a mapping $w : S \mapsto \mathbb{Q}$ that associates a rational number to each string in S (see Sect. 3 for why we choose rationals). For clearer distinction, a population S is also called an *unweighted population*. Unweighted populations are interchangeable with weighted populations with unit weights.

The weights can have different interpretations: positive integer weights could be used to simply store multiplicity of individuals in the population, whereas fractional weights could represent each individual's importance or fitness. Weights in the interval $(0, 1]$ could represent probabilities of being present in the set.

In the following, let (S, w_S), (T, w_T), and (U, w_U) each be weighted populations. For simplicity, let $w_A(x) = 0$ if $x \notin A$ and let $f = g \star h$ denote $f(x) = g(x) \star h(x)$, where $\star$ is any operation. In a slight abuse of notation, we define three binary operations for two populations (S, w_S) (abbreviated as S) and (T, w_T) (abbreviated as T):

$S \uplus T = U$ is the *sum* of S and T, where $U = S \cup T$ and $w_U = w_S + w_T$.

$S \cap T = U$ is the *product* of S and T, where $U = S \cap T$ and $w_U = w_S \times w_T$.

$S \setminus T = U$ is the *set difference* of S and T, where $U = S \setminus T$ and $w_U = w_S$.

A *weighted finite state machine* (WFSM) over an alphabet Σ and a semiring $\mathbb{K}$ is a 6-tuple $M = (Q, E, \sigma, k, q_0, F)$, where (Q, E) is a directed graph of states as vertices, with *initial* state $q_0 \in Q$ and *final* states $F \subseteq Q$. Edges $q_i \to q_j$ are labeled by characters $\sigma : E \mapsto \Sigma$ and *weights* $k : E \mapsto \mathbb{K}$. An example WFSM is shown in Fig. 1B.

This M defines a language $\mathcal{L}(M) \subseteq \Sigma^*$ and an associated function $w_M : \Sigma^* \mapsto \mathbb{K}$ by its graph structure and edge labels. For $q_\ell \in F$, an *accepting path* $\pi = q_0 \to \ldots \to q_\ell$ is *labeled* $x = \sigma(q_0 \to q_1) \ldots \sigma(q_{\ell-1} \to q_\ell)$ and has *weight* $k(\pi) = \prod_{j=0}^{\ell-1} k(q_j \to q_{j+1})$, where $\prod$ uses the product operator in $\mathbb{K}$. The weight $w_M(x)$ of a string x is the sum of all weights of accepting paths labeled with x, or zero in absence of such paths.

A WFSM is *deterministic* if no two edges from the same source state are labeled with the same character. A deterministic WFSM is called *minimal* if it has the least number of states out of all deterministic WFSMs accepting the same language and weights. To *minimize* a deterministic WFSM is to find a corresponding minimal WFSM.

Ordinary FSMs without weight can be recovered from WFSMs by substituting for $\mathbb{K}$ the two-element Boolean algebra. If we let $\mathbb{K} = \mathbb{Q}$ instead, then any WFSM M defines a set of strings $\mathcal{L}(M)$ and an associated w_M that together constitute a weighted population. The three operations defined above can then be implemented with weighted variants of union, intersection, and difference constructions. Restricting ourselves to deterministic WFSMs, we can compress these WFSMs by minimization.

2.1 Compressing Unweighted Populations

We now briefly review the existing approach from the AIS field where finite state machines (FSMs) are used as compressed representations of populations. This review delineates what this approach can already accomplish and where it currently falls short.

The AIS algorithms we will consider are classifiers that are based on populations of so-called *detectors*, where each detector is a string that represents a small part of a set $\mathcal{U}$ (universe) of objects to classify. By extension, detector populations can therefore represent ("cover") subsets of $\mathcal{U}$. Although this framework is general and allows $\mathcal{U}$ to be any set, in this paper we focus on strings over some alphabet Σ, where each string consists of ℓ characters ($\mathcal{U} = \Sigma^\ell$).

In addition to the *universe*: $\mathcal{U}$ that represents the objects to be classified, we define a set of *detectors* $\mathcal{D}$; possibly $\mathcal{U} = \mathcal{D}$. A *matching function* $m : \mathcal{D} \mapsto 2^\mathcal{U}$, where $2^\mathcal{U}$ denotes the powerset of $\mathcal{U}$, associates every detector with the elements it recognizes. We define the inverse matching function $m^{-1} : \mathcal{U} \mapsto 2^\mathcal{D}$ as $m^{-1}(x) = \{d \in \mathcal{D} \mid x \in m(d)\}$. In this setting, a population is simply a subset of $\mathcal{D}$.

Definition 1 (Matching Rules). *Given an alphabet Σ and a string length ℓ, we define the following matching rules:*

1. r-Contiguous matching: $\mathcal{U} = \mathcal{D} = \Sigma^\ell$, $m_r(x) = \{y \in \mathcal{U} : \exists i \in \{1, \ldots, \ell - r + 1\} : \forall j \in \{i, \ldots, i+r-1\} : x_j = y_j\}$*; in words, x and y have at least r identical consecutive characters. The parameter r, called* matching radius*, controls the number of strings each detector matches: increasing r means matching fewer strings. This pattern matching rule is common in AIS.*
2. r-Hamming matching: $m_r(x) = \{y \in \mathcal{U} : |\{i : x_i \neq y_i\}| \leq r\}$*; in words, x and y have at most r non-identical characters. Here, increasing the matching radius r means matching more strings.*

Using any such matching rule, we can build classification algorithms based on detector populations. Inspired by the way that T cells are generated and selected in real immune systems, there are two main population-based classification algorithms that were studied in the AIS field. Both are so-called *one-class* classification algorithms that take an input sequence $S \in \mathcal{U}^*$ (also called *self*) to construct a detector population P, which is then used to determine whether the elements of a second input sequence $T \in \mathcal{U}^*$ belong to the same class as the elements of S:

Definition 2 (Positive and Negative Selection). *For an input sequence $S = (s_1, \ldots, s_n)$, $S \in \mathcal{U}^n$, a detector type $\mathcal{D}$ and a matching function m, we call a detector population $P \subseteq \mathcal{D}$*

1. positively selected *if $P \subseteq \bigcup_i m^{-1}(s_i)$ is a set of detectors that match at least one input string.*
2. negatively selected *if $P \subseteq \mathcal{D} \setminus \bigcup_i m^{-1}(s_i)$ is a set of detectors that do not match any input string.*

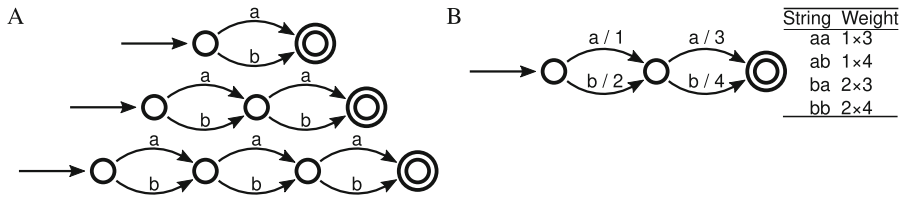


Fig. 1. FSMs as compact representations of sets of strings. (A) FSMs encoding the string set $\{a, b\}^{\ell}$ for $\ell \in \{1, 2, 3\}$. Although the string sets grow exponentially in size as a function of ℓ, FSM size is linear in ℓ. (B) A weighted FSM, along with a table showing the strings it contains and the associated weights for these strings. Path transitions are labeled (character/weight). Each unique path represents a string (the concatenation of every such character along the path) with its associated weight (the product of every weight along the path).

A negatively (positively) selected population P is maximal *if there is no strict superset of P that is also negatively (positively) selected with respect to the same input S. Given a population P and an element $t \in \mathcal{U}$, we define the* scoring function $P(y) = |P \cap m^{-1}(y)|$ *as the number of detectors in P that match y.*

For a positively selected population, the scoring function can be understood as a *normalcy score* (a high value $P(y)$ means that y is "similar to" S) whereas for negative selection, the interpretation is the opposite (*anomaly score*). The scores output by a positively or negatively selected population can be used for threshold-based classification. Note that we did not define which specific detectors are used, only which detectors *could* be in the population. A simple method is generating detectors by rejection sampling. However, depending on the input, rejection rates can be high [7,8], while at the same time huge populations can be required to achieve acceptable classification results [29].

These issues motivated the development of population compression techniques [11,12]. The basic idea is that FSMs can compactly represent large sets of strings with shared structure (Fig. 1A). This representation has several advantages. For any given population P, the smallest FSM that represents P can be computed in polynomial time in the size of P. The operations that are required to implement positively and negatively selected populations, such as union, intersection, set difference, and computation of cardinality, can all be efficiently performed directly on the FSM representation. Since FSMs are generic data structures, this approach benefits from a large body of work done to develop efficient FSM algorithms and high-quality, mature software implementations such as the OpenFST framework [1]. The FSM-based approach can improve runtime by orders of magnitude [30,31]. The key requirement is that we need to be able to efficiently generate an FSM that represents the population $m^{-1}(s_i)$ (Definition 2); whether this is possible depends on the matching rule that is used.

If we are able to employ the FSM approach, we no longer need to generate populations by random sampling and can instead just directly use maximal detector populations [30,31]. Interestingly, although positive and negative selec-

tion using randomly generated detectors can give different results, for many matching rules this is not the case when maximal detector sets are used.

Remark 1. For the matching rules in Definition 1, positive and negative selection with maximal detector sets are equivalent classifiers.

Proof. If P^+ and P^- are the maximal positively and negatively selected detector populations w.r.t. an input S, then we have $P^+ \cup P^- = \mathcal{D}$ and therefore $|P^+ \cap m^{-1}(y)| + |P^- \cap m^{-1}(y)| = |m^{-1}(y)|$. Since $|m^{-1}(y)|$ is the same value for all $y \in \mathcal{U}$ for the matching rules in Definition 1, the scoring function of negative selection is a constant minus the scoring function of positive selection, and vice versa.

Beyond positive and negative selection, further operations that could be performed with FSM-compressed classifier populations include sampling from P or incrementally modifying P [31]. However, some important limitations remain. Crucially for classification tasks, negative and positive selection do not take into account possible multiplicity of strings in the input sequences. In many real-world cases, it is important to distinguish, say, an input element that occurs once in the training data from another that occurs thousands of times. Moreover, AIS classifiers process the entire input data only once. Many other population-based algorithms modify the populations in more complex ways that require, for instance, weighted sampling from the population or pruning of low-weight elements from the population. Such operations are not supported by FSM-compressed populations.

3 Weighted Population Compression

We extend the FSM-based population compression approach explained in the previous section to *weighted* populations as defined in Sect. 2. We aim to compress the weighted populations with WFSMs and to implement the three weighted population operators using WFSM algorithms.

These WFSMs are preferably as small as possible. However, using conventional floats for edge weights instead of rationals causes WFSMs to explode in size with the number of operations performed on them. This is no minor issue; for the algorithms described in the next section, this size explosion prevented us from doing any testing on real data. We will demonstrate the issue and its solution using a simple task.

In this simple task, a WFSM M is constructed to accept the language $\{0, 1, 2\}^6$ with unit weight for each string. Rather than constructing M directly, single-string WFSMs are added successively in a binary tree recursion, minimizing intermediate and final WFSMs. This is similar to the procedure used for weighted positive selection in Sect. 4.

Thanks to the minimization of the final result, even this indirect construction should yield the minimal WFSM consisting of 7 states and 18 edges. In Fig. 2B,

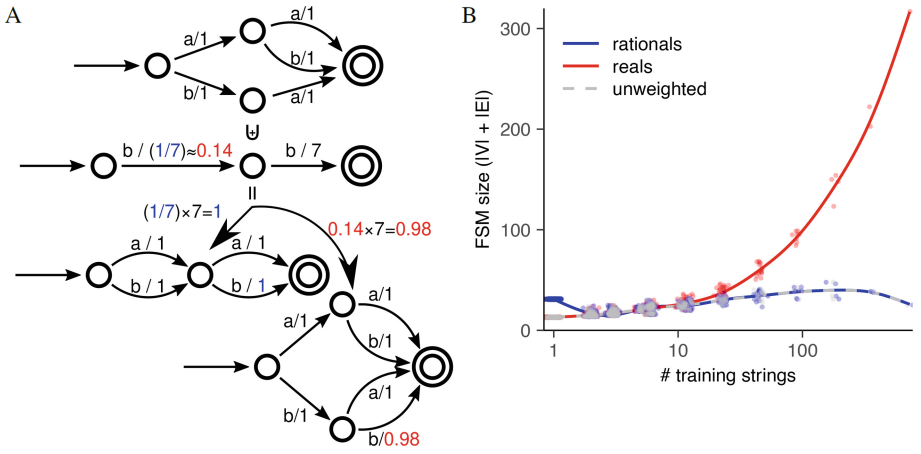


Fig. 2. Exact arithmetic helps keeping WFSMs small. (A) Weighted union between two WFSMs, using exact and inexact arithmetic. With exact arithmetic the path weight for bb is $\frac{1}{7} \times 7 = 1$, which can be merged with ab's path weight of 1 to obtain a small WFSM (bottom left, blue). However, with inexact arithmetic this weight becomes $0.14 \times 7 = 0.98$, which cannot be merged like before, leading to a larger WFSM (bottom right, red). (B) A (W)FSM accepting $\{0, 1, 2\}^6$ is constructed by successive union of single-string (W)FSMs in a binary tree recursion. With unweighted FSMs, compression is achieved by minimization. With WFSMs with float weights, inaccuracies accumulate and prevent states from being merged (red line). Exact rational artihmetic rescues the desired behavior (blue line). Points show FSM sizes for different numbers of training strings (with some jitter added in the x-direction to avoid overplotting). Lines show a Loess smoothing of the data. (Color figure online)

we show how WFSM size grows with the number of strings contained in intermediate stages of computing M. An unweighted FSM implementation indeed contains 7 states and 18 edges, whereas a WFSM implemented with float weights produces a "minimal" M with 80 states and 237 edges. The issue stems from a difference between minimizing unweighted FSMs and minimizing WFSMs. In unweighted FSMs, two states can be merged into a single state if all of their edges have the same character label and destination. In WFSMs—after redistributing weights [25]—edges need also have the same weight label. This requirement is hard to meet with inexact float arithmetic. Figure 2A (red line) demonstrates how inexact arithmetic can lead to a increase in WFSM size.

Typical solutions for this problem with floats are quantization or using approximate equality operators, both of which are implemented in OpenFST. However, these strategies are ultimately unsuccessful at preventing error accumulation when many repeated WFSM operations are performed, as will often be necessary in population-based algorithms. Therefore, we instead implemented weights using an exact rational representation from the Boost library [15], which restored the equivalence with the unweighted FSM in the final output, with intermediate stages being at most four times larger (Fig. 2B, blue line).

4 A WFSM-Based Artificial Immune System

We now apply the WFSM approach in two different ways. First, in this section, we use it to construct a weighted version of an AIS classification algorithm, and investigate its performance compared to the unweighted baseline. Second, in the next section, we will use it to implement an EA.

4.1 Positive Selection Using WFSMs

Given the equivalence established by Remark 1, we focus our experiments on positive selection. Consider the following weighted versions of the standard positive selection algorithm defined above:

For an input sequence $S = (s_1, \ldots, s_n) \in \mathcal{U}^n$, a detector type $\mathcal{D}$ and a matching function m, *weighted positive selection* uses the detector population $P = \biguplus_i m^{-1}(s_i)$ with weights $w(d) = |\{i : d \in m^{-1}(s_i)\}|$ where each detector is weighted by the number of input samples it recognizes. The *scoring function* $P_w(t)$ assigns a normalcy score $\sum_{d \in P \cap m^{-1}(t)} w(d)$ to every $t \in \mathcal{U}$.

Algorithm 1: Weighted positive selection

input : Samples $S = (s_1, \ldots, s_k) \in \mathcal{U}^k, T = (t_1, \ldots, t_l) \in \mathcal{U}^l$
output: Scores $D_w(t)$ for each $t_i \in T$
$M \leftarrow \biguplus_{i=1}^{n} M[s_i]$
foreach $i \in \{1, \ldots, l\}$ **do**
  output $|M \cap M[t_i]|$
end

We can implement weighted positive selection in a similar manner as done previously for the unweighted version (Definition 2). An important prerequisite is that for each input string $s \in \mathcal{U}$, we are able to generate a WFSM $M[s]$ containing all detectors that recognize s with unit weights, i.e., $w_{M[s]}(d) = \mathbf{1}(d \in m^{-1}(s))$, $\mathbf{1}$ being an indicator function. Earlier work shows how to construct such FSMs for common matching rules [31]. Then we can implement weighted positive selection as shown in Algorithm 1.

4.2 Probabilistic Classification

We now investigate whether and how this extension improves classifier performance.

Standard unweighted negative and positive selection algorithms were based on the assumption that the universe $\mathcal{U}$ of elements to be classified is disjointly partitioned into "self" and "nonself" subsets. The goal was to estimate the boundary between these two classes [13,32], which in theory can be solved perfectly without taking the multiplicity of input strings into account: a single witness string is enough to decide membership.

Unfortunately, many real-world problems do not fit this assumption. For example, suppose we were to distinguish language based on n-grams (short sequences of letters). It is known that even short n-grams such as 3-grams contain enough information to solve this task satisfactorily [9]. However, given enough input text, almost every combination of 3 input letters likely occurs at least once in the input regardless of the language (it has been pointed out that *llj* is not a typical letter sequence in English but "only a *killjoy* would claim" it never occurs [9]). This should make it critical to not only consider the presence or absence of a string, but also its frequency. Interestingly, previous research has shown that negative selection algorithms can nevertheless solve such problems reasonably well [35]—but the amount of input text used in that study was relatively small.

One way to model the aforementioned type of classification problems is by considering a "fuzzy membership function" $f : x \mapsto [0, 1]$ that assigns a *degree of membership* of each string to every class. Despite existing results on the performance of negative selection on classification problems, we hypothesized that unweighted AIS should perform poorly on such fuzzy problems.

To test our hypothesis, we first defined a very simple toy example of a fuzzy classification problem. In this *noisy bitstring problem*, we consider random bitstrings $X(c, \mu)$ where $c \in \{0, 1\}^\ell$ and $0 \leq \mu \leq 1$. $X(c, \mu)$ is generated by the following algorithm: draw a random number x from a geometric distribution with parameter $1 - \mu$. Let $x' = \min(x, \ell)$. Flip x' randomly chosen bits of c and return the result. In particular, $X(c, 0)$ is always c, and $X(c, 1)$ is always the bitwise complement of c.

One fuzzy classification problem is defined by membership functions f_0 and f_1:

$$f_0(x) = \text{Prob}(X(0^\ell, \mu) = x); f_1(x) = \text{Prob}(X(1^\ell, \mu) = x)$$

Particularly, for $0 < \mu < 1$, every bitstring has a nonzero probability of occurrence in both $X(c, 0)$ and $X(c, 1)$, but for $\mu \ll 1$, we have for example that $f_0(0^\ell) \gg f_1(0^\ell)$. Our population D is now tasked with assigning a score $D(t)$ to every string such that the distributions $D(X(0^\ell, \mu))$ and $D(X(1^\ell, \mu))$ are as different as possible—we will use the area under the receiver operating characteristic curve (AUC) to measure this difference.

When simulating unweighted versions of positive selection, we found the seemingly paradoxical effect that performance was reasonable for small input samples, but then rapidly degraded for larger input samples (Fig. 3A, black line). This occurred because with larger samples it became more and more likely to find the center string of the "foreign" class in the input. By contrast, weighted positive selection should not be "fooled" by such rare events because it can also learn from the frequencies of patterns in the input strings. Indeed, while small samples have little multiplicity and the two versions initially behaved very similarly, the performance of weighted selection kept improving with larger samples as expected (Fig. 3A, red line). Thus, we conclude that fuzzy classification problems can be difficult to solve using unweighted AIS classifiers, especially at large sample sizes. Weighted positive selection performed better and was also more robust to higher noise rates μ (Fig. 3B), which—similar to larger input sizes—

endanger performance by increasing the frequency of foreign-looking strings in the training input.

A well-known problem with AIS classifiers is their sometimes extreme sensitivity to the threshold parameter t that determines which strings are "similar" [7,8,29,35]. We also found this effect for unweighted positive selection on noisy bitstrings, where the "best" t additionally depended on input size (Fig. 3C). Interestingly, we found that weighted positive selection was more robust to the choice of t, with $t = 2, 3, 4$ now giving very similar performances throughout a range of input sizes (Fig. 3D).

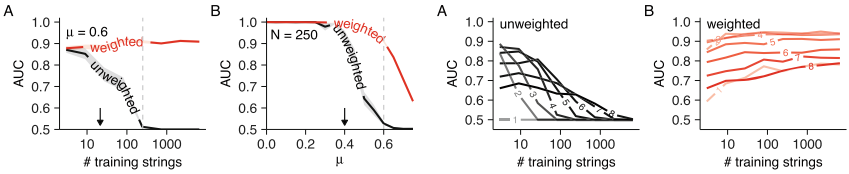


Fig. 3. Weighted and unweighted positive selection on the noisy bitstring problem. Throughout, AUC (mean ± SEM) of 20 independent runs is shown for $\ell = 8$; default values are $\mu = 0.6, t = 5, N = 250$, and the test set contains 100 random samples per class. Unweighted positive selection performs worse with increasing input size (A) and at high noise (B). It is also more sensitive to the choice of the parameter t (C, D). Dashed lines in (A,B) highlight the same parameter combination. Arrows represent the values of μ and N for which the expected number of strings with a majority of foreign bits in a training sample exceeds 1. (Color figure online)

These results suggest that on fuzzy classification problems, our weighted WFSM-AISs outperform their unweighted counterparts and are less sensitive to parameters like input size and detection threshold. The question remains: was this an extreme example, or does the same apply to real-world datasets?

4.3 Language Anomaly Detection

We revisited the problem of language anomaly detection as considered previously [35]. In that study, detector populations selected on English strings could detect test strings from "anomalous" languages among English strings reasonably well. However, the training sets used were relatively small (< 1000 English strings, using contiguous matching with $t = 3$)—small enough that foreign-looking 3-grams are unlikely to appear in the training data. We therefore asked: would the performance of such an unweighted AIS degrade as "unlikely" letter patterns *do* start to appear among English training strings?

To test this hypothesis, we downloaded the published set of strings from [35], as well as ∼800,000 English strings from the King James bible for training. From these data, we extracted 3-letter strings, and used our WFSM-AIS to perform both weighted and unweighted positive selection trained on randomly sampled

inputs of up to 50,000 English strings. When detecting Latin among English strings, we found that weighted positive selection outperforms its unweighted counterpart at large input sizes (Fig. 4A,B).

The input size where unweighted selection starts to perform badly depended on the threshold t; unlikely 2-grams might appear even in relatively small training sets of $\sim$100 strings, whereas the most unlikely 3-grams are rare enough that they do not appear in training sets of up to $\sim$1000 strings. Nevertheless, even these rare patterns eventually caused the performances of the weighted and unweighted AISs to diverge as inputs reach a size of several thousands of strings (Fig. 4A,B). Similar results were observed when substituting different languages for the anomalous strings (Fig. 4C). The size of the effect depended on the general similarity between English and the "anomalous" language considered—in line with the intuition that adding weights should not help learn a difference that is not there, such as when comparing English to more English.

All in all, our results demonstrate that when anomalous inputs have a non-zero probability to appear among training data, unweighted AISs perform poorly as the training set size increases. By contrast, weighted AISs are able to circumvent this issue by learning the information contained in the multiplicity in the training data. These findings suggest that weights will be crucial when applying AISs to real-world datasets.

5 A WFSM-Based Evolutionary Algorithm

We now turn our attention to applying the WFSM approach to a different kind of algorithm. We design a simple EA that we apply to solve Boolean satisfiability problems.

The Boolean satisfiability problem is to decide whether a given Boolean formula has a truth assignment over its variables such that the formula is satisfied. For our purposes, we define these problems as a set of n *clauses*, where each clause is a set of *literals*, with literals being integers in $1, \ldots, \ell$ or $-1, \ldots, -\ell$. A *truth assignment* x is a string in $\{0,1\}^\ell$. Such an x *satisfies* a clause c when there is some literal l in c, such that $x_{|l|}$ is 1 if l is positive and 0 if negative. Otherwise, the clause is *falsified*. If there is an x that satisfies all of a problem's clauses, then the problem is *satisfiable*.

Most Boolean satisfiability problems are NP-hard, but some randomized search heuristics, such as the WalkSAT algorithm [26], perform decently. In the next section, we use WFSMs to devise a novel approach to tackle such problems.

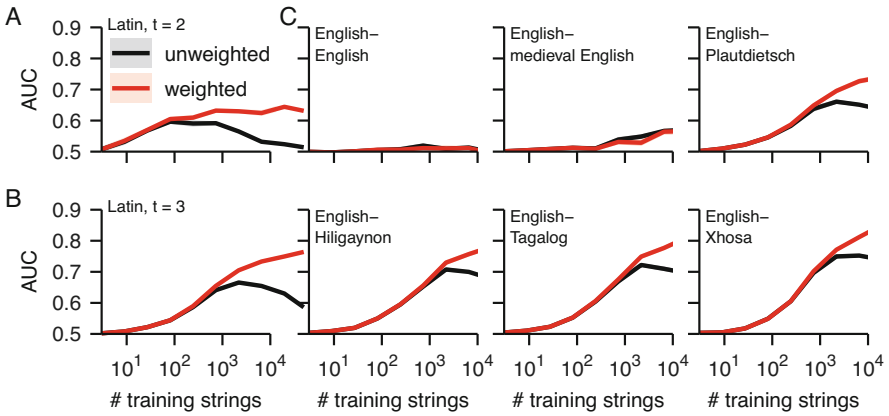


Fig. 4. Language anomaly detection performance of a positive selection algorithm drops for large training sets and is rescued by adding weights. 3-grams were extracted from Latin or English strings, and (un)weighted positive selection performed with t-contiguous matching against English strings for $t = 2$ (A) or $t = 3$ (B). Plots show the AUC (mean ± SEM) of 20 independent runs for 100 test strings from each language. English-English (C, top left) is a negative control experiment where both "normal" and "anomalous" strings are taken from the same language.

5.1 A WFSM-Based Algorithm for Boolean Satisfiability Problems

The central idea behind our WFSM-based approach is best explained when compared to our baseline, the well-known 1+1-EA that uses a population $P = \{x\}$ of size 1. In each step, each letter x is changed to a random other letter with mutation probability p_m, and x is replaced by x' if $w(x') > w(x)$. Though simple, the 1+1-EA can be very effective, and more sophisticated EAs do not necessarily outperform it [5]. However, a potential issue for the 1+1-EA on satisfiability problems is the *glassiness* of the solution space that some problems can exhibit [22]. Glassy problems have a rugged fitness landscape full of local minima, which the 1+1-EA could take a long time to escape from.

Our WFSM-based approach is based on the following idea: instead of moving through the fitness landscape along individual trajectories, we can explore entire subregions at once by constructing *Hamming balls* centered at a given candidate solution and jointly evaluating the fitness of all strings in this Hamming ball.

This joint evaluation proceeds as follows. For each clause c_i for $i \in \{1, \ldots, n\}$ we preconstruct a WFSM L_i that accepts local solutions: all strings that satisfy the corresponding clause with unit weights. Then $\biguplus_{i=1}^{n}(L_i \cap B)$, with B being our Hamming ball, results in a WFSM that contains all strings in B solving at least one clause, with a weight equal to the number of clauses solved. Each L_i is pre-constructed by taking the complement of the easily constructed WFSM that accepts strings that falsify c_i.

Algorithm 2 outlines an FSM-based EA that capitalizes on this idea. Similar to an EA, the algorithm proceeds in rounds where in each round a candidate for

Algorithm 2: WFSM-based evolutionary algorithm for Boolean satisfiability

input : Clauses $c_1, \ldots, c_n$, Hamming radius r, pruning fraction p.
output: A string satisfying all clauses.
for $i \in 1, \ldots, n$ **do**
  $L_i \leftarrow \Sigma^\ell \setminus \{x : x \text{ falsifies } c_i\}$
end
$s \leftarrow$ sampled string from one of the L_i
$P \leftarrow \biguplus_{i=1}^{n} (L_i \cap \{s\})$
while *heaviest string in* P *weighs less than* n **do**
  $s \leftarrow$ sampled string from P
  $B \leftarrow$ Hamming ball around s with radius r, unit weights
  $E \leftarrow \biguplus_{i=1}^{n} (L_i \cap B)$
  $P \leftarrow (P \setminus E) \uplus E$
  $P \leftarrow \text{prune}(P, p)$
end
output heaviest string in P

"reproduction" is selected (sampled) proportionally to its fitness. The Hamming ball around the chosen string is expanded (reproduction), the fitness of all offspring is evaluated, and the offspring is inserted into the population. At the end of each round, a pruning is performed that removes all strings whose weights are lower than p times the maximum fitness. Note that, unlike in most EAs, the population size of this algorithm is not fixed.

5.2 The WFSM-Based Algorithm Escapes Local Optima

We investigated a collection of difficult satisfiability problems from SATLIB called "Uniform Random-3-SAT" [17]. This collection consists of randomly generated problems with a fixed number of clauses and variables that are unusually difficult to solve [6, 23]. We expected such problems to have a glassy solution landscape [19], and hypothesized that an EA operating in such a landscape might benefit from Algorithm 2's ability to support large population sizes.

We selected four satisfiable problems with 50 variables and 218 clauses at random from the "Uniform Random-3-SAT" collection to run our EAs on. We performed six runs of both EAs on each problem, cutting them short if no solution is found in ten minutes. Algorithm 2 and the 1+1-EA both find strings that solve at least 210 of the clauses within seconds (Fig. 5A). In all runs, our 1+1-EA got stuck in a local optimum after these first seconds. Algorithm 2 finds new optima comparatively quickly. When Algorithm 2 finds a new optimum, its population size falls (Fig. 5B), because the pruning step removes strings that solve too few clause with respect to the newly found best string. During a stall, the population grows, building up an increasingly large repository of high fitness strings from which to expand further.

A single "clutch" of offspring consists of 20876 strings, whose fitness is evaluated concurrently in $n + 1$ automata operations. However, this large amount

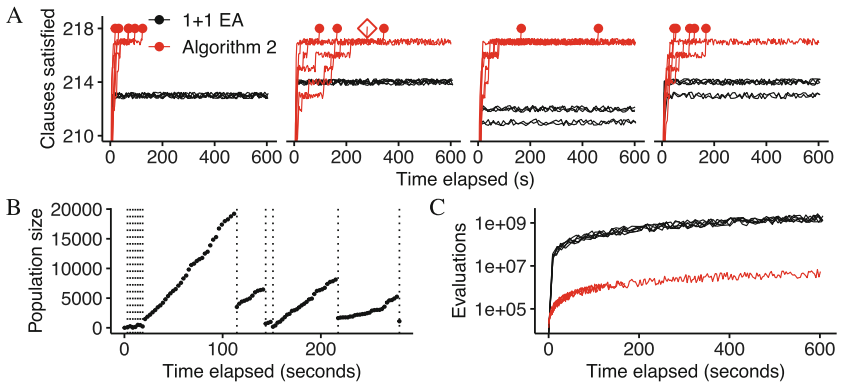


Fig. 5. Algorithm 2's population expands out of local optima. (A) Algorithm 2 ($r = 3$, $p = 0.99$) and a 1+1-EA ($p_m = 0.02$) running on four Boolean satisfiability problems. Jitter is added in the y direction and the y-axis is interrupted at 210 clauses. The diamond in the top right indicates a run for which in (B) population size is shown. Vertical dotted lines indicate when more clauses are satisfied. (C) Cumulative number of fitness evaluations for runs in the rightmost subplot of (A), counting two evaluations of the same string twice. Jitter is added in the y direction.

of simultaneous fitness evaluations alone does not explain why Algorithm 2 find the optimum faster; in fact, the 1+1-EA performs two orders of magnitude more fitness evaluations per time unit (Fig. 5C). Taken together, a picture emerges of how Algorithm 2 escapes local optima. Each round, a clutch of 20876 strings is inserted into a population whose size never exceeds 20000. Almost all offspring does not survive their first iteration. Stringent selection notwithstanding, the population keeps expanding into new territory, keeping above a "waterline" set by pruning. When encountering higher territory, this "waterline" is raised accordingly.

6 Discussion and Future Work

In this paper, we have developed and tested an approach to implement population compression for population-based algorithms that use weighted strings. Our approach is an extension of earlier work in the AIS field [20] that supports weighted instead of unweighted populations. We have demonstrated that (1) the addition of weights can considerably improve the performance and robustness of AIS algorithms; (2) the addition of weights makes it possible to leverage the population compression approach for implementing an EA. We are not aware of other EAs based on (W)FSMs, although there are other approaches to represent populations in compressed or implicit form [2,16,21]. In contrast to these approaches, FSMs allow for mass operations performed on all strings simultaneously.

It is important to mention some limitations of our approach. Compared to more standard implementations of a population-based algorithm, using WFSMs

requires significantly more advanced algorithms and data structures. To some extent, this is mitigated by the availability of WFSM libraries, but some issues remain. First, building WFSM-based population algorithms may require merging very large numbers of small classifiers. The OpenFST framework we currently use implements WFSM union in a way that does not yield a minimal result, such that we need to call minimization after every union to keep FSM size manageable. Second, handling weights in WFSMs proved to be more challenging than we anticipated. Our use of exact rational algebra proved critical to get our WFSM-AISs to work even on small input samples—without it, even populations trained on as few as 1000 input characters could rapidly blow up to 100 s of megabytes in size. While the use of rationals greatly improved this and allowed us to perform the experiments reported in this paper, it is not a complete solution because the rationals themselves could eventually "blow up" and use numerators or denominators that are too large to be represented as integers. While this did not cause any major issues in the experiments reported in this paper, we expect it to become problematic when storing large numbers of strings with very different weights at very different orders of magnitude.

On the other hand, these new issues that we encountered while using WFSMs in a new application domain might lead to interesting research questions: how can we develop fast and robust algorithms for use with very large numbers of small FSMs and related operations on them? For unweighted FSMs with a "levelled" structure that population models typically use (i.e., acyclic FSMs where all paths between two nodes have the same length), Textor *et al.* [31] implemented a custom FSM union algorithm that directly outputs a minimal FSM. This could be extended to WFSMs in future work. Likewise, we could use a directly determinizing union algorithm like the one by Mohri [24], created for a similar use-case. Minimization may also be sped up by the algorithm of Eisner [10]. Since WFSMs are generic data structures that are used in many different fields, such research may be useful outside of the context of natural computing. We look forward to working on such issues with researchers in the automata community.

We expect our approach to be applicable to many population-based algorithms. The most obvious candidate is perhaps the learning classifier system (LCS) [34]. Surprisingly, the close similarity of AIS and LCS has not been explored much in the literature—even though the original motivation behind these algorithms is very different, a LCS can in fact be viewed as a more sophisticated version of an AIS where positive or negative selection is only an initial step to build the classifier population, which is then evolved further by reinforcement learning. We hypothesize that the WFSM framework developed in this paper should be equally useful to increase the scale of LCS such that richer, more interesting problems can be studied. We hope that the insights gleaned from such work will allow us to better understand parallel distributed information processing systems in Nature, including but not limited to the immune system.

Acknowledgments. JT and GS were supported by NWO grant VI.Vidi.192.084 (to JT).
Disclosure of Interests. The authors do not declare any conflict of interest.

References

1. Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., Mohri, M.: OpenFst: a general and efficient weighted finite-state transducer library. In: Holub, J., Zdarek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 11–23. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76336-9_3
2. Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm. In: Machine Learning Proceedings 1995, pp. 38–46. Elsevier (1995)
3. Blum, C.: Ant colony optimization: introduction and recent trends. Phys. Life Rev. **2**(4), 353–373 (2005). https://doi.org/10.1016/j.plrev.2005.10.001
4. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. **35**(3), 268–308 (2003). https://doi.org/10.1145/937503.937505
5. Borisovsky, P., Eremeev, A.: Comparing evolutionary algorithms to the (1+1)-ea. Theoret. Comput. Sci. **403**(1), 33–41 (2008). https://doi.org/10.1016/j.tcs.2008.03.008
6. Cheeseman, P.C., Kanefsky, B., Taylor, W.M., et al.: Where the really hard problems are. In: IJCAI, vol. 91, pp. 331–337 (1991)
7. D'haeseleer, P.: An immunological approach to change detection: theoretical results. In: Proceedings 9th IEEE Computer Security Foundations Workshop. IEEE Computer Society Press (1996). https://doi.org/10.1109/csfw.1996.503687
8. D'haeseleer, P., Forrest, S., Helman, P.: An immunological approach to change detection: algorithms, analysis and implications. In: Proceedings 1996 IEEE Symposium on Security and Privacy. IEEE Computer Society Press (1996). https://doi.org/10.1109/secpri.1996.502674
9. Dunning, T.: Statistical Identification of Language. Tech. Rep. MCCS 94-273, New Mexico State University (1994)
10. Eisner, J.: Simpler and more general minimization for weighted finite-state automata. In: Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, pp. 64–71 (2003). https://doi.org/10.3115/1073445.1073454
11. Elberfeld, M., Textor, J.: Efficient algorithms for string-based negative selection. In: Andrews, P.S., et al. (eds.) ICARIS 2009. LNCS, vol. 5666, pp. 109–121. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03246-2_14
12. Elberfeld, M., Textor, J.: Negative selection algorithms on strings with efficient training and linear-time classification. Theor. Comput. Sci. **412**, 534–542 (2011). https://doi.org/10.1016/j.tcs.2010.09.022
13. Forrest, S., Perelson, A., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In: Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy. IEEE Computer Society Press (1994). https://doi.org/10.1109/risp.1994.296580
14. Gad, A.G.: Particle swarm optimization algorithm and its applications: a systematic review. Archiv. Comput. Methods Eng. **29**(5), 2531–2561 (2022). https://doi.org/10.1007/s11831-021-09694-4
15. Boost. Boost C++ libraries. https://www.boost.org

16. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Trans. Evol. Comput. **3**(4), 287–297 (1999). https://doi.org/10.1109/4235.797971
17. Hoos, H.H., Stützle, T.: Satlib: an online resource for research on sat. Sat **2000**, 283–292 (2000)
18. Jansen, T.: Analyzing Evolutionary Algorithms: The Computer Science Perspective. Springer, Incorporated (2013). https://doi.org/10.1007/978-3-642-17339-4
19. Kirkpatrick, S., Selman, B.: Critical behavior in the satisfiability of random Boolean expressions. Science **264**(5163), 1297–1301 (1994). https://doi.org/10.1126/science.264.5163.1297
20. Liśkiewicz, M., Textor, J.: Negative selection algorithms without generating detectors. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2010), pp. 1047–1054. ACM (2010). https://doi.org/10.1145/1830483.1830673
21. Manso, A., Correia, L.: Genetic algorithms using populations based on multisets. New Trends Artif. Intell. EPIA **2009**, 53–64 (2009)
22. Martin, O.C., Monasson, R., Zecchina, R.: Statistical mechanics methods and phase transitions in optimization problems. Theoret. Comput. Sci. **265**(1–2), 3–67 (2001). https://doi.org/10.1016/s0304-3975(01)00149-9
23. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distributions of sat problems. In: Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992), pp. 459–465. AAAI Press (1992)
24. Mohri, M.: On some applications of finite-state automata theory to natural language processing. Nat. Lang. Eng. **2**(1), 61–80 (1996). https://doi.org/10.1017/S135132499600126X
25. Mohri, M.: Minimization algorithms for sequential transducers. Theoret. Comput. Sci. **234**(1–2), 177–201 (2000). https://doi.org/10.1016/S0304-3975(98)00115-7
26. Schoning, T.: A probabilistic algorithm for k-sat and constraint satisfaction problems. In: 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039). SFCS-99. IEEE Computer Society (1999). https://doi.org/10.1109/sffcs.1999.814612
27. Schröder, G., Textor, J.: Population-based algorithms built on weighted automata (implementation). Zenodo (2024). https://doi.org/10.5281/zenodo.12205008
28. Sisson, S.A., Fan, Y., Tanaka, M.M.: Sequential Monte Carlo without likelihoods. Proc. Natl. Acad. Sci. **104**(6), 1760–1765 (Feb 2007). https://doi.org/10.1073/pnas.0607208104
29. Stibor, T., Mohr, P., Timmis, J., Eckert, C.: Is negative selection appropriate for anomaly detection? In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. ACM (2005). https://doi.org/10.1145/1068009.1068061
30. Textor, J.: A comparative study of negative selection based anomaly detection in sequence data. In: Coello Coello, C.A., Greensmith, J., Krasnogor, N., Liò, P., Nicosia, G., Pavone, M. (eds.) ICARIS 2012. LNCS, vol. 7597, pp. 28–41. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33757-4_3
31. Textor, J., Dannenberg, K., Liśkiewicz, M.: A generic finite automata based approach to implementing lymphocyte repertoire models. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 129–136 (2014). https://doi.org/10.1145/2576768.2598331
32. Timmis, J., Hone, A., Stibor, T., Clark, E.: Theoretical advances in artificial immune systems. Theor. Comput. Sci. **403**(1), 11–32 (2008). https://doi.org/10.1016/j.tcs.2008.02.011

33. Timmis, J., Knight, T., de Castro, L.N., Hart, E.: An Overview of Artificial Immune Systems, pp. 51–91. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-662-06369-9_4
34. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: a complete introduction, review, and roadmap. J. Artif. Evol. Appl. **2009**, 1–25 (2009). https://doi.org/10.1155/2009/736398
35. Wortel, I.M., Keşmir, C., de Boer, R.J., Mandl, J.N., Textor, J.: Is T cell negative selection a learning algorithm? Cells **9**(3), 690 (2020). https://doi.org/10.3390/cells9030690

Automatic Brain Tumor Segmentation Using Convolutional Neural Networks: U-Net Framework with PSO-Tuned Hyperparameters

Shoffan Saifullah[1,2] and Rafał Dreżewski[1(✉)]

[1] Faculty of Computer Science, AGH University of Krakow, 30-059 Krakow, Poland
{saifulla,drezew}@agh.edu.pl, shoffans@upnyk.ac.id
[2] Department of Informatics, Universitas Pembangunan Nasional Veteran Yogyakarta, Yogyakarta 55281, Indonesia

Abstract. Accurate segmentation of brain tumors from magnetic resonance imaging (MRI) data is imperative for precise diagnosis and treatment planning. Manual segmentation, while accurate, is labor-intensive and subject to human error. In this study, we propose an innovative approach leveraging a modified convolutional neural network (CNN) architecture, U-Net, optimized using Particle Swarm Optimization (PSO) to tackle this challenge. Our method achieves significantly improved segmentation accuracy through pre-training hyperparameter tuning, particularly adjusting learning rates and dropout rates with PSO. Compared to existing methods, we observe enhancements of up to 4 p.p. in the Dice Similarity Coefficient (DSC) and 2 p.p. in the Jaccard Index (JI). Using skip connections and dropout layers in CNN-U-Net enables the effective capture of intricate features while mitigating overfitting, resulting in robust segmentation performance. Experimental results showcase the superiority of our approach across different tumor classes, including Meningioma, Glioma, and Pituitary, as well as overall, with maximum DSC and JI values of 94.14% and 89.02%, respectively. Comparative analysis against established techniques underscores the reliability and robustness of our proposed method. By demonstrating the efficacy of deep learning coupled with metaheuristic optimization in medical image segmentation, our study contributes to advancing the field's understanding and applications. This research lays a foundation for future automated brain tumor segmentation developments, with implications for clinical practice and patient care.

Keywords: Brain tumor segmentation · Convolutional neural networks · Particle swarm optimization · Medical image analysis · Deep learning

1 Introduction

Brain tumors represent a significant health concern globally, with profound implications for patients' quality of life and survival rates [7,10]. These tumors

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 333–351, 2024.
https://doi.org/10.1007/978-3-031-70071-2_21

can manifest in various forms, ranging from benign to malignant, and their accurate diagnosis and treatment planning are paramount in clinical practice [22]. However, brain tumor segmentation from medical imaging data poses several challenges [50], including tumor heterogeneity, irregular shapes, and variable contrasts [32], making manual segmentation by radiologists laborious and prone to subjective interpretations [2].

The importance of brain tumor segmentation lies in its critical role in guiding treatment decisions, such as surgical planning, radiation therapy, and monitoring treatment response [14,15,23]. Accurate delineation of tumor boundaries is essential for optimizing treatment efficacy while minimizing damage to healthy brain tissue [38,44]. Although manual segmentation [46,50] is considered the gold standard, it is also time-consuming, subject to inter-observer variability, and may not scale well to large datasets [3,28].

Recently, advancements in deep learning, particularly Convolutional Neural Networks (CNNs), have revolutionized medical image analysis [1,48], offering automated solutions for segmentation tasks [6]. Among these architectures, the U-Net has emerged as a popular choice due to its ability to capture both local and global context while preserving spatial information [12,47,59]. The U-Net architecture comprises an encoder-decoder structure with skip connections, enabling precise segmentation by combining low-level and high-level features [5,13].

Previous research has demonstrated the efficacy of CNNs and U-Net in addressing various challenges associated with medical image segmentation, including brain tumor segmentation [4,17,49]. These methods leverage the representational power of deep learning models to extract meaningful features from raw image data, enabling accurate delineation of tumor regions [29,51]. However, achieving optimal performance with these models often requires careful selection and tuning of hyperparameters [24,38,52].

Hyperparameters such as learning rate and dropout [39] rate play crucial roles in determining the performance [55] and generalization ability of U-Net models. Learning rate controls the step size during gradient descent optimization, influencing the convergence speed and stability of the training process. Dropout regularization helps prevent overfitting by randomly deactivating neurons during training, promoting model robustness [27].

While previous studies have explored various optimization techniques for U-Net hyperparameters, such as grid search and random search, these methods are computationally intensive and may not guarantee optimal results [55]. In this study, we propose a novel approach to U-Net optimization using Particle Swarm Optimization (PSO) to fine-tune hyperparameters, focusing specifically on learning rate and drop-out. PSO is a metaheuristic optimization algorithm inspired by social behavior in nature, wherein particles adjust their position in the search space based on their experience and the experience of their neighbors.

By leveraging PSO to optimize U-Net hyperparameters, we aim to enhance the segmentation performance of the model while reducing the computational burden associated with traditional optimization methods. Our approach builds upon the strengths of deep learning and metaheuristic optimization, offering a

robust and efficient solution for automatic brain tumor segmentation from MRI images.

The rest of the paper is divided into three parts. Section 2 details the materials and methods, covering dataset, preprocessing, modified U-Net segmentation architecture, integration of CNN-U-Net framework, and PSO-based hyperparameter tuning. Section 3 presents experimental results evaluating the proposed PSO-tuned CNN-U-Net framework, analyzing strengths and limitations, and comparing with existing methods. Lastly, Sect. 4 presents conclusions, emphasizing the significance of the proposed methodology in automatic brain tumor segmentation, and suggests future research directions.

2 Materials and Methods

2.1 Dataset and Preprocessing

Our research used the dataset and preprocessing steps applied to enhance brain MRI images before segmentation. The dataset, curated by Cheng et al. [9] and available on Kaggle.com, comprises 3064 MRI images categorized into Meningioma, Glioma, and Pituitary classes (708, 1426, and 930 images, respectively). Each image has a resolution of 512 × 512 pixels and 24-bit depth, as illustrated in Fig. 1.

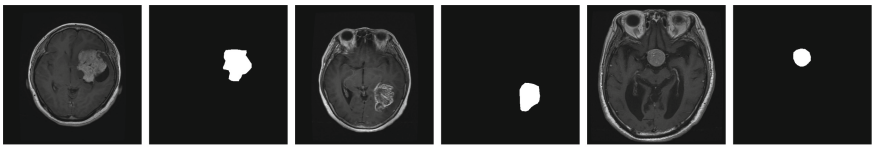


Fig. 1. Sample brain MRI image dataset with Ground Truth (Mask) of (a) Meningioma, (b) Glioma, and (c) Pituitary.

Our research initiates with preprocessing the MRI images to optimize their quality and prepare them for segmentation tasks [42,43]. Essential preprocessing techniques are employed to standardize image characteristics for efficient processing and analysis. Resizing—images are resized to 225 × 225 pixels to standardize dimensions for computational efficiency and to mitigate the overhead. This resizing ensures uniformity, optimizes memory usage, and enhances segmentation model efficiency. Conversion to 8-bit depth—images are converted to 8-bit depth to focus on intensity information and simplify the processing. This conversion optimizes memory utilization, streamlines segmentation algorithms, and ensures compatibility across datasets for efficient grayscale processing.

2.2 Modified CNN-U-Net Framework for Medical Image Segmentation

This section presents a detailed exposition of the modified CNN-U-Net framework tailored explicitly for medical image segmentation tasks. Our approach is underpinned by a fusion of convolutional neural networks (CNNs) and the U-Net architecture, augmented with several strategic modifications to enhance feature extraction and segmentation accuracy.

At the core of our architecture (Fig. 2) lies the utilization of feature maps P1, P2, P3, and P4 [61], acting as the encoder for input images. These feature maps are generated through a hierarchical down-sampling process characterized by down-sampling rates of 64, 128, 256, and 512, respectively. The feature maps, denoted as P*, encode increasingly abstract representations of the input images, facilitating the accurate delineation of foreground and background regions. This hierarchical feature extraction mechanism enables the model to capture intricate spatial and contextual information crucial for accurate segmentation.

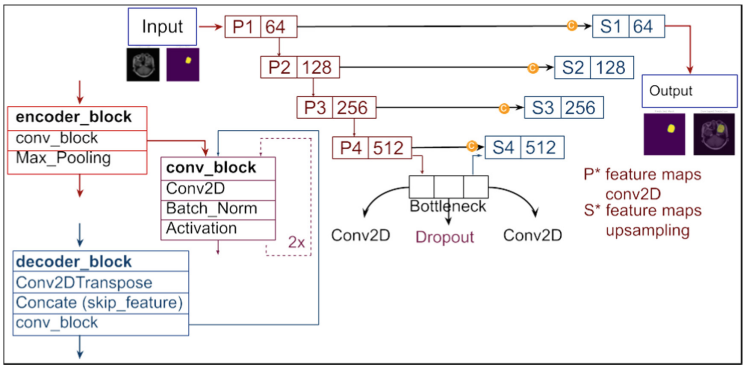


Fig. 2. Modified U-Net architecture using feature maps and dropouts.

Furthermore, our architecture incorporates feature maps S1, S2, S3, and S4 [56], representing feature maps from different scales in the decoding stages. Each feature map from P* and S* is seamlessly fused through skip connections, enabling the model to integrate high-level semantic information with low-level spatial details. This fusion of features from multiple scales enhances the model's ability to localize and segment fine-grained structures in medical images, thereby improving segmentation accuracy.

Integral to our modified framework, skip connections (Concate(skip_feature)) merge feature maps from corresponding encoder and decoder layers. This operation concatenates the output from a previous layer to a subsequent layer, preserving crucial spatial context and aiding in accurate segmentation by reconnecting lost details from deeper network layers. To combat overfitting—a common challenge in deep learning models—dropout layers are strategically placed within our

network. These layers randomly deactivate a subset of neurons during training, promoting a robust model that generalizes well to new, unseen data by ensuring no single neuron is overly critical to the network's output.

Each conv_block (Convolutional Block) within our architecture consists of several layers, including Conv2D layers, activation functions (typically ReLU), and often batch normalization layers. These blocks are crucial for processing input images through various filters to extract meaningful feature maps. Conv2D refers to the two-dimensional convolutional layers that perform our model's bulk computational processing. These layers apply multiple filters to the input to extract vital image features. The operation carried out by these layers involves the convolution of the layer's filters with the input, producing a tensor of outputs that form the backbone of our feature learning mechanism.

In addition to architectural modifications, we propose the integration of dropout regularization to enhance the robustness and generalization capabilities of the CNN-U-Net framework. Dropout is strategically applied at various network layers during training, effectively preventing overfitting by stochastically dropping units from the network. This regularization technique encourages the network to learn more robust and generalizable features, improving segmentation performance on unseen data.

To optimize the performance and accuracy of our CNN-U-Net framework, we adopt a metaheuristic approach for hyperparameter optimization. Metaheuristic algorithms, such as Particle Swarm Optimization (PSO), can fine-tune critical hyperparameters, including dropout and learning rates. This iterative optimization process enables us to identify the optimal configuration of hyperparameters that maximizes segmentation accuracy while minimizing the risk of overfitting.

2.3 Metaheuristics Algorithms—PSO Approach for Tuning Hyperparameters

In this section, we delve into the utilization of metaheuristic algorithms, particularly the Particle Swarm Optimization (PSO) approach, for finely tuning critical hyperparameters within the U-Net architecture [16,60]. Specifically, we focus on optimizing dropout rates and learning rates, which are pivotal elements influencing model robustness and convergence during medical image segmentation tasks.

Particle Swarm Optimization (PSO) is a potent optimization algorithm inspired by the collective behavior of organisms, such as birds flocking or fish schooling. Within PSO, a population of candidate solutions, termed particles, traverses the search space, with each particle dynamically adjusting its position based on its experience and the experiences of its neighbors [45]. Through iterative updates, particles collectively converge towards the optimal solution.

The PSO-based tuning of the U-Net hyperparameters include the following steps:

- Initialization: the process begins with the initialization of a swarm of particles, each representing a potential solution within the hyperparameter space. These particles are initialized with random positions and velocities within predefined ranges for dropout rates and learning rates.
- Evaluation: each particle evaluates its associated hyperparameter configuration by training the U-Net model with the given dropout rates and learning rates. The performance of each configuration is assessed using a predefined objective function, typically based on segmentation accuracy metrics like the Dice Similarity Coefficient (DSC) or Jaccard Index (JI).
- Update Personal and Global Bests: each particle maintains a record of its personal best position (the hyperparameter configuration that yielded the best performance) and the global best position (the best configuration discovered by any particle in the swarm). These positions are updated iteratively as better solutions are found.
- Update Velocity and Position: through iterative updates, each particle adjusts its velocity and position based on its experience and the experiences of its neighbors. This adjustment is guided by the personal best position and global best position, enabling particles to converge towards promising regions of the search space.
- Convergence Criterion: the PSO algorithm iterates until a convergence criterion is met, typically defined by a maximum number of iterations or a threshold for performance improvement. Once the convergence criterion is satisfied, the algorithm terminates, and the best hyperparameter configuration discovered by the swarm is returned.
- Final Model Training: the U-Net model is trained using the best hyperparameter configuration identified by the PSO algorithm. This involves retraining the model with optimal dropout rates and learning rates to obtain the final segmentation model.

The integration of PSO for tuning hyperparameters within the U-Net framework offers a robust and efficient optimization technique for achieving optimal segmentation accuracy. By leveraging swarm intelligence and collaborative search dynamics, PSO enables the identification of near-optimal hyperparameter configurations, thereby enhancing the performance and effectiveness of our segmentation model. Through extensive experimentation and analysis, we evaluate the efficacy of PSO in optimizing dropout rates and learning rates for our U-Net framework, providing valuable insights into the effectiveness of the PSO approach for hyperparameter optimization in medical image segmentation tasks.

2.4 Evaluation Measures

In this section, we explore the evaluation measures utilized to assess the performance of our proposed U-Net framework for brain tumor segmentation [34]. We employ a comprehensive set of metrics, including accuracy, loss, Dice Similarity Coefficient (DSC), and Jaccard Index (JI), to quantify the effectiveness of the model during both training and validation phases [45,47,57]. Additionally,

these metrics are leveraged to evaluate the segmentation results obtained by the model, providing insights into its segmentation accuracy and efficacy.

Accuracy and loss are fundamental metrics used to measure the performance of machine learning models, including those used for medical image segmentation. Accuracy represents the ratio of correctly predicted instances to the total number of instances in the dataset, providing a measure of the model's overall correctness. Loss, on the other hand, quantifies the discrepancy between the predicted outputs of the model and the ground truth labels. Standard loss functions include categorical cross-entropy and mean squared error, which are optimized during the training process to minimize prediction errors.

The Dice Similarity Coefficient (DSC) and Jaccard Index (JI) are widely used metrics for evaluating the quality of segmentation results in medical imaging tasks. These metrics provide measures of overlap or similarity between the predicted segmentation masks and the ground truth masks. The DSC, also known as the F1 score, computes the similarity between two sets by measuring the intersection over the average of their sizes. It is defined as $DSC = \frac{2\times|A\cup B|}{|A|+|B|}$, where A represents the predicted segmentation mask and B represents the ground truth mask. Similarly, the Jaccard Index (JI) computes the ratio of intersection to union of two sets and is defined as $JI = \frac{|A\cap B|}{|A\cup B|}$.

During the training phase, accuracy and loss are computed iteratively to monitor the performance of the model and guide the optimization process [54,58]. The model is trained to minimize the loss function while maximizing the accuracy of the training data. In the validation phase, accuracy and loss are calculated on a separate validation dataset to assess the generalization performance of the model. This allows us to evaluate how well the model performs on unseen data and identify potential overfitting or underfitting issues.

Once the model is trained and validated, segmentation results are evaluated using the DSC and JI metrics [36]. These metrics quantify the degree of correspondence between the predicted and ground truth masks, providing quantitative measures of segmentation accuracy and efficacy [21]. By employing a comprehensive set of evaluation measures, including accuracy, loss, DSC, and JI, we gain valuable insights into the performance of our U-Net framework for brain tumor segmentation. These metrics enable us to assess the model's effectiveness during training and validation and evaluate the quality of segmentation results, ultimately guiding the development and refinement of our segmentation approach.

3 Results and Discussions

This section presents the outcomes and discussions of our study on automatic brain tumor segmentation using the PSO-tuned CNN-U-Net framework. We analyze the results of hyperparameter optimization with PSO, evaluate the performance of our method on brain tumor segmentation tasks, present segmentation results, and compare our method with previous approaches. Through concise analysis and discussion, we highlight the effectiveness and significance of our proposed method.

3.1 PSO-Based CNN-U-Net Hyperparameters Optimization

In this section, we delve into the performance analysis of our proposed PSO-tuned CNN-U-Net framework, which aims to optimize hyperparameters such as the learning rate and dropout to enhance segmentation accuracy.

Experimental research began with systematic exploration of various learning rates and dropouts. The learning rate, responsible for determining the step size during optimization, was varied from 10^{-4} to 10^{-2}. In contrast, the dropout rate, serving as a regularization technique to prevent overfitting, was explored from 0.1 to 0.5. Through iterative experiments, we meticulously evaluated the impact of different hyperparameter configurations on model convergence and segmentation accuracy.

The optimization process is visualized in Fig. 3, which illustrates the convergence behavior and performance trends of the PSO algorithm across various combinations of learning rates and dropout rates. This visualization provided valuable insights into the optimization process, highlighting the regions of the hyperparameter space where the PSO algorithm converged most effectively, facilitating the identification of optimal hyperparameter configurations.

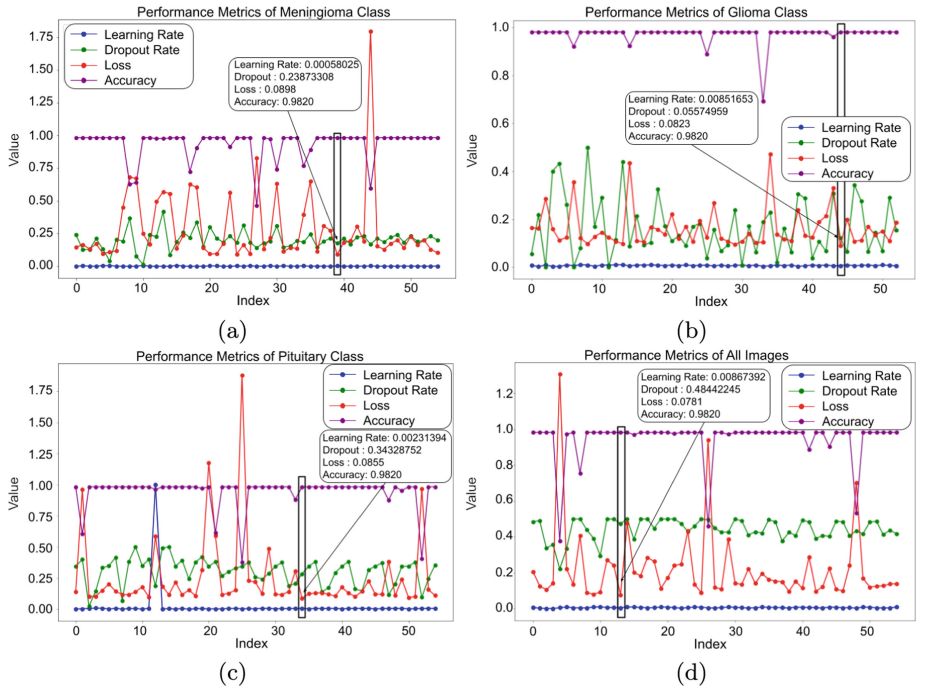


Fig. 3. Evaluation of PSO optimization for Meningioma (a), Glioma (b), Pituitary (c), and All Images (d) with tuned hyperparameters of Learning Rate and Dropout, depicting Accuracy and Loss metrics.

To cater to the unique characteristics of each class of brain tumors, we conducted class-specific optimization of hyperparameters. For instance, the Meningioma class exhibited optimal segmentation results with the learning rate of 0.00058025 and the dropout of 0.23873308. Similarly, the Glioma class demonstrated superior performance, with the learning rate of 0.00851653 and the dropout of 0.05574959. The Pituitary class achieved enhanced segmentation accuracy with the learning rate of 0.00231394 and the dropout of 0.34328752.

Furthermore, considering the dataset encompassing all classes, we identified the most effective hyperparameter configuration. The learning rate of 0.00867392 and the dropout of 0.48442245 emerged as the optimal combination, demonstrating remarkable convergence and segmentation accuracy.

These findings underscore the significance of personalized parameter tuning in maximizing segmentation performance across diverse classes of brain tumors. The meticulous optimization of hyperparameters using PSO facilitated the refinement of the CNN-U-Net framework, enhancing its ability to delineate brain tumor boundaries accurately. These insights provide valuable guidance for optimizing deep learning-based segmentation models for medical image analysis, with implications for enhancing diagnostic accuracy and clinical decision-making.

3.2 Performance of the Proposed Method

The performance evaluation of our proposed model on training and validation sets over 50 epochs provides valuable insights into its efficacy in brain tumor segmentation. Figure 4 illustrates the trends of key performance metrics, including accuracy, loss, Dice Similarity Coefficient (DSC), and Jaccard Index (JI), throughout the training process. The blue lines represent training metrics, while the red lines depict validation metrics.

The graphical representation in Fig. 4 showcases a notable improvement in performance metrics over successive iterations, indicating the effectiveness of our model. Specifically, the accuracy of our model exhibits robust performance, reaching up to 0.99 while maintaining a minimal loss of 0.008. The DSC and JI metrics also demonstrate considerable efficacy, with values surpassing 0.90 and 0.82, respectively. These results demonstrate the suitability of the proposed model for training and testing, emphasizing the efficacy of the applied metaheuristic PSO algorithm.

Furthermore, an in-depth analysis of the results obtained from the PSO-tuned U-Net hyperparameters reveals significant performance enhancements, as detailed in Table 1. Table 1 presents the performance metrics, including accuracy, loss, DSC, and JI, for different classes of brain tumors during both training and validation phases. For each class, the table comprises three rows representing distinct hyperparameter configurations. The first row corresponds to the hyperparameters optimized using PSO. The second row represents a configuration without dropout (DO), while the third row reflects a configuration with optimal dropout (DO).

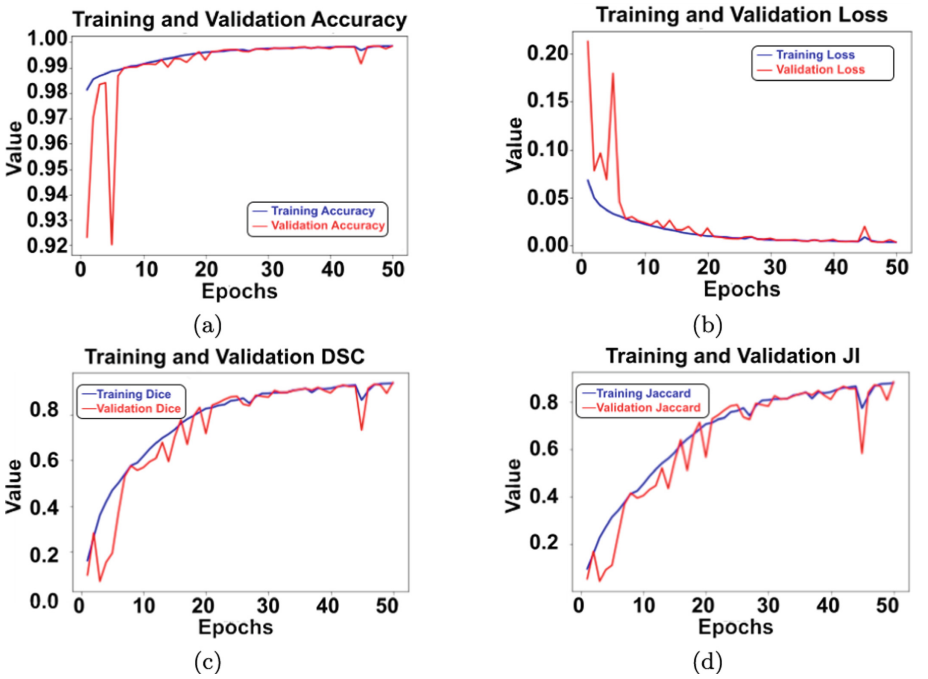


Fig. 4. Performance graph of our proposed model: (a) Accuracy, (b) Loss, (c) Dice Similarity Coefficient (DSC), and (d) Jaccard Index (JI), where blue-lines represent training and red-lines represent validation. (Color figure online)

Table 1. Performance results of model evaluation during training and testing, showcasing Accuracy, Loss, Dice Coefficient Similarity (DSC), and Jaccard Index (JI) based on PSO-tuned hyperparameters, compared with configurations without Dropout (DO) and with optimal DO settings, focusing on Learning Rate (LR) and Dropout (DO).

Class	*	Hyperparameters		Training				Validation			
		LR	DO	Acc	Loss	DSC	JI	Acc	Loss	DSC	JI
Meningioma	1	0.00058025	0.23873308	0.9987	0.0032	0.9432	0.8934	0.9986	0.0032	0.9414	0.8902
	2	0.001	0	0.9986	0.0034	0.9402	0.8880	0.9984	0.0039	0.9344	0.8779
	3	0.001	0.5	0.9985	0.0037	0.9372	0.8827	0.9985	0.0036	0.9352	0.8793
Glioma	1	0.00851653	0.05574959	0.9974	0.0061	0.9123	0.8398	0.9975	0.0058	0.9146	0.8436
	2	0.001	0	0.9970	0.0070	0.9007	0.8206	0.9973	0.0065	0.9036	0.8255
	3	0.001	0.5	0.9973	0.0063	0.9113	0.8381	0.9974	0.0060	0.9101	0.8360
Pituitary	1	0.00231394	0.34328752	0.9991	0.0021	0.9202	0.8541	0.9992	0.0020	0.9205	0.8546
	2	0.001	0	0.9991	0.0021	0.9009	0.8222	0.9990	0.0024	0.9127	0.8417
	3	0.001	0.5	0.9988	0.0027	0.8996	0.8199	0.9988	0.0027	0.9033	0.8258
All Images	1	0.00867392	0.48442245	0.9984	0.0037	0.9300	0.8704	0.9985	0.0036	0.9312	0.8722
	2	0.001	0	0.9983	0.0040	0.9263	0.8640	0.9983	0.0040	0.9253	0.8621
	3	0.001	0.5	0.9982	0.0044	0.9206	0.8543	0.9980	0.0048	0.9150	0.8449

*) 1: PSO Optimization; 2: configuration without DO; 3: configuration with optimal DO.

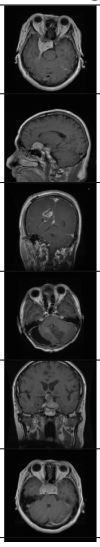
For the Meningioma class, the optimized hyperparameters, with the learning rate (LR) of 0.00058025 and the dropout rate (DO) of 0.23873308, yield impressive results, achieving the accuracy of 0.9987 and the DSC of 0.9432 on the training set. Similar trends are observed for the Glioma and Pituitary classes, with optimized hyperparameters leading to substantial improvements in segmentation accuracy and performance metrics.

Overall, the results demonstrate the efficacy of our proposed PSO-tuned CNN-U-Net framework in accurately delineating brain tumor boundaries. The meticulous optimization of hyperparameters using PSO enhances segmentation accuracy and facilitates practical applications in medical image analysis. These findings highlight the potential of our approach in improving diagnostic accuracy and aiding clinical decision-making in neuroimaging.

3.3 Segmentation Results

The segmentation results comprehensively evaluate the proposed model's performance in identifying and outlining brain tumor regions. Table 2 presents a detailed comparison between the predicted and ground truth masks and quantitative metrics, including the Dice Similarity Coefficient (DSC) and Jaccard Index (JI).

Table 2. Results of brain tumor segmentation illustrating the Predicted Mask and Overlap compared to the Original Image and Mask, according to evaluation metrics including Dice Similarity Coefficient (DSC) and Jaccard Index (JI).

MRI Image	Mask	Predicted	Overlap	Description
				Class: Meningioma DSC: 0.9204 JI: 0.8252
				Class: Meningioma DSC: 0.9753 JI: 0.9518
				Class: Glioma DSC: 0.9520 JI: 0.9085
				Class: Glioma DSC: 0.9618 JI: 0.9264
				Class: Pituitary DSC: 0.9495 JI: 0.9038
				Class: Pituitary DSC: 0.9445 JI: 0.8948

For the Meningioma class, the segmentation accuracy is notably high, with DSC values of 0.9204 and 0.9753 and corresponding JI values of 0.8252 and 0.9518. These metrics indicate a strong agreement between the predicted and ground truth masks, suggesting that the model effectively captures the intricate boundaries of Meningioma tumors. The high DSC and JI values indicate the model's ability to accurately segment Meningioma tumors, facilitating precise tumor localization and characterization.

Similarly, the segmentation results for the Glioma class demonstrate robust performance, with DSC values of 0.9520 and 0.9618 and JI values of 0.9085 and 0.9264. These findings underscore the model's efficacy in accurately delineating Glioma tumor regions. The high DSC and JI values reflect the model's capability to accurately capture the spatial extent of Glioma tumors, facilitating accurate tumor segmentation and localization.

The segmentation model achieves favorable results in the Pituitary class, yielding DSC values of 0.9495 and 0.9445 and JI values of 0.9038 and 0.8948. These metrics highlight the model's ability to accurately identify and segment Pituitary tumors, contributing to its overall efficacy in tumor localization. The high DSC and JI values underscore the model's ability to accurately delineate Pituitary tumor regions, facilitating precise tumor segmentation and characterization.

The observed high DSC and JI values across all tumor classes indicate the robustness and accuracy of the proposed model in brain tumor segmentation. The model's ability to accurately delineate tumor regions demonstrates its potential for clinical applications, including computer-aided diagnosis and treatment planning. Moreover, the segmentation results underscore the effectiveness of the proposed methodology, highlighting its utility in automated brain tumor segmentation for clinical decision-making.

3.4 Comparison of the Proposed Method with Other Approaches

In this section, we thoroughly compare our proposed method for brain tumor segmentation and several existing techniques. We evaluate these methods based on performance metrics such as the Dice Similarity Coefficient (DSC) and Jaccard Index (JI), which are pivotal in assessing the accuracy of segmentation results. Our experimental evaluation covers various classes of tumors, including Meningioma, Glioma, and Pituitary, comparing our method with well-established approaches such as ANN-AC [25], ANN-MACWE [53], U-Net Multimodal [40], U-Net AT (OUAT) [20], Edge U-Net [30], and CNN-based [11] methods. Across all tumor classes, our method consistently outperforms these existing approaches, exhibiting notable percentage point (p.p.) differences in DSC and JI scores.

For instance (Table 3), our proposed method achieves excellent results with DSC values above 91% for all three classes and JI values above 84%. Specifically, in Meningioma segmentation, our method achieves remarkable DSC of 94.14% and JI of 89.02%, surpassing the performance of ANN-AC [25] by 4.17 p.p. and 11.51 p.p., respectively. However, it slightly lags behind U-Net Multimodal by 0.45 p.p. for DSC in the Meningioma class. In the Glioma class, our method

achieves DSC of 91.46% and JI of 84.36%, which is superior to most compared methods but falls short by 0.3% points for DSC and 0.32% points for JI when compared to U-Net Multimodal.

Table 3. Comparison of image segmentation results (Dice Similarity Coefficient (DSC) and Jaccard Index (JI)) between the proposed method and other approaches for each class of brain tumors (Meningioma, Glioma, and Pituitary).

Method	Evaluation Measures					
	Meningioma		Glioma		Pituitary	
	DSC	JI	DSC	JI	DSC	JI
ANN-AC [25]	0.8997	0.7751	0.6554	0.4342	0.8395	0.7359
ANN-MACWE [53]	–	0.8598	–	0.7642	–	0.7684
U-Net Multimodal [40]	0.9459	–	0.7241	–	0.9108	–
U-Net AT (OUAT) [20]	0.8243	–	0.6077	–	0.7848	–
Edge U-Net [30]	0.888	0.7985	0.9176	0.8478	0.8728	0.77743
CNN-based [11]	0.894	–	0.779	–	0.813	–
Proposed Method	0.9414	0.8902	0.9146	0.8436	0.9205	0.8546

Moreover, our method is particularly effective in the Pituitary class, achieving DSC of 92.05% and JI of 85.46%, surpassing Edge U-Net [30] by 4.77 p.p. and 7.717 p.p., respectively. Such performance results in the case of the Pituitary class demonstrate the efficacy of our modified U-Net architecture in handling diverse tumor characteristics.

Moving on to overall image segmentation (Table 4), we evaluate our method against various existing techniques such as MST-based [33], CNN U-Net [37], MAG-Net [18], Optimized U-Net [19], WBSO [31], Pipeline U-Net [26], Ensemble U-Net-ResNet [41], Modified ResNet [8], and ResUnet-TL [35]. Our method stands out by consistently achieving superior Dice Similarity Coefficient (DSC) scores, recording a high of 93.12%, the best among all compared methods, indicating precise identification and delineation of tumor boundaries.

However, while our DSC score is the highest, our Jaccard Index (JI) score of 87.22% is lower by 5.82 p.p. compared to ResUnet-TL, which achieved JI of 93.04%. This gap underscores an area for potential refinement to improve our method's overall similarity and area overlap with the ground truth masks.

The success of our proposed method can be attributed to several factors, including advanced optimization techniques, architectural modifications, and the utilization of metaheuristic Particle Swarm Optimization (PSO) algorithm. By optimizing key hyperparameters such as learning rate and dropout, our method enhances the adaptability and robustness of the segmentation model, resulting in more accurate delineation of tumor regions. The significant improvements observed in DSC and JI metrics underscore the superiority of our method in accurately capturing tumor boundaries and spatial overlap with ground truth masks. The detailed evaluation and comparative analysis validate the effectiveness of our method, positioning it as a valuable tool for medical imaging applications and clinical diagnosis.

Table 4. Comparison of image segmentation results (Dice Similarity Coefficient (DSC) and Jaccard Index (JI)) between the proposed method and other approaches for all tumor classes.

Method	Evaluation Measures	
	DSC	JI
MST based [33]	0.8469	0.7742
CNN U-Net [37]	0.76	0.67
MAG-Net [18]	0.74	0.6
Optimized U-Net [19]	0.8232	0.82
WBSO [31]	–	0.6646
Pipeline U-Net [26]	–	0.8312
Ensemble U-Net-ResNet [41]	0.8731	0.7902
Modified ResNet [8]	0.83	0.84
ResUnet-TL [35]	0. 9237	0.9304
CNN-based [11]	0.828	–
Proposed Method	0.9312	0.8722

In conclusion, our proposed method enhances brain tumor segmentation by demonstrating improved accuracy and adaptability compared to existing approaches. The detailed evaluation and comparative analysis provided in this section illustrate the potential of our method to segment brain tumor regions from MRI images more effectively. While our results show notable improvements, they represent a progressive development in the field rather than a breakthrough, underscoring the need for further validation and research to thoroughly verify our findings' broader applicability.

4 Conclusion

This study introduced a novel approach to brain tumor segmentation utilizing a modified CNN-U-Net framework optimized with Particle Swarm Optimization (PSO) algorithm. We achieved improved segmentation accuracy by leveraging advanced deep-learning techniques and metaheuristic optimization. Our method effectively addressed the challenge of hyperparameter tuning by employing PSO to optimize parameters like learning rate and dropout, resulting in enhanced segmentation performance. Furthermore, integrating skip connections and dropout layers in the CNN-U-Net architecture facilitated the capture of intricate features while mitigating overfitting.

Through comprehensive experiments and evaluation, our proposed method has demonstrated consistent performance across different classes of tumors, including Meningioma, Glioma, and Pituitary, and for all images combined. Comparative analysis against existing techniques revealed significant improvements in the Dice Similarity Coefficient (DSC) and Jaccard Index (JI), highlighting the efficacy and adaptability of our approach. These findings highlight the potential of deep learning coupled with metaheuristic optimization algorithms in advancing medical image segmentation, offering promising applications in computer-aided diagnosis and treatment planning. Future research will focus on

validating and extending the applicability of our approach to larger datasets and various pathological conditions to further enhance its clinical utility.

Acknowledgement. This research was supported by the Polish Ministry of Science and Higher Education funds assigned to AGH University of Krakow and by PLGrid under grant no. PLG/2023/016757.

References

1. Abdou, M.A.: Literature review: efficient deep neural networks techniques for medical image analysis. Neural Comput. Appl. **34**(8), 5791–5812 (2022). https://doi.org/10.1007/s00521-022-06960-9
2. Abdusalomov, A.B., Mukhiddinov, M., Whangbo, T.K.: Brain tumor detection based on deep learning approaches and magnetic resonance imaging. Cancers **15**(16), 4172 (2023). https://doi.org/10.3390/cancers15164172
3. Agravat, R.R., Raval, M.S.: A survey and analysis on automated glioma brain tumor segmentation and overall patient survival prediction. Arch. Comput. Methods Eng. **28**(5), 4117–4152 (2021). https://doi.org/10.1007/s11831-021-09559-w
4. Akter, A., et al.: Robust clinical applicable CNN and U-Net based algorithm for MRI classification and segmentation for brain tumor. Expert Syst. Appl. **238**, 122347 (2024). https://doi.org/10.1016/j.eswa.2023.122347
5. Al-Murshidawy, M.A.A., Al-Shamma, O.: A review of deep learning models (U-Net architectures) for segmenting brain tumors. Bull. Electr. Eng. Inform. **13**(2), 1015–1030 (2024). https://doi.org/10.11591/eei.v13i2.6015
6. Balwant, M.: A review on convolutional neural networks for brain tumor segmentation: methods, datasets, libraries, and future directions. IRBM **43**(6), 521–537 (2022). https://doi.org/10.1016/j.irbm.2022.05.002
7. Besnard, J., et al.: Social cognition in adult survivors of brain tumors: studying the relationship between theory of mind and quality of life. Brain Inj. **38**(3), 160–169 (2024). https://doi.org/10.1080/02699052.2024.2309246
8. Chatterjee, P., Das Sharma, K., Chakrabarti, A.: Weakly supervised learning in domain transfer scenario for brain lesion segmentation in MRI. Multimedia Tools Appl. (2024). https://doi.org/10.1007/s11042-023-17888-0
9. Cheng, J., et al.: Enhanced performance of brain tumor classification via tumor region augmentation and partition. PLoS ONE **10**(10), e0140381 (2015). https://doi.org/10.1371/journal.pone.0140381
10. Choe, Y.H., Lee, S., Lim, Y., Kim, S.H.: Machine learning-derived model for predicting poor post-treatment quality of life in Korean cancer survivors. Support. Care Cancer **32**(3), 143 (2024). https://doi.org/10.1007/s00520-024-08347-z
11. Díaz-Pernas, F.J., Martínez-Zarzuela, M., Antón-Rodríguez, M., González-Ortega, D.: A deep learning approach for brain tumor classification and segmentation using a multiscale convolutional neural network. Healthcare **9**(2), 153 (2021). https://doi.org/10.3390/healthcare9020153
12. Esmaeilzadeh Asl, S., Chehel Amirani, M., Seyedarabi, H.: Brain tumors segmentation using a hybrid filtering with U-Net architecture in multimodal MRI volumes. Int. J. Inf. Technol. **16**(2), 1033–1042 (2024). https://doi.org/10.1007/s41870-023-01485-3
13. Feng, Y., Cao, Y., An, D., Liu, P., Liao, X., Yu, B.: DAUnet: a U-shaped network combining deep supervision and attention for brain tumor segmentation. Knowl.-Based Syst. **285**, 111348 (2024). https://doi.org/10.1016/j.knosys.2023.111348

14. Fernando, K.R.M., Tsokos, C.P.: Deep and statistical learning in biomedical imaging: state of the art in 3D MRI brain tumor segmentation. Inf. Fusion **92**, 450–465 (2023). https://doi.org/10.1016/j.inffus.2022.12.013
15. Galldiks, N., et al.: Contribution of PET imaging to radiotherapy planning and monitoring in glioma patients - a report of the PET/RANO group. Neuro Oncol. **23**(6), 881–893 (2021). https://doi.org/10.1093/neuonc/noab013
16. Ghosh, S., Singh, A., Kumar, S.: BBBC-U-Net: optimizing U-Net for automated plant phenotyping using big bang big crunch global optimization algorithm. Int. J. Inf. Technol. **15**(8), 4375–4387 (2023). https://doi.org/10.1007/s41870-023-01472-8
17. Guo, X., Lin, X., Yang, X., Yu, L., Cheng, K.T., Yan, Z.: UCTNet: uncertainty-guided CNN-Transformer hybrid networks for medical image segmentation. Pattern Recogn., 110491 (2024). https://doi.org/10.1016/j.patcog.2024.110491
18. Gupta, S., Punn, N.S., Sonbhadra, S.K., Agarwal, S.: MAG-Net: multi-task attention guided network for brain tumor segmentation and classification. In: Srirama, S.N., Lin, J.C.-W., Bhatnagar, R., Agarwal, S., Reddy, P.K. (eds.) BDA 2021. LNCS, vol. 13147, pp. 3–15. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-93620-4_1
19. Ingle, A., Roja, M., Sankhe, M., Patkar, D.: Efficient segmentation and classification of the tumor using improved encoder-decoder architecture in brain MRI images. Int. J. Electr. Comput. Eng. Syst. **13**(8), 643–651 (2022). https://doi.org/10.32985/ijeces.13.8.4
20. Isunuri, B.V., Kakarla, J.: Fast brain tumour segmentation using optimized U-Net and adaptive thresholding. Automatika **61**(3), 352–360 (2020). https://doi.org/10.1080/00051144.2020.1760590
21. Jyothi, P., Singh, A.R.: Deep learning models and traditional automated techniques for brain tumor segmentation in MRI: a review. Artif. Intell. Rev. **56**(4), 2923–2969 (2023). https://doi.org/10.1007/s10462-022-10245-x
22. Kaur, G., Chaudhary, N.: Brain tumor detection using machine learning hybrid approach. In: 2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), pp. 1076–1081. IEEE. https://doi.org/10.1109/UPCON59197.2023.10434384
23. Khalighi, S., Reddy, K., Midya, A., Pandav, K.B., Madabhushi, A., Abedalthagafi, M.: Artificial intelligence in neuro-oncology: advances and challenges in brain tumor diagnosis, prognosis, and precision treatment. npj Precis. Oncol. **8**(1), 80 (2024). https://doi.org/10.1038/s41698-024-00575-0
24. Kumar, G.M., Parthasarathy, E.: Development of an enhanced U-Net model for brain tumor segmentation with optimized architecture. Biomed. Signal Process. Control **81**, 104427 (2023). https://doi.org/10.1016/j.bspc.2022.104427
25. Kumar, S.B., Panda, R., Agrawal, S.: Brain magnetic resonance image tumor detection and segmentation using edgeless active contour. In: 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1–7. IEEE (2020). https://doi.org/10.1109/ICCCNT49239.2020.9225296
26. Kumar Bhatt, S., Srinivasan, D.S., Prakash, P.: Brain tumor segmentation pipeline model using U-Net based foundation model. Data Metadata **2**, 197 (2023). https://doi.org/10.56294/dm2023197
27. Lin, H., Zeng, W., Zhuang, Y., Ding, X., Huang, Y., Paisley, J.: Learning rate dropout. IEEE Trans. Neural Netw. Learn. Syst. **34**(11), 9029–9039 (2023). https://doi.org/10.1109/TNNLS.2022.3155181

28. Lorenzen, E.L., et al.: A national study on the inter-observer variability in the delineation of organs at risk in the brain. Acta Oncol. **60**(11), 1548–1554 (2021). https://doi.org/10.1080/0284186X.2021.1975813
29. Lotlikar, V.S., Satpute, N., Gupta, A.: Brain tumor detection using machine learning and deep learning: a review. Curr. Med. Imaging Formerly Curr. Med. Imaging Rev. **18**(6), 604–622 (2022). https://doi.org/10.2174/1573405617666210923144739
30. Gab Allah, A.M., Sarhan, A.M., Elshennawy, N.M.: Edge U-Net: brain tumor segmentation using MRI based on deep U-Net model with boundary information. Expert Syst. Appl. **213**, 118833 (2023). https://doi.org/10.1016/j.eswa.2022.118833
31. Mano, A., Anand, S.: Method of multi-region tumour segmentation in brain MRI images using grid-based segmentation and weighted bee swarm optimisation. IET Image Proc. **14**(12), 2901–2910 (2020). https://doi.org/10.1049/iet-ipr.2019.1234
32. May, J.L., Garcia-Mora, J., Edwards, M., Rossmeisl, J.H.: An illustrated scoping review of the magnetic resonance imaging characteristics of Canine and Feline brain tumors. Animals **14**(7), 1044 (2024). https://doi.org/10.3390/ani14071044
33. Mayala, S., Herdlevær, I., Haugsøen, J.B., Anandan, S., Gavasso, S., Brun, M.: Brain tumor segmentation based on minimum spanning tree. Front. Signal Process. **2** (2022). https://doi.org/10.3389/frsip.2022.816186
34. Micallef, N., Seychell, D., Bajada, C.J.: Exploring the U-Net++ model for automatic brain tumor segmentation. IEEE Access **9**, 125523–125539 (2021). https://doi.org/10.1109/ACCESS.2021.3111131
35. Murmu, A., Kumar, P.: A novel Gateaux derivatives with efficient DCNN-Resunet method for segmenting multi-class brain tumor. Med. Biol. Eng. Comput. **61**(8), 2115–2138 (2023). https://doi.org/10.1007/s11517-023-02824-z
36. Natarajan, A., Kumarasamy, S.: Efficient segmentation of brain tumor using FL-SNM with a metaheuristic approach to optimization. J. Med. Syst. **43**(2), 25 (2019). https://doi.org/10.1007/s10916-018-1135-y
37. Raghu, S., Lakshmi, T.A.: Brain Tumor detection based on MRI Image Segmentation Using U-Net. Ann. RSCB **26**(1), 579–594 (2022)
38. Ranjbarzadeh, R., Zarbakhsh, P., Caputo, A., Tirkolaee, E.B., Bendechache, M.: Brain tumor segmentation based on optimized convolutional neural network and improved chimp optimization algorithm. Comput. Biol. Med. **168**, 107723 (2024). https://doi.org/10.1016/j.compbiomed.2023.107723
39. Ravikiran, H.K., Jayanth, J., Sathisha, M.S., Bindu, K.: Optimizing sheep breed classification with bat algorithm-tuned CNN hyperparameters. SN Comput. Sci. **5**(2), 219 (2024). https://doi.org/10.1007/s42979-023-02544-z
40. Razzaghi, P., Abbasi, K., Shirazi, M., Rashidi, S.: Multimodal brain tumor detection using multimodal deep transfer learning. Appl. Soft Comput. **129**, 109631 (2022). https://doi.org/10.1016/j.asoc.2022.109631
41. Roshan, S., Tanha, J., Zarrin, M., Babaei, A.F., Nikkhah, H., Jafari, Z.: A deep ensemble medical image segmentation with novel sampling method and loss function. Comput. Biol. Med., 108305 (2024). https://doi.org/10.1016/j.compbiomed.2024.108305
42. Saifullah, S., Dreżewski, R.: Enhanced medical image segmentation using CNN based on histogram equalization. In: 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), pp. 121–126 (2023). https://doi.org/10.1109/ICAAIC56838.2023.10141065
43. Saifullah, S., Dreżewski, R.: Modified histogram equalization for improved CNN medical image segmentation. Procedia Comput. Sci. **225**(C), 3021–3030 (2023). https://doi.org/10.1016/j.procs.2023.10.295

44. Saifullah, S., Dreżewski, R.: Advanced medical image segmentation enhancement: a particle-swarm-optimization-based histogram equalization approach. Appl. Sci. **14**(2), 923 (2024). https://doi.org/10.3390/app14020923
45. Saifullah, S., Drezewski, R.: Improved brain tumor segmentation using modified u-net based on particle swarm optimization image enhancement. In: Genetic and Evolutionary Computation Conference (GECCO 2024 Companion), Melbourne, VIC, Australia. ACM, New York (2024). https://doi.org/10.1145/3638530.3654339
46. Saifullah, S., Dreżewski, R.: Redefining brain tumor segmentation: a cutting-edge convolutional neural networks-transfer learning approach. Int. J. Electr. Comput. Eng. (IJECE) **14**(3), 2583 (2024). https://doi.org/10.11591/ijece.v14i3.pp2583-2591
47. Saifullah, S., Pranolo, A., Dreżewski, R.: Comparative analysis of image enhancement techniques for braintumor segmentation: contrast, histogram, and hybrid approaches. E3S Web Conf. **501**, 1020 (2024). https://doi.org/10.1051/e3sconf/202450101020
48. Singh, A.K., Mishra, A.: Revolutionizing brain tumor diagnosis: harnessing convolutional neural networks for enhanced prediction and classification. In: 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT), pp. 245–250. IEEE (2024). https://doi.org/10.1109/IC2PCT60090.2024.10486347
49. Srinivasan, S., Durairaju, K., Deeba, K., Mathivanan, S.K., Karthikeyan, P., Shah, M.A.: Multimodal biomedical image segmentation using multi-dimensional U-convolutional neural network. BMC Med. Imaging **24**(1), 38 (2024). https://doi.org/10.1186/s12880-024-01197-5
50. Suhirman, S., Saifullah, S., Hidayat, A.T., Kusuma, M.A., Drezewski, R.: Real-time mask-wearing detection in video streams using deep convolutional neural networks for face recognition. Int. J. Electr. Comput. Eng. (IJECE) **14**(1), 1005 (2024). https://doi.org/10.11591/ijece.v14i1.pp1005-1014
51. Talukder, M.A., et al.: An efficient deep learning model to categorize brain tumor using reconstruction and fine-tuning. Expert Syst. Appl. **230**, 120534 (2023). https://doi.org/10.1016/j.eswa.2023.120534
52. Tandel, G.S., Tiwari, A., Kakde, O.: Performance enhancement of MRI-based brain tumor classification using suitable segmentation method and deep learning-based ensemble algorithm. Biomed. Signal Process. Control **78**, 104018 (2022). https://doi.org/10.1016/j.bspc.2022.104018
53. Thias, A.H., Al Mubarok, A.F., Handayani, A., Danudirdjo, D., Rajab, T.E.: Brain tumor semi-automatic segmentation on MRI T1-weighted images using active contour models. In: 2019 International Conference on Mechatronics, Robotics and Systems Engineering (MoRSE), pp. 217–221. IEEE (2019). https://doi.org/10.1109/MoRSE48060.2019.8998651
54. Wang, G., Li, W., Ourselin, S., Vercauteren, T.: Automatic brain tumor segmentation based on cascaded convolutional neural networks with uncertainty estimation. Front. Comput. Neurosci. **13** (2019). https://doi.org/10.3389/fncom.2019.00056
55. Wojciuk, M., Swiderska-Chadaj, Z., Siwek, K., Gertych, A.: Improving classification accuracy of fine-tuned CNN models: impact of hyperparameter optimization. Heliyon **10**(5), e26586 (2024). https://doi.org/10.1016/j.heliyon.2024.e26586
56. Xie, Y., et al.: An omni-scale global-local aware network for shadow extraction in remote sensing imagery. ISPRS J. Photogramm. Remote. Sens. **193**, 29–44 (2022). https://doi.org/10.1016/j.isprsjprs.2022.09.004

57. Yamanakkanavar, N., Lee, B.: Using a patch-wise m-net convolutional neural network for tissue segmentation in brain MRI images. IEEE Access **8**, 120946–120958 (2020). https://doi.org/10.1109/ACCESS.2020.3006317
58. Yaqub, M., et al.: State-of-the-Art CNN optimizer for brain tumor segmentation in magnetic resonance images. Brain Sci. **10**(7), 427 (2020). https://doi.org/10.3390/brainsci10070427
59. Zhang, H., et al.: Efficient brain tumor segmentation with lightweight separable spatial convolutional network. ACM Trans. Multimedia Comput. Commun. Appl. (2024). https://doi.org/10.1145/3653715. https://dl.acm.org/doi/10.1145/3653715
60. Zhang, L., Peng Lim, C., Liu, C.: Enhanced bare-bones particle swarm optimization based evolving deep neural networks. Expert Syst. Appl. **230**, 120642 (2023). https://doi.org/10.1016/j.eswa.2023.120642
61. Zhou, T., Yu, Z., Cao, Y., Bai, H., Su, Y.: Study on an infrared multi-target detection method based on the pseudo-two-stage model. Infrared Phys. Technol. **118**, 103883 (2021). https://doi.org/10.1016/j.infrared.2021.103883

Learning Discretized Bayesian Networks with GOMEA

Damy M. F. Ha[1(✉)], Tanja Alderliesten[1], and Peter A. N. Bosman[2]

[1] Leiden University Medical Center, Radiotherapy, Leiden, The Netherlands
D.M.F.ha@lumc.nl, T.Alderliesten@lumc.nl

[2] Centrum Wiskunde and Informatica, Evolutionary Intelligence, Amsterdam, The Netherlands
peter.bosman@cwi.nl

Abstract. Bayesian networks model relationships between random variables under uncertainty and can be used to predict the likelihood of events and outcomes while incorporating observed evidence. From an eXplainable AI (XAI) perspective, such models are interesting as they tend to be compact. Moreover, captured relations can be directly inspected by domain experts. In practice, data is often real-valued. Unless assumptions of normality can be made, discretization is often required. The optimal discretization, however, depends on the relations modelled between the variables. This complicates learning Bayesian networks from data. For this reason, most literature focuses on learning conditional dependencies between sets of variables, called structure learning. In this work, we extend an existing state-of-the-art structure learning approach based on the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) to jointly learn variable discretizations. The proposed Discretizing Bayesian Network GOMEA (DBN-GOMEA) obtains similar or better results than the current state-of-the-art when tasked to retrieve randomly generated ground-truth networks. Moreover, leveraging a key strength of evolutionary algorithms, we can straightforwardly perform DBN learning multi-objectively. We show how this enables incorporating expert knowledge in a uniquely insightful fashion, finding multiple DBNs that trade-off complexity, accuracy, and the difference with a predetermined expert network.

Keywords: Bayesian networks · Evolutionary Algorithms · Multi-objective Optimization · Explainable AI · Discretization

1 Introduction

Bayesian Networks (BNs) [14,20] are probabilistic graphical models that model relationships between random variables under uncertainty. The structure of the relationships between variables is captured in a Directed Acyclic Graph (DAG). The process of optimizing the DAG to fit given (tabular) data as good as possible, which is often called structure learning, has been extensively researched in the

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 352–368, 2024.
https://doi.org/10.1007/978-3-031-70071-2_22

literature (e.g., [14,19]) and applied to many real-world applications, including in the medical domain [7,21,28], geology and environmental modeling domain [2,16,22], and (risk and safety) management [11,23,29].

In the real-world scenarios, it is not uncommon to have a mix of discrete and continuous random variables. Achieving optimal incorporation of continuous variables is however not straightforward. In the literature, there are various methods to extend discrete BNs with continuous variables. For example, a common method is to call upon a domain expert, who is tasked to pre-discretize continuous variables before structure learning or to model the continuous variables with a parametric distribution. It might however be difficult to consult a domain expert or they might not always be able to correctly model the variables. Non-parametric modelling of variables [5,12] on the other hand, does not require expert knowledge. However, in non-parametric models, normality is usually assumed. Discretization techniques [8–10,16,26] offer an alternative as neither expert knowledge is required a priori, nor must the assumption of normality hold. The optimal discretization however, depends on the relations modelled between the variables, necessitating simultaneous optimization.

In this work, we extend a state-of-the-art structure learning approach based on the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) family of algorithms [19] to jointly learn variable discretizations. The proposed Discretizing Bayesian Network GOMEA (DBN-GOMEA) is compared to the existing state-of-the-art DBN learning algorithm on randomly generated problems. For the task of retrieving randomly generated ground-truth networks, we find that DBN-GOMEA obtains similar or better performance than the state-of-the-art. Moreover, leveraging key strengths of EAs in multi-objective optimization, it is possible to straightforwardly perform DBN learning multi-objectively. The proposed approach is fundamentally different from e.g., [27], where a bi-objective search is performed on (proxies of) the accuracy and complexity and e.g., [1], where prior knowledge is included in the search by altering prior model probabilities according to expert knowledge. Our multi-objective approach leverages a tri-objective search to incorporate expert knowledge in a uniquely insightful fashion that enables finding multiple DBNs that trade-off (proxies of) model accuracy, complexity, and difference to a pre-determined expert network. The code is available at: https://github.com/damyha/dbn_gomea.

2 Discrete Bayesian Networks

A BN B is defined using a DAG G, which represents $\mathbf{X} = \{X_1, \cdots, X_N\}$ random variables. For each random variable X_i there is a node in G with which a (conditional) probability distribution $P(X_i|\text{pa}(X_i))$ is associated, where the probability of X_i is conditioned on the parent nodes of X_i in G, i.e., $\text{pa}(X_i)$. An example of G is shown in Fig. 1, where $\text{pa}(X_3) = \{X_1, X_2\}$, and X_3 is a parent of X_4. Node X_3, together with spouse X_6 are also parents of X_5. Given G and all conditional probabilities Θ, the probability distribution over all variables (data features) $\mathbf{X}$ can be written as a product of the individual conditional node probabilities, as is shown in Eq. 1.

$$\mathrm{P}(X_1, \cdots, X_N) = \prod_{i=1}^{N} \mathrm{P}(X_i | \mathrm{pa}(X_i)) \tag{1}$$

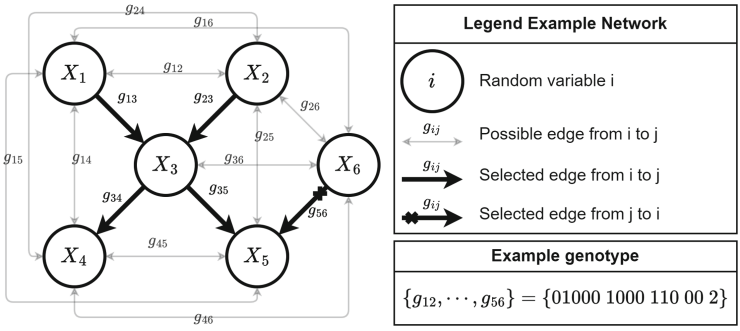


Fig. 1. Example of a DAG representing a BN (black) and all possible edges (grey). (Color figure online)

2.1 BN-GOMEA

In recent work, a state-of-the-art score-based BN structure learning algorithm was developed, called BN-GOMEA [19]. BN-GOMEA employs an Evolutionary Algorithm (EA) from the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) family. BN-GOMEA was found to outperform other EAs, greedy hill-climbing, and tabu-list based algorithms such as Ordering-Based Search, Sparse Candidate, and Max-Min Hill-Climbing. In this work, we will build upon BN-GOMEA. Therefore, key features of BN-GOMEA will be summarized.

In BN-GOMEA, BN structures are learned by encoding BNs as a string of discrete variables, one for each possible edge between two random variables i and j ($i \neq j$). The value each variable can take is 0, meaning no edge between i and j, 1, meaning a directed edge from i to j or 2, meaning a directed edge from j to i. The encoding uses $\ell_{\text{total}} = \frac{N}{2}(N-1)$ variables to represent all edges in a network of N random variables. An example of a BN with a genotype representation is given in Fig. 1. The encoding, however, permits cyclic networks. A repair operator involving depth-first search is used to remove any detected cycles. During the depth-first search, the last edge that completes a cycle is removed.

BN-GOMEA employs a linkage model that captures interdependencies between problem variables. The type of linkage model used, consists of Family of Subset (FOS) elements. Each FOS element is a set of indices indicating dependencies between problem variables identified by those indices. The FOS elements used in BN-GOMEA are governed by a linkage tree (Lt), a hierarchical clustering structure that follows from merging groups of variables iteratively

until one set containing all variables remains. Each set of variables in the hierarchical clustering is a FOS element used by BN-GOMEA. Hierarchical clustering is performed using the mutual information metric and using the population to compute the required probabilities. The Gene-pool Optimal Mixing (GOM) variation operator in BN-GOMEA subsequently leverages this Lt. Each solution in the population undergoes GOM. First, the solution is cloned. Then, the Lt is traversed in a random order. For each FOS element in the Lt, a random donor solution is selected from the population. The variables indicated by the FOS element are then copied from the donor to the offspring solution. Changes resulting in worse fitness are reverted, otherwise they are kept.

Other than the GOM operator, BN-GOMEA's excellent performance is also attributed to the ability to employ partial evaluations. Each GOM step only changes part of a solution. Partial evaluations allow efficient recalculations of fitness if only few variables are changed. This requires decomposable fitness functions, which is the case for BN-GOMEA's BDeu function, a commonly used BN learning fitness function [6]. Another reason for BN-GOMEA's success is that a local search operator is additionally used. Upon initialization and after creating offspring using GOM, local search is applied to every solution. The local search operator traverses all variables in a random order and assesses the fitness when an alternative value is used for a variable, i.e., any value in $\{0, 1, 2\}$ different from the current value. Only if the best option result in a better fitness value, the variable value is changed accordingly.

Finally, BN-GOMEA also makes use of an Interleaved Multi-start Scheme (IMS), which runs multiple populations of various sizes side by side, avoiding the need to tune the population size manually. Specifically, the IMS ensures that a population of size n_{pop}, executes 4 generations before a population of $2 \cdot n_{\text{pop}}$ executes a single generation, starting from a base population of size 2.

3 Discretization of Continuous Random Variables in Bayesian Networks

3.1 DBN-GOMEA

We extend BN-GOMEA such that continuous random variables can be taken into consideration without the need to discretize them a priori, i.e., the variables are discretized during structure learning. We refer to this extension as Discretizing BN-GOMEA (DBN-GOMEA).

First, the BDeu score used in BN-GOMEA is replaced by a density-based fitness function. Reasons are: 1) Discretization turns the continuous data into a histogram, which is essentially still a probability density function (that assumes a uniform data distribution within each bin (also referred to as discretization)). 2) The BDeu with discretization inherently maximizes the log-likelihood by placing all continuous samples into a single bin, since the likelihood of being in a single bin becomes 1.

The fitness function that is used is the log-likelihood calculated across these densities, as is shown in Eq. 2, where $\mathbf{x_i} \in \mathbb{R}^N$ is a training sample i from a

training data set of size n. To make the fitness invariant to the data range, the data is normalized to $[0, 1]$ prior to calculating the densities. This is similar to what has been done in [26]. As a penalty term, the penalty term in the BIC score [25] is used, where the model complexity $\mathrm{C}(G)$ is dependent on the number of parent discretizations: $|\mathrm{pa}(X_i)|$, the number of discretizations of X_i: $|X_i|$, and n, as shown in Eq. 3. This results in the fitness function displayed in Eq. 4.

$$\mathrm{LL}(\mathbf{X}, G) = \prod_{i=1}^{n} \log(f_{\mathrm{density}}(\mathbf{x_i})) \tag{2}$$

$$\mathrm{C}(G) = \sum_{i=1}^{N} |\mathrm{pa}(X_i)| \cdot (|X_i| - 1) \cdot \log(\frac{n}{2}) \tag{3}$$

$$\mathrm{fitness}(\mathbf{X}, G) = \mathrm{LL}(\mathbf{X}, G) - \mathrm{C}(G) \tag{4}$$

To discretize continuous variables, we consider two common discretization methods in this paper, namely: Equal-Width (EW) and Equal-Frequency (EF). In EW discretization, data is split into 'k' equally ranged bins. In EF discretization, data is sorted and split into 'k' equally filled bins. In DBN-GOMEA, to facilitate discretization, the discretization count k for each continuous random variable is optimized by appending all k's to BN-GOMEA's solution representation. I.e., the representation is enlarged with N_c variables where N_c is the number of continuous variables.

Because the solution representation is altered, the local search operator of BN-GOMEA is changed accordingly. DBN-GOMEA retains the local search operator for the network topology. For problem variables that represent discretization counts, their values are increased and decreased by one, i.e., $\{k - 1, k + 1\}$, when performing local search. Discretization counts are furthermore constrained to a minimum and maximum value. In this work, the discretizations are constrained to a range of 2 to 15 to keep computation times feasible.

3.2 LDBN: The Current State-of-the-Art

In [8], a discretization method is proposed that finds a discretization Λ that maximizes a likelihood score, given a BN structure. The method uses Bayes' rule to maximize: $P(\Lambda) \cdot P(D|\Lambda)$, where $P(\Lambda)$ is the prior of a discretization policy and $P(D|\Lambda)$ is the probability of the data given the discretization policy. The likelihood is formulated by making assumptions, one of which is that the prior probability of a discretization boundary between two unique consecutive sample values is proportional to their difference. Dynamic programming is used to maximize the likelihood. By doing pre-calculations, the discretization runtime is reduced to $\mathcal{O}(r \cdot n^2)$, where r is a constant and n is the sample size.

In [8], LDBN is introduced as a state-of-the-art algorithm that combines the Bayesian discretization method with a structure learning algorithm. Structure learning in LDBN proceeds by first applying EW discretization to all continuous

random variables, where the number of discretizations is the largest number of instantiations amongst all discrete random variables. Subsequently, an arbitrary random variable is selected as a starting point. The remaining random variables are randomly selected and sequentially added to the BN structure as child nodes. An edge between the potential child node and BN structure materializes when the K2 score of the network improves when adding the edge. Upon edge addition, the Bayesian discretization is applied on the new node and its Markov blanket in a sequentially random order. Given the inherent randomness in both the structure learning and discretization process, LDBN executes both processes multiple times. We kindly refer the interested reader to [8] for more details.

3.3 Post-structure Learning Discretization

In the literature, EW and EF discretization are commonly used. However, real-world data rarely align perfectly with EW or EF distributions. The Bayesian method performs a more fined-grained optimization. However, as later will be shown, it is computationally expensive, especially compared to EW and EF discretization. Consequently, it might be interesting to refine the discretization boundaries post structure learning.

In this paper, we perform discretization with the Bayesian method mentioned above after structure learning on networks obtained with EW and EF discretization. For comparison, we use a state-of-the-art real-valued optimization algorithm (Real-Valued GOMEA (RV-GOMEA) [3,4]) to optimize the bin boundaries using the density fitness function (Eq. 4). This may increase the likelihood of the learned BN, without changing the complexity. For this boundary optimization all boundaries are encoded in a single solution. Rather than directly optimizing the boundaries, the unique data values $\mathbf{u}$ of each continuous random variable are sorted, after which the bin boundaries are optimized as sample indices. The boundary at sample index i is defined as the midway point between unique sample u_i and u_{i+1}. Compared to direct boundary optimization, optimizing the sample indices makes the flat landscape between samples indices to be of equal size.

4 Multi-objective Learning

A major limitation of Single-Objective (SO) BN learning is that the weight of the complexity term is not straightforward to set [27]. Furthermore, an expert may have their own beliefs about what the Bayesian network should look like. Taking a Multi-Objective (MO) perspective can offer a solution to these issues. First, with MO search, the user does not need to set the penalty factor a priori. Furthermore, the search returns many networks from which a domain expert can choose a network that matches (partly) with their own prior belief or discover new knowledge this way.

A straightforward way to do MO search is to optimize (a proxy of) accuracy and model complexity as in e.g., [27]. For GOMEA, MO variants exist that are

direct extensions of the SO versions, necessitating no further adaptions the genotype or the variation operator. Using the density function as is, an MO problem formulation can be straightforwardly obtained by using Eq. 2 and 3 as separate objectives. See Sect. 4.1 for more on this. The networks obtained through the MO search, or a subset thereof, can then be shown to an expert, who chooses the most appropriate network, trading off the fit to the data and the complexity of the network. The inclusion of expert knowledge however, could provide additional guidance to the search by optimizing a third objective: the distance to an a priori determined expert network. To this end, we use the Kullback-Leibler divergence (KL) as shown in Eq. 5, where $P(\mathbf{X})$ is the probability distribution of the expert network, $Q(\mathbf{X})$ is the probability distribution of a candidate network, and $\mathcal{X}$ is the sample space. The KL divergence is 0 when two probability distributions are identical, and is larger than 0 otherwise. An example of solutions in objective space of an MO run is shown in Fig. 2. Figure 2 shows the trade-off between density against complexity, with the KL divergence to the expert network projected as a heat-map. The expert and ground truth networks are also shown, along with the respective BNs. BNs of 2 solutions surrounding the ground truth are also shown.

$$D_{KL}(P||Q) = \sum_{\mathbf{X}\in\mathcal{X}} P(\mathbf{X}) \log\left(\frac{P(\mathbf{X})}{Q(\mathbf{X})}\right) \tag{5}$$

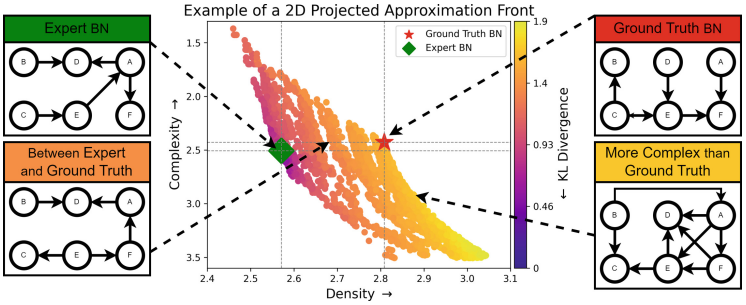


Fig. 2. Example of an approximation front of an MO run along with the ground truth and expert networks. BNs of solutions surrounding the ground truth are also shown.

4.1 MO-DBN-GOMEA

For the MO optimization, the Multi-Objective version of GOMEA (MO-GOMEA) [18] is used. MO-GOMEA can similarly exploit partial evaluations to achieve enhanced efficiency. MO-GOMEA uses domination-based optimization, i.e., it uses the concept of Pareto dominance to find better solutions (test for improvements). A solution is said to Pareto dominate another solution if it

is not worse in any objective and better in at least one objective. In this work, we will build upon MO-GOMEA to create MO-DBN-GOMEA. Therefore, key features of MO-GOMEA will be briefly summarized.

Given m objectives to be optimized, a population in MO-GOMEA is partitioned into c clusters of equal sizes. For each cluster, an Lt is learnt, similar as in Sect. 2.1. The m clusters with the best mean objective values are selected to optimize the respective individual objective functions using the SO GOM operator. For the remaining clusters, the MO GOM operator is used. Different from SO GOM, an altered solution is accepted in MO GOM if: 1) the altered solution dominates the unaltered solution, 2) the altered solution has the same objective values, 3) the altered solution is not dominated by any solution in the elitist archive. In this work we used an elitist archive with a maximum size of 10,000 solutions to maximize solution collection while balancing computation time. Which solutions to accept into the archive once the limit is reached, is governed by an adaptive gridding technique. For details, see [18].

MO-GOMEA uses the IMS to regulate its population size and number of clusters c. By default, the population size starts at 8 and is multiplied by 2 for every new population. The number of clusters starts at $m + 1$, and is incremented by 1 for every new population.

We extended MO-GOMEA to solve discrete problems beyond only binary problems by following the suggestions in [18]. The most important change, is replacing the binary Lt with the discrete Lt of [19]. Furthermore, the BN structure representation, as proposed in Sect. 2 is integrated into MO-GOMEA. The new structure learning algorithm is dubbed MO-DBN-GOMEA.

5 Experiments and Results

5.1 Network Generation

In this work, randomly generated BN structures and probability distributions are used to assess the performance of the algorithms. The network generator algorithm of [13] is used to generate random BNs. Probability distributions are generated using a novel method described below. Data sets are subsequently sampled from the ground truth networks to use for learning. In [13], random BN structures are generated under constraints. The maximum number of parent random variables is set to 6. The maximum number of edges in a network is set to be at most 40% of all possible edges ℓ_{total}. These constraints have been chosen, such that networks can be evaluated within reasonable time.

The probability distributions are randomly generated by first categorizing the random variables into discrete and continuous. Each network contains 10% discrete random variables with a minimum of at least one discrete variable. Both discrete and continuous random variables are then randomly assigned between 2 and 6 discretizations. E.g., 5 discretizations can produce the following sample values: $\{1, 2, 3, 4, 5\}$. A discrete probability table is then generated for each random variable, that maps the possible parent values to a probability of a specific discretization value. The probability tables are generated in three ways: EW, EF

or random probability distributions. For EF probability distributions, the probability of sampling any value is equiprobable. For EW and random probability distributions, random probability tables are generated.

Discrete samples can now be retrieved. For the continuous variables, however, the discrete probability tables must be converted into continuous probability distributions. For this, a mapping is generated that maps each discrete value to a continuous range from which values can be uniformly sampled. E.g., if a continuous random variable has 3 discretizations with ranges: $[1.0, 2.0\rangle$, $[2.0, 2.5\rangle$, $[2.5, 3.5\rangle$, and a discrete value of 2 is sampled, a continuous sample is produced by uniformly sampling from $[2.0, 2.5\rangle$. The sample ranges are designed to be adjacent and non-overlapping, and are randomly generated. For EW probability distributions, the sample ranges are set to be equally spaced. For EF and random probability distributions, the sample ranges are determined randomly.

5.2 Metrics

Algorithm performance is evaluated using network structure metrics, specifically the accuracy and sensitivity. The accuracy quantifies the proportion of correctly identified edges TP and correctly identified absent edges TN in a candidate network relative to all possible edges ℓ_{total} of the ground truth network (see Eq. 6). Sensitivity measures the proportion of correctly identified edges TP as a proportion of the total edges in the ground truth network: ℓ_{edges} (see Eq. 6). For TP, directionality of the edge is not considered, due to intrinsic limitations of score-based functions on identifying correct edge directionality. Note that this definition is different from other works such as e.g., [17]. The performance of the discretization, with respect to the ground truth network, is evaluated with the KL divergence as previously outlined in Sect. 4.

$$\text{Accuracy} = \frac{TP + TN}{\ell_{\text{total}}} \qquad \text{Sensitivity} = \frac{TP}{\ell_{\text{edges}}} \tag{6}$$

5.3 Single-Objective Scalability

Single-Objective Scalability in Terms of Sample Size.
The scalability in terms of sample size is shown for various algorithms in Fig. 3. For this, 30 ground truth networks with 8 random variables each were generated with EW, EF and random probability distributions. For the KL divergence, 50.000 test samples are generated. DBN-GOMEA with EW, EF, and Bayesian Discretization (BD), as well as the algorithm from [8] (LDBN) are compared. DBN-GOMEA-EF and DBN-GOMEA-EW are run on an Intel E5-2690, utilizing one core, 2GB of memory and 24 h of computation time per run. The memory requirements of BD scale with $\mathcal{O}(n^2)$. Therefore, LDBN and DBN-GOMEA-BD are run on a newer E5-4650 with 20GB of memory per run. As only one E5-4650 was available, it was infeasible to execute all algorithms on the E5-4650.

Figure 3 shows that, in general, DBN-GOMEA with the appropriate discretization techniques finds better network structures as well as better KL divergence when the sample size grows. DBN-GOMEA-EW furthermore obtains perfect network retrieval for EW distributions, given ≥ 6400 samples. Conversely, DBN-GOMEA-EF fails to achieve perfect network retrieval on EF distributions because the EF data, being sampled from the ground truth network, is not perfectly EF distributed. In contrast, EW discretization is less sensitive to variations in EW distributed data. Interestingly, Bayesian Discretization often runs out of memory when the sample size grows. The time at which the best solution of a run was found (Time until best solution) therefore plummets, as only a few small networks are evaluated. For small sample sizes however, excluding in case of EW distributions, DBN-GOMEA-BD can find better network structures and similar or better KL divergence compared to other DBN-GOMEA variants.

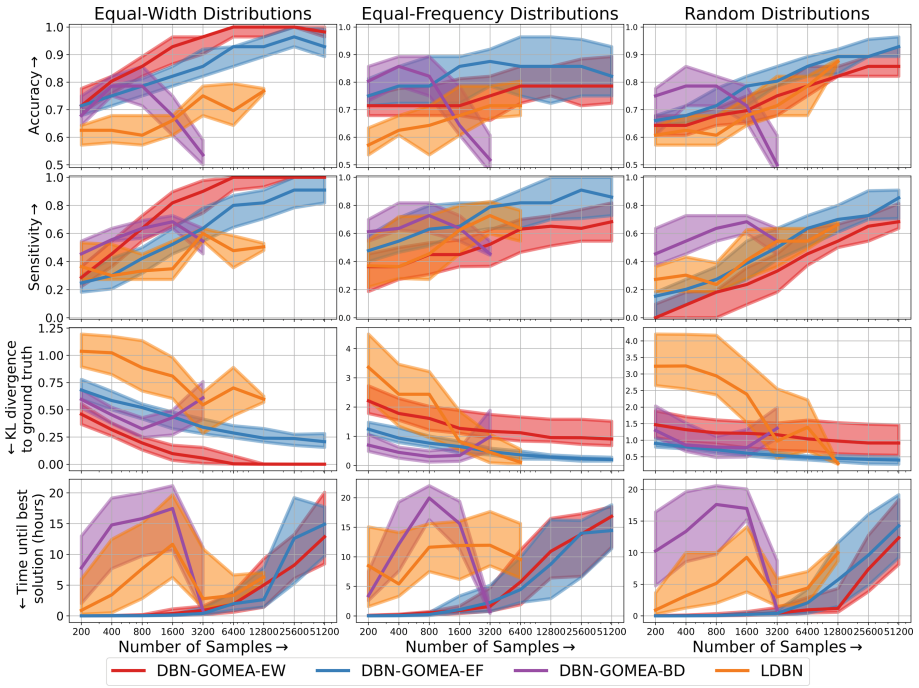


Fig. 3. Scalability in terms of sample size for 30 random networks with 8 random variables having EW, EF and Random probability distributions. Solid lines represent medians, while shaded areas cover the interquartile ranges. Arrows on the y-axis indicate metric improvement direction.

Testing Differences in the KL Divergence of Random Distributions. To validate the results, differences in the KL divergence of the random probabil-

ity distributions are examined with a Mann-Whitney U statistical test. Pair-wise testing is done with respect to the best average results. This results in 18 tests. Table 1 displays the mean ± the standard deviation of the KL divergence. Note that the mean is different than median shown in Fig. 3. The best average results or statistically equivalent results are shown in bold. An alpha value of 0.05 is used and is Bonferroni corrected for 45 tests as 27 more tests will later be performed. Table 1 shows that DBN-GOMEA-EF is the best, with DBN-GOMEA-BD being comparable when there a few samples.

Table 1. The average KL divergence and standard deviations with respect to the ground truth networks with random distributions (Fig. 3, right column) for various algorithms. The best average and statistically equivalent KL divergences are in bold.

Number of samples	DBN-GOMEA-EW	DBN-GOMEA-EF	DBN-GOMEA-BD	LDBN
200	1.57 ± 0.58	**0.94** ± 0.21	1.62 ± 0.89	3.43 ± 1.27
400	1.45 ± 0.59	**0.83** ± 0.19	**1.16** ± 0.78	3.39 ± 1.17
800	1.38 ± 0.61	**0.73** ± 0.18	**0.94** ± 0.65	3.37 ± 1.24
1600	1.32 ± 0.64	**0.64** ± 0.17	**1.02** ± 0.73	2.66 ± 1.48
3200	1.26 ± 0.64	**0.57** ± 0.18	1.82 ± 1.43	–
6400	1.21 ± 0.66	**0.50** ± 0.17	–	–
12800	1.16 ± 0.66	**0.46** ± 0.17	–	–
25600	1.09 ± 0.63	**0.43** ± 0.18	–	–
51200	1.07 ± 0.62	**0.41** ± 0.18	–	–

Single-Objective Scalability in Terms of Random Variables.
The scalability in terms of the number of random variables (i.e., nodes) in a network is shown in Fig. 4. For this, 30 ground truth networks, with random probability distributions, are generated per ground truth network size. Each run was performed using 500 training samples, on a single core of an AMD Genoa 9654, with 2GB of memory and a computation budget of 24 h.

Figure 4 shows that DBN-GOMEA-BD obtains more accurate and more sensitive network structures if there are only a few nodes. However, this superiority diminishes beyond more than 12 nodes. The time it takes to find the best solution however nears the computation budget of 24 h. This raises the question if DBN-GOMEA-BD needs more time to converge. The network metrics obtained by other algorithms for more than 12 nodes seem to be similar.

Testing Differences in the KL Divergence.
Despite having similar network structures, the KL divergence varies per algorithm due to differences in discretizations. To test for statistical differences, a

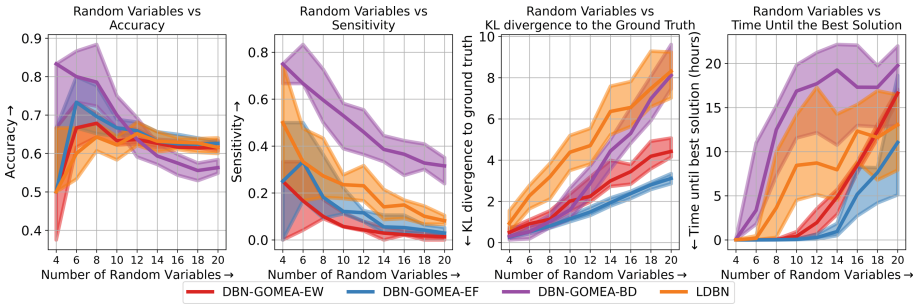


Fig. 4. Scalability in terms of random variables. For each network size on the x-axis, 30 ground truth networks with 500 samples and random probability distributions were generated. Solid lines represent medians, while shaded areas cover the interquartile ranges. Arrows on the y-axis indicate metric improvement direction.

Mann-Whitney U statistical test is performed. Pair-wise testing is done with respect to the best average results. This results in 27 tests. Table 2, shows the mean ± the standard deviation of the KL divergence. The results with the best average and statistically equivalent results are shown in bold. An alpha value of 0.05 is used, with a Bonferroni correction of 45 as 18 tests were already performed. Table 2 shows that, similar to Table 1, DBN-GOMEA-EF performs best, with DBN-GOMEA-BD performing comparably when there are few random variables.

Table 2. The average KL divergence ± the standard deviation of various algorithms optimized on 30 ground truth networks with 500 samples and random probability distributions (Fig. 4, 3rd plot from the left) for various number of random variables. The best KL scores and the statistically equivalent results are marked in bold.

Random Variables	DBN-GOMEA-EW	DBN-GOMEA-EF	DBN-GOMEA-BD	LBDN
4	**0.58** ± 0.45	**0.29** ± 0.14	**0.45** ± 0.40	1.11 ± 0.81
6	0.98 ± 0.56	**0.51** ± 0.16	**0.57** ± 0.40	2.31 ± 0.86
8	1.26 ± 0.51	**0.78** ± 0.18	**1.09** ± 0.63	3.04 ± 1.05
10	2.11 ± 0.69	**1.15** ± 0.22	**1.89** ± 1.16	4.42 ± 1.35
12	2.57 ± 0.91	**1.54** ± 0.25	3.02 ± 1.06	4.63 ± 1.07
14	3.29 ± 0.93	**1.98** ± 0.28	4.49 ± 1.32	6.42 ± 1.46
16	3.47 ± 0.82	**2.32** ± 0.29	5.69 ± 1.25	6.97 ± 1.73
18	4.32 ± 1.06	**2.76** ± 0.30	6.95 ± 1.43	7.85 ± 1.74
20	4.66 ± 0.82	**3.14** ± 0.33	8.33 ± 1.77	8.35 ± 1.52

5.4 Post-Structure Learning Discretization

Figures 3 and 4 showed that DBN-GOMEA-EW and DBN-GOMEA-EF retrieve relatively accurate networks relatively fast. However, both algorithms perform a coarse discretization compared to e.g., BD. To explore post-structure learning effects, network structures from Fig. 3, obtained using DBN-GOMEA-EW and DBN-GOMEA-EF on data with random probability distributions, undergo further discretization using BD and RV-GOMEA. BD and RV-GOMEA were both given a computation budget of 24 h and were run on an E5-4650 with 20 GB of memory and E5-2690 with 2 GB of memory, respectively.

The effect of further discretization after completing structure learning, together with the original obtained network structure and discretization is shown in Fig. 5. Note that the initial 24 h spent on structure learning is not included in Fig. 5. Figure 5 shows that when RV-GOMEA is applied (purple and orange), the median KL divergence improves compared to not doing post-structure learning discretization (red and blue) regardless whether EW or EF discretization was used to obtain the structure. Conversely, the BD method worsens the KL divergence in case of limited samples. Beyond 12800 samples, BD runs out of memory.

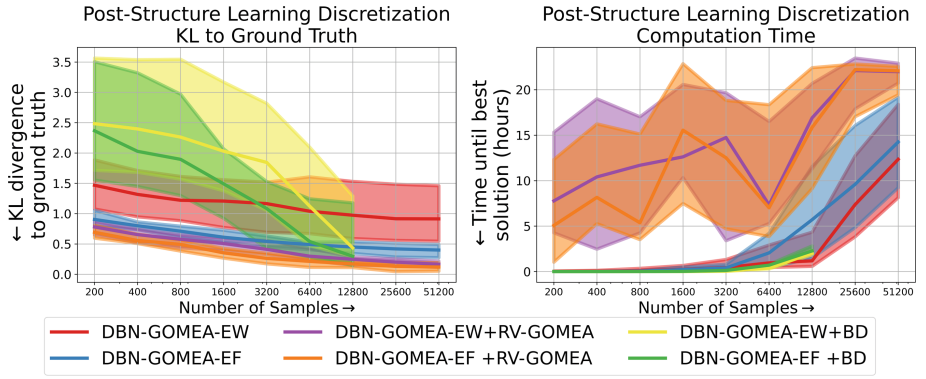


Fig. 5. Networks structures with random probability distributions of Fig. 3, obtained using DBN-GOMEA-EW/EF, are further discretized. Solid lines represent medians, while shaded regions covert the interquartile ranges. Arrows on the y-axis indicate the metric improvement direction.

5.5 Multi-objective Experiment

Additional to the ground truth networks, expert networks that represent domain expert hypotheses are also generated. Expert networks mirror the ground truth by copying 50% of the edges of the ground truth (randomly selected), i.e., 50% of ℓ_{edges}. This is similar to what has been done in [1]. Additionally, after copying edges, the expert networks also randomly generate edges that do not appear in

the ground truth network. The number of incorrect edges is also set to 50% of ℓ_{edges}. Expert networks with cycles are rejected and resampled. For the continuous variables, the experts randomly discretize the data randomly into 2 to 4 bins.

We compare MO-DBN-GOMEA with EW and EF discretization. SO algorithms DBN-GOMEA-EW and DBN-GOMEA-EF are also run. In an explainable AI setting, overly complex networks might face rejection by an expert. For this reason, proposed networks with a complexity (Eq. 3) larger than 10 times the expert network are assigned a constraint value proportionate to the difference in complexity. This threshold, however, is problem- and expert dependent and here only serves as an example. The number of maximum discretizations is also decreased from 15 to 9 to reduce the complexity of proposed solutions.

The results of the MO search on 30 randomly generated networks with 10 random variables and random probability distributions is shown in Fig. 6 for various sample sizes. Each run was performed on a single core of an AMD Genoa 9654, with 2GB of memory and a computation budget of 24 h. The first two columns of Fig. 6 show the network accuracy with respect to the ground truth and expert networks. For the MO algorithms, the most accurate solutions in the elitist archive are displayed per run. For the SO algorithms, the final solution's network accuracy is shown. Interestingly, MO algorithms outperform SO algorithms in terms of network accuracy, including better accuracy with respect to the expert networks due to the explicit additional objective. The gap in accuracy between MO and SO with respect to the ground truth narrows, however, with increasing sample sizes. In Fig. 6, the KL divergence is also shown. For the MO algorithms, the best KL divergence is shown amongst all solutions with the highest network accuracy. For the SO algorithms, the final solution's KL divergence is shown. Interestingly, both the MO and SO algorithms using EF discretization obtain similar KL divergence to the ground truth network.

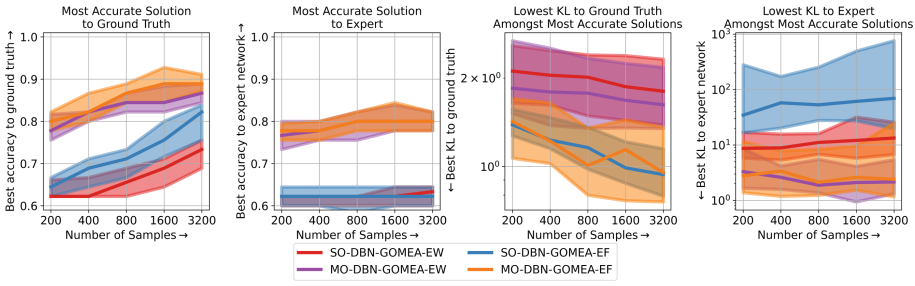


Fig. 6. MO vs SO scalability in terms of sample size on ground truth networks with 10 nodes, random probability distributions, and random expert networks. The solid lines are medians, while the shaded areas encompass the first and third quartiles. The arrows on the y-axis point in the direction of improvement per metric.

6 Discussion

In the solution encoding, network variables can take three values, namely: $\{0, 1, 2\}$. When e.g., Equal Width (EW) or Equal Frequency (EF) discretization is applied, the number of discretizations for every continuous variable is also encoded in the solution, which can take a value between 2 and a maximum value. Due to differing value ranges for network and discretization variables, the linkage tree tends to cluster discretization variables together, in the lower parts of the tree. Mixing the network variables and discretization variables could potentially speed up the optimization, as graphically, discretization variables and edges are structurally related. For this, normalizing the mutual information could help.

In Sect. 5.3, the effect of the sample size on the network accuracy was shown. Despite large sample sizes, DBN-GOMEA-EF was unable to fully re-obtain the EF ground truth network. Conversely, DBN-GOMEA with Bayesian Discretization (BS) obtained better KL divergence than DBN-GOMEA-EF for smaller sample sizes. This suggests the need for more scalable and sophisticated discretization techniques, beyond EW and EF discretization. An interesting approach would be to use a mixed-integer algorithm, such as [15,24]. The integer-based network structure encoding could then simultaneously be optimized with the real-valued bin boundary optimization.

In this work, a multi-objective Bayesian network learning algorithm was also introduced. However, only EW and EF discretization have been ran, because BD was found to be too computationally expensive to run.

In some real-world domains, blindly trusting machine learning models is not acceptable from a legal aspect. The multi-objective approach proposed in this work however could be useful when used as an advisory model, as it provides the possibility to inspect multiple possible models and trade-off between complexity and accuracy. We therefore consider exploring the potential added value of our approach from an explainable AI perspective, by having domain experts interact with the found works, to be interesting future work.

7 Conclusion

In this work, for the first time, a full Bayesian network learning algorithm based on a modern model-based EA (GOMEA) is presented, which discretizes continuous variables while performing structure learning. The single-objective variant of the proposed algorithm (DBN-GOMEA) obtains similar or better results than the state-of-the-art when tasked to retrieve randomly generated ground-truth networks. The multi-objective variant of the proposed algorithm (MO-DBN-GOMEA) not only obtains similar or better the results than the single-objective variant, it also enables incorporating expert knowledge in a uniquely insightful fashion, finding multiple discrete Bayesian networks that trade-off complexity, accuracy, and the difference with a pre-determined expert network.

Acknowledgements. This research is part of the research programme Open Competition Domain Science-KLEIN with project number OCENW.KLEIN.111, which is financed by the Dutch Research Council (NWO). We also thank NWO for the Small Compute grant on the Dutch National Supercomputer Snellius.

References

1. Amirkhani, H., Rahmati, M., Lucas, P.J.F., Hommersom, A.: Exploiting experts' knowledge for structure learning of Bayesian networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(11), 2154–2170 (2017). https://doi.org/10.1109/TPAMI.2016.2636828
2. Beuzen, T., Marshall, L., Splinter, K.D.: A comparison of methods for discretizing continuous variables in Bayesian networks. Environ. Model. Softw. **108**, 61–66 (2018)
3. 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. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3071178.3071272
4. Bouter, A., Bosman, P.A.N.: A joint python/c++ library for efficient yet accessible black-box and gray-box optimization with gomea. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO 2023), pp. 1864–1872. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3583133.3596361
5. Bubnova, A.V., Deeva, I., Kalyuzhnaya, A.V.: Mixbn: library for learning Bayesian networks from mixed data. Procedia Comput. Sci. **193**, 494–503 (2021)
6. Buntine, W.: Theory refinement on Bayesian networks. In: Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence (UAI 1991), pp. 52–60. Morgan Kaufmann Publishers Inc., San Francisco (1991)
7. de Campos, L.M., Cano, A., Castellano, J.G., Moral, S.: Bayesian networks classifiers for gene-expression data. In: 2011 11th International Conference on Intelligent Systems Design and Applications, pp. 1200–1206 (2011). https://doi.org/10.1109/ISDA.2011.6121822
8. Chen, Y.C., Wheeler, T.A., Kochenderfer, M.J.: Learning discrete Bayesian networks from continuous data. J. Artif. Int. Res. **59**(1), 103–132 (2017)
9. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 1022–1029. Morgan Kaufmann, Chambery (1993)
10. Friedman, N., Goldszmidt, M.: Discretizing continuous attributes while learning bayesian networks. In: Saitta, L. (ed.) Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco (1996)
11. Hosseini, S., Ivanov, D.: Bayesian networks for supply chain risk, resilience and ripple effect analysis: a literature review. Exp. Syst. Appl. **161**, 113649 (2020)
12. Ickstadt, K., et al.: Nonparametric Bayesian networks. In: Bayesian Statistics 9. Oxford University Press, Oxford (2011). https://doi.org/10.1093/acprof:oso/9780199694587.003.0010
13. Ide, J.S., Cozman, F.G., Ramos, F.T.: Generating random Bayesian networks with constraints on induced width. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), pp. 353–357. IOS Press, NLD (2004)

14. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press, Cambridge (2009)
15. Li, R., et al.: Mixed integer evolution strategies for parameter optimization. Evolution. Comput. **21**(1), 29–64 (2013). https://doi.org/10.1162/EVCO_a_00059
16. Lima, M.D., Nassar, S.M., Rodrigues, P.I.R., Filho, P.J.F., Jacinto, C.M.: Heuristic discretization method for Bayesian networks. J. Comput. Sci. **10**(5), 869–878 (2014)
17. Liu, Z., Malone, B., Yuan, C.: Empirical evaluation of scoring functions for Bayesian network model selection. BMC Bioinformatics **13**, S14 (2012). https://doi.org/10.1186/1471-2105-13-S15-S14
18. Luong, N.H., La Poutré, H., Bosman, P.A.: Multi-objective gene-pool optimal mixing evolutionary algorithm with the interleaved multi-start scheme. Swarm Evol. Comput. **40**, 238–254 (2018)
19. Orphanou, K., Thierens, D., Bosman, P.A.N.: Learning Bayesian network structures with Gomea. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2018), pp. 1007–1014. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3205455.3205502
20. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
21. Reijnen, C., et al.: Preoperative risk stratification in endometrial cancer (endorisk) by a Bayesian network model: a development and validation study. PLOS Med. **17**(5), 1–19 (2020). https://doi.org/10.1371/journal.pmed.1003111
22. Ropero, R.F., Renooij, S., van der Gaag, L.C.: Discretizing environmental data for learning Bayesian-network classifiers. Ecol. Model. **368**, 391–403 (2018)
23. Rostamabadi, A., Jahangiri, M., Zarei, E., Kamalinia, M., Alimohammadlou, M.: A novel fuzzy Bayesian network approach for safety analysis of process systems; an application of HFACS and SHIPP methodology. J. Clean. Prod. **244**, 118761 (2020)
24. Sadowski, K.L., Thierens, D., Bosman, P.A.: GAMBIT: a parameterless model-based evolutionary algorithm for mixed-integer problems. Evolution. Comput. **26**(1), 117–143 (2018). https://doi.org/10.1162/evco_a_00206
25. Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
26. Suzuki, J.: Learning bayesian network structures when discrete and continuous variables are present. In: van der Gaag, L.C., Feelders, A.J. (eds.) Probabilistic Graphical Models, pp. 471–486. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11433-0_31
27. Wu, T., Qian, H., Liu, Z., Zhou, J., Zhou, A.: Bi-objective evolutionary Bayesian network structure learning via skeleton constraint. Front. Comput. Sci. **17**(6) (2023). https://doi.org/10.1007/s11704-023-2740-6
28. Zhao, G., Feng, Q., Chen, C., Zhou, Z., Yu, Y.: Diagnose like a radiologist: hybrid neuro-probabilistic reasoning for attribute-based medical image diagnosis. IEEE Trans. Pattern Anal. Mach. Intell. **44**(11), 7400–7416 (2022). https://doi.org/10.1109/TPAMI.2021.3130759
29. Zhou, Z., Yu, X., Zhu, Z., Zhou, D., Qi, H.: Development and application of a bayesian network-based model for systematically reducing safety risks in the commercial air transportation system. Saf. Sci. **157**, 105942 (2023)

Pareto-Informed Multi-objective Neural Architecture Search

Ganyuan Luo[1], Hao Li[1], Zefeng Chen[1(✉)], and Yuren Zhou[2]

[1] School of Artificial Intelligence, Sun Yat-sen University, Zhuhai, China
{luogy7,lihao75}@mail2.sysu.edu.cn, chenzef5@mail.sysu.edu.cn
[2] School of Software Engineering, Sun Yat-sen University, Zhuhai, China
zhouyuren@mail.sysu.edu.cn

Abstract. Aiming at the auto-design of powerful neural architectures with a requirement of compromising multiple objectives, this paper introduces a novel approach called *Pareto-informed Multi-objective Neural Architecture Search* (PiMO-NAS), which employs a solution generator influenced by Tchebycheff decomposition to explore the objective space of multi-objective NAS. Our methodology initiates with a transformation of discrete search space into continuous form, followed by iterative solution optimization, in which Gaussian Process (GP) surrogate models are utilized to establish a mapping from decision space to objective space. Subsequently, a solution generator, directed by preference vectors from the objective space, is designed to generate decision vectors to map the weights from the objective space back to the decision space. This solution generator is further optimized based on the gradients derived from the GP models. To ensure diversity in the solution pool, the solution generator synthesizes new candidate solutions guided by preference vectors generated by a well-designed adaptive sampler. In order to verify the performance of the proposed PiMO-NAS, a series of experiments were conducted within two typical NAS search spaces (i.e., the Once-For-All(OFA) and AutoFormer based ones, covering both convolutional neural networks and vision transformers), and more than 30 state-of-the-art NAS methods and models were employed for performance comparisons. Experimental results showcase that our approach can outperform most peers in terms of search time and solution quality, and has fantastic ability to efficiently discover high-performing neural architectures. In the OFA-based search spaces, compared with the MSuNAS, the proposed PiMO-NAS was able to achieve similar performance in two-thirds of the number of iterations, thereby saving about 24% of the search time. In the AutoFormer-based search space, we successfully approached a strong baseline formed by a single-objective evolutionary algorithm with restricted parameter quantities, approximating the entire Pareto front in a comparable timeframe.

Keywords: Neural architecture search (NAS) · Multi-objective optimization · Surrogate · Pareto front

This research is supported by the National Natural Science Foundation of China under Grants 62206313 and 62232008.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 369–385, 2024.
https://doi.org/10.1007/978-3-031-70071-2_23

1 Introduction

In recent years, a paradigm shift in computer vision has been witnessed with the advent of neural networks such as convolutional neural networks(CNNs)[18–20,29,34] and vision transformers (ViTs)[12,25,37,37]. Pioneering the revolution of deep learning (DL), CNNs have established themselves as a cornerstone in the DL field, thanks to their exceptional capability in handling image data. Meanwhile, ViTs have demonstrated remarkable capabilities in capturing long-range dependencies, a feature critical for understanding complex visual contexts, which marked a significant departure from CNNs[17]. However, the quest for efficient and powerful neural architectures in computer vision raises a critical question: *how to balance multiple conflicting objectives (such as accuracy and model size)*? As the success of CNNs and ViTs in visual recognition is contingent upon the meticulous architecture design, it requires substantial computational resources and expert knowledge. This challenge has prompted the exploration of neural architecture search (NAS) methods, which automates the process of architectural design, leading to models that are both efficient and accurate[1,21,24,32].

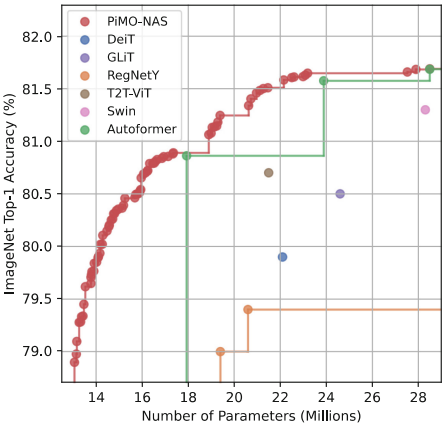


Fig. 1. Solutions obtained by our proposed PiMO-NAS in the AutoFormer-small space. Solutions obtained by PiMO-NAS outperforms previous SOTA models like DeiT and RegNetY. Note that PiMO-NAS only needs *a single search of 2.16 GPU days* to obtain numerous good-quality solutions (over 200), while AutoFormer needs *three rounds of searches (each of which spent 1.83 GPU days)* to obtain three good-quality solutions.

Among these, one-shot NAS (OSNAS) method innovatively decouples the training and searching phases in the process of neural architecture optimization[2,6,24,30,31]. A number of distinct approaches have showcased the versatility and efficiency of one-shot NAS in different contexts. For example, the AutoFormer[6], an OSNAS method based on weight entanglement, is demonstrated to provide plug-and-play ViTs for different budgets. This approach effectively identifies the optimal ViT configurations that balance computational efficiency and high performance in visual tasks. Similarly, Once-For-All (OFA)[3] represents an application of OSNAS extending to CNNs. Developed by Cai *et al.*, the OFA trains a diverse and comprehensive supernet capable of adapting to a wide range of hardware constraints and efficiency requirements with progressive shrinking technique (Fig. 1).

This work primarily focuses on multi-objective neural architecture search on one-shot NAS search space. Drawing inspiration from the key idea of Pareto set learning[23], we develop an innovative approach, named *Pareto-Informed*

Multi-Objective Neural Architecture Search (*PiMO-NAS* for short), to address a crucial challenge in deploying neural networks across varied budgets.

In brief, our major contributions in this paper are summarized as three-fold.

1. A simplified yet effective method is developed to relax the traditionally rigid discrete search space of NAS, and an innovative integration of Pareto set learning into one-shot NAS is put forward.
2. A novel adaptive sampler that can dynamically generate preference vectors is designed to address the challenge in terms of the inequality in sub-problems caused by uneven improvements in hypervolume across different Tchebycheff decomposition vectors. This can endow with our proposed PiMO-NAS algorithm a strong capability of adapting to diverse search spaces.
3. Our proposed PiMO-NAS algorithm is systemically compared with more than 30 state-of-the-art NAS methods and models. This comparative analysis showcases the potential of PiMO-NAS, highlighting its superior performance in balancing multiple objectives in neural architecture search.

The structure of this paper is organized as follows. Section 2 delves into the foundations of multi-objective NAS and also provides a detailed exposition on Pareto set learning, outlining its principles and relevance in the context of NAS. Section 3 elaborates the details of our proposed PiMO-NAS algorithm. Section 4 is dedicated to the experimental studies of our research, including the design of our experiments and the experimental results. Section 5 summarizes the key findings and contributions of our work.

2 Preliminaries

2.1 Multi-objective NAS

The NAS aims to automate the process of designing optimal neural architectures, finding the best neural network parameters by performing optimization within a predefined search space. In the search space, neural networks can be encoded as directed acyclic graphs (DAGs) of a series of hidden states and operations[14]: $s^t = o^t(s^{t-1})$, where s^t represents the t-th hidden layer and o^t is a specific operation (e.g., fully connection, convolution, pooling and activation functions) for converting s^{t-1} to s^t. Sometimes, the encoding could be simplified to vectors. For instance, in AutoFormer, the neural network is encoded as a vector in the form of [*Embed dim, MLP ratio, Head Num, Depth Num*], allowing representation as discrete vectors[6].

At this point, the NAS process can be described as a problem of maximizing/minimizing a specific objective function (i.e., neural architecture's performance indicator, such as test set accuracy) within the search space. In practice, evaluating a neural architecture requires training on a training set D_{train} and then testing on a test set D_{test}, often consuming a large amount of time. One-shot NAS partially mitigates this issue through methods like parameter sharing[31] and weight entanglement[6], but the evaluation process remains costly compared

with common optimization problems. Further, in real-world applications, accuracy is usually not the only objective to optimize and a trade-off among different conflicting objectives needs to be pursued. Particularly, when balancing multiple indicators (such as model complexity and error rate on test data), the NAS process becomes an expensive multi-objective optimization problem:

$$\min_{\alpha \in \mathbb{A}} \mathbf{F}(\alpha) = (f_1(\alpha), f_2(\alpha), \ldots, f_m(\alpha)) \tag{1}$$

where the symbol α denotes a neural architecture in the search space $\mathcal{A}$, and $\mathbf{F} : \mathcal{A} \to \mathbb{R}^m$ is an m-dimensional vector-valued objective function.

In practical multi-objective NAS problems, it is often impossible to find a single solution that simultaneously optimizes multiple indicators such as computation cost and accuracy. Therefore, the problem evolves into the intricate challenge of identifying the Pareto front (PF) across varying trade-offs. Previous works have simplified the problem into a constrained single-objective optimization problem[6]. Alternatively, the problem can be further transformed into an unconstrained single-objective optimization problem through scalarization, as seen in the works of Cai *et al.*[4] and Tan *et al.*[35]. Another prevalent approach is to search for a set of non-dominated solutions to approximate the PF, subsequently applying multi-objective optimization algorithms, such as the NSGA algorithm[26,27].

2.2 Pareto Set Learning (PSL)

In the field of expensive multi-objective optimization, earlier methods generally employ surrogate models that map the decision space to the objective space to find a collection of well-distributed non-dominated solutions. Unlike conventional approaches that seek to identify a representative set of such solutions, the PSL adopts an inverse approach. It utilizes a surrogate model from the objective space back to the decision space, aiming to predict a broad approximation of the decision variables that could generate Pareto optimal outcomes[23]. This process is facilitated by a *set model* that maps all valid trade-off preferences $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum_{i=1}^m \lambda_i = 1\}$ to their corresponding Pareto solutions, thus enabling the exploration of the entire PF by modifying the trade-off preferences.

Specifically, the PSL employs the so-called augmented Tchebycheff approach to scalarize the multi-objective optimization problem, and aims to find optimal parameters θ^* for the set model $\boldsymbol{x}^*(\boldsymbol{\lambda}) = h_{\boldsymbol{\theta}^*}(\boldsymbol{\lambda})$ which could generate a competitive solution set $\mathcal{M}_{psl} = \{\boldsymbol{x}^*(\boldsymbol{\lambda}) | \boldsymbol{\lambda} \in \Lambda\}$ for augmented Tchebycheff scalarization $g_{tch\_aug}(\boldsymbol{x}|\boldsymbol{\lambda})$, where

$$h_{\boldsymbol{\theta}^*}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{x} \in \mathcal{X}} g_{tch\_aug}(\boldsymbol{x}|\boldsymbol{\lambda}), \forall \boldsymbol{\lambda} \in \Lambda \tag{2}$$

The deployment of set models in PSL has not only made the traversal through the multi-dimensional space of preferences delineated by $\boldsymbol{\lambda}$ more straightforward but also more efficient. This efficiency gain is pivotal for enabling the utilization of hypervolume-based improvement strategies in batch selection, which is formalized as

$$\mathcal{B}_{hv} = \{\boldsymbol{x}^*(\boldsymbol{\lambda}) \mid \Delta HV(\boldsymbol{x}^*(\boldsymbol{\lambda}), \mathcal{M}_{psl}) > 0,\ \boldsymbol{\lambda} \in \Lambda\} \tag{3}$$

where $\mathcal{B}_{hv}$ denotes the set of batch-selected solutions, and $\Delta HV(\boldsymbol{x}^*(\boldsymbol{\lambda}), \mathcal{M}_{psl})$ quantifies the hypervolume improvement contributed by the new solution $\boldsymbol{x}^*(\boldsymbol{\lambda})$ over the current Pareto set $\mathcal{M}_{psl}$. This selection approach optimizes the trade-off exploration by quantitatively measuring the expansion of the PF, thereby providing a methodological advancement in the field of multi-objective optimization that is both effective and computationally viable.

3 Method

In this section, we delineate our methodological framework for PiMO-NAS. Initially, we expound upon the definitions and compositions of our search spaces, which are derived from AutoFormer[6] and OFA[3] search spaces. These spaces, characterized by their discrete nature, present limitations for the PSL technique. Subsequently, we introduce our novel framework designed to estimate the function landscape, employing a synergistic approach that integrates a surrogate model with a recurrent solution generator[10]. Lastly, we propose an adaptive sampling strategy tailored to form a more balanced PF.

3.1 Search Space

In order to validate the effectiveness of PiMO-NAS, while considering the inherent differences between CNNs and ViTs, we selected typical one-shot NAS spaces from OFA (for CNNs) and AutoFormer (for ViTs) as our exploration grounds.

OFA Search Space. As part of our methodology, we utilize the Once-For-All approach, which offers a comprehensive search across several dimensions crucial for optimizing CNNs. These dimensions include depth, input resolution, width, and kernel size[3]. For a detailed exploration of these parameters, refer to our online supplementary materials[28] , where we follow the configuration settings of MSuNAS [26] and detail our parameter choices and methodology.

AutoFormer Search Space. To optimize the ViT architecture, we employ an AutoFormer-based search methodology, which segments the architecture into critical dimensions such as Embedding Dimension, MLP Ratio, Head Number, and Depth[6]. Detailed parameter exploration for these dimensions is extensively described in our online supplementary materials[28].

Continuous Transformation of Search Space. Similar to many NAS problems, both OFA and AutoFormer employ discrete search spaces to encode neural architectures. In our approach, we map the domain of discrete encoding simply onto the range $[0, 1]$, thus transforming the problem into minimizing a cost function in a continuous real space regarding an n-dimensional decision vector. To restrict the ineffective space generated by zero-padding, we introduce a masking technique that will be further elaborated in Sect. 3.2.

3.2 Pareto-Informed Adaptive Search

Task Formulation. After implementing the aforementioned encoding and transformation of search space, we relaxed the decision space for both OFA-based and AutoFormer-based search spaces into a continuous domain. Consequently, the multi-objective NAS problem considered in this paper is transformed into a multi-objective optimization problem on a continuous space, and can be formalized as the following parametric bi-objective optimization problem:

$$\min_{\mathbf{x}\in\mathbb{X}} \mathbf{F}(\mathbf{x}) = (Error(\mathbf{x}), Params(\mathbf{x})) \tag{4}$$

where $\mathbf{F}$ denotes the objective vector consisting of two objective functions (i.e., $m = 2$), $\mathbf{x}$ represents the decision vector embodied as $\mathbf{x} = (x_1, x_2, \ldots, x_n), x_i \in [0, 1]$, and $\mathbb{X} \subseteq \mathbb{R}^n$ is the search space (also known as decision space).

Building on this bi-objective optimization problem, we developed PiMO-NAS for searching its PF in an iterative optimization manner. In a nutshell, each iteration of our proposed PiMO-NAS divides the fixed-length continuous neural architecture search process into three critical phases, which will be elaborated in the followings.

Surrogate-Model-Based PF Estimation. In the first phase of our approach, it is crucial to estimate the overall PF, which necessitates the use of a surrogate model. Drawing inspiration from Pareto set learning[23] and referencing the analysis in MSuNAS[26], we opt for Gaussian Process (GP) as our surrogate, and we independently run a GP model for each objective function.

At the onset of our algorithm, we employ Latin Hypercube Sampling (LHS) to select a set of candidate solutions from the search space. Besides, our initial population also includes both all-ones and all-zeros decision vectors to estimate the functional boundaries. These solutions are then evaluated on the ImageNet[11] dataset to generates data for the training of our GP surrogate model.

Once the evaluations are completed, we normalize the results to align with the GPs. The GPs are then executed to perform regression analysis on this data, and the regression's outcome is an estimate of the overall function landscape.

Exploration via Solution Generator. In the second phase, we aim to thoroughly explore the landscape of the estimated objective function space. To achieve this, we utilize a solution generator, which is designed to operate under the guidance of a preference vector (denoted as $\boldsymbol{\lambda}$). The solution generator's primary task is to produce optimal solutions for single-objective minimization problems, as determined by the augmented Tchebycheff scalarization approach as Eq. (5), guided by the given preference vector $\boldsymbol{\lambda}$ in $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum_{i=1}^m \lambda_i = 1\}$.

$$\hat{g}_{\text{tch\_aug}}(\mathbf{x}|\boldsymbol{\lambda}) = \max_{1\le i\le m} \left\{ \lambda_i \left| \hat{f}_i(\mathbf{x}) - (z_i^* - \epsilon) \right| \right\} + \rho \sum_{i=1}^{m} \lambda_i \hat{f}_i(\mathbf{x}) \tag{5}$$

where $\rho > 0$ and $\epsilon > 0$ are small positive scalars used in [22], $z^* = (z_1^*, \ldots, z_m^*)$ is the ideal vector, and $\hat{f}_i(\mathbf{x})$ is the lower confidence bound (LCB) for the i-th objective function provided by the corresponding GP surrogate model.

Given the block-wise stacked architecture inherent in both OFA-based and AutoFormer-based search spaces, we drew inspiration from works like [31,35] to devise our solution generator. This generator is based on a Gated Recurrent Unit (GRU) architecture[10] as shown in Fig. 2.

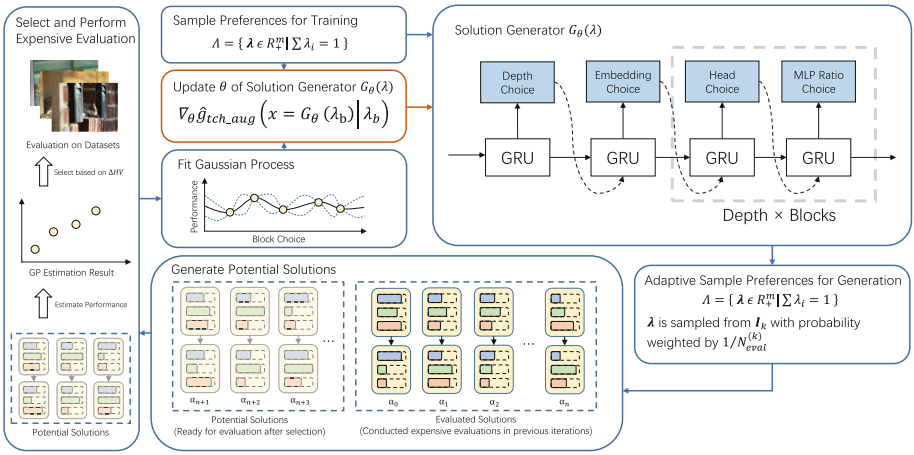


Fig. 2. Illustration for each iteration of our proposed PiMO-NAS. (1) GP surrogate model is utilized for estimating the function landscape from evaluated solutions. Note that we independently run a GP model for each objective function. (2) Solution Generator is responsible for generating the expected optimal solutions based on arbitrary target function preferences $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum_{i=1}^m \lambda_i = 1\}$.

The solution generator $G_\theta(\cdot)$ initiates its operation by mapping a preference vector $\boldsymbol{\lambda}$ to an embedding using a linear layer. This embedding serves as the initial hidden state h^0 for the GRU. Subsequently, various linear heads, tailored to the specific stages of the model architecture, generate different types of decision variables. These heads operate in distinct stages, ensuring that the variable generation is aligned with the structural of the neural network blocks.

Building upon the foundation laid by the GP approximation of the function landscape, our solution generator $G_\theta(\cdot)$, where θ denotes the parameters of $G_\theta(\cdot)$, embarks on a journey of exploration, guided by a set of preference vectors $\{\boldsymbol{\lambda}_b\}_{b=1}^B$ obtained via random sampling. The solution generator's purpose is to probe the approximated terrain, seeking to enhance the lower bounds of performance for the solutions it generates through gradient descent:

$$\theta \leftarrow \theta - \eta \sum_{b=1}^{B} \nabla_\theta \hat{g}_{\text{tch\_aug}}(\mathbf{x} = G_\theta(\boldsymbol{\lambda}_b) | \boldsymbol{\lambda}_b) \tag{6}$$

In addition, it is noteworthy that the nature of our search space involves models with variable depth, while our encoding scheme employs fixed-length vectors. To address this discrepancy, we introduce a masking technique for the solutions generated beyond their designated depth limits. Specifically, a mask is applied to map the excess portions of the solution to zero. This approach ensures consistency with the zero-padding scheme previously established for managing the dimensionality of decision vectors in our search space.

Batch Selection with Adaptive Sampler. In the final phase of each iteration, we focus on the generation and evaluation of candidate solutions derived from an extensive exploration of the function landscape. We design an adaptive sampler to dynamically generate preference vector as exploration direction:

$$\boldsymbol{\lambda} = (w, 1-w), w \in [0,1] \tag{7}$$

where w is the weight of the first objective, and we feed $\boldsymbol{\lambda}$ into the solution generator. The sampler achieves this by dividing the sampling domain of weight w into K equal intervals and tracking the outcomes of candidate solutions generated from each interval. It maintains a record of the number of solutions from each interval that are accepted for costly evaluation $\mathbf{N}_{\text{eval}} = (N_{\text{eval}}^{(1)}, N_{\text{eval}}^{(2)}, \ldots)$, and the sampling of preference is converted into a two-stage process:

$$P(I = I_k) = \frac{1/N_{\text{eval}}^{(k)}}{\sum_{k=1}^{K} 1/N_{\text{eval}}^{(k)}}, \quad \text{for } k = 1, 2, \ldots, K \tag{8}$$

$$w \sim U(l^{(k)}, u^{(k)}) \tag{9}$$

where $N_{\text{eval}}^{(k)}$ represents the number of solutions generated from $w \in I_k$, and K is the total number of distinct intervals $I_k = [l^{(k)}, u^{(k)}] = [(k-1)/K, k/K]$. This discretization allows for a more controlled exploration of the solution space.

Once the sampling process is completed, the preference vectors are input into the solution generator to obtain candidate solutions for the subproblems. These candidate solutions are then subjected to a batch selection process that filters the most promising subset to advance the search process. The batch selection procedure is guided by the expected hypervolume improvement within the framework of GP:

$$\Delta HV(\hat{\mathbf{F}}(X_{\text{cand}})) = HV(\mathbf{y}_{t-1} \cup \hat{\mathbf{F}}(X_{\text{cand}})) - HV(\mathbf{y}_{t-1}) \tag{10}$$

where X_{cand} are the candidate solutions generated by the solution generator from a batch of weight vectors. The term $\mathbf{y}_{t-1}$ represents the performance indicators of the solution set that has been evaluated in previous iterations, and $\hat{\mathbf{F}}(\cdot)$ denotes the estimated performance indicators provided by the GP models.

Summary of PiMO-NAS The complete framework of our PiMO-NAS algorithm is encapsulated and presented as Algorithm 1. In our comprehensive framework, we have incorporated the GP estimation, landscape exploration, and solution generation processes as mentioned above. Notably, unlike typical continuous optimization problems, the inherent discreteness of the original space in our problem leads to many slightly different solutions in the relaxed space being identical when evaluated by the function. To circumvent this, we applied a deduplication step when selecting solutions for evaluation, thereby avoiding redundant and costly evaluation procedures.

Algorithm 1: PiMO-NAS Framework

Input: $\mathbf{F}(\mathbf{x})$ as Eq. (4); Exploration Iter: T; Training Batch Size: B; Training Iter: N_{iter}; Interval Seperation Num: K; Number of Solutions Generated per Iteration: P;

Output: $\{\mathbf{x}_T, \mathbf{y}_T\}$ and Solution Generator $G_{\theta_T}(\cdot)$

Randomly sample initial population $\{\mathbf{x}_0, \mathbf{y}_0 = \mathbf{F}(\mathbf{x}_0)\}$;
for $t = 1$ *to* T **do**
  Train GPs based on $\{\mathbf{x}_{t-1}, \mathbf{y}_{t-1}\}$;
  Initialize θ_t of solution generator G_{θ_t};
  for $i = 1$ *to* N_{iter} **do**
    Sample preferences $\{\boldsymbol{\lambda}_b\}_{b=1}^{B} \sim \Lambda$;
    Update θ_t using gradient descent as in Eq. (6);
  end
  Sample P intervals with replacement denoted as $\{I^{(p)}\}_{p=1}^{P}$ via the adaptive sampler in Eq. (8);
  Sample P preference vectors $\{\boldsymbol{\lambda}^{(p)} = [w^{(p)}, 1 - w^{(p)}], w^{(p)} \in I^{(p)}\}_{p=1}^{P}$;
  Generate solutions $X_p = \{G_{\theta_t}(\boldsymbol{\lambda}^{(p)})\}_{p=1}^{P}$;
  Find non-repetitive subset X_{cand} with highest ΔHV from X_p;
  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} \cup X_{cand}$;
  $\mathbf{y}_t \leftarrow \mathbf{y}_{t-1} \cup \mathbf{F}(X_{cand})$;
end

4 Experiments

In this section, we introduce the experimental framework utilized to evaluate the proposed PiMO-NAS method. The central aim of our experiments is to demonstrate the effectiveness of PiMO-NAS by applying it within two specific one-shot NAS spaces: the AutoFormer space [6] and an OFA-based space shared with MSuNAS[26].

4.1 OFA-Based Search Space

We applied PiMO-NAS to the same NAS space as MSuNAS, starting with a population of 100. In each iteration, a solution generator with a hidden layer of 128

dimensions was used. Throughout the training, we randomly sampled 20 preferences per iteration, over 2000 iterations, until convergence. Using the Adaptive Sampler, 15 preferences were sampled to generate candidate solutions. The final selection involved screening 8 solutions based on expected hypervolume improvement for evaluation on the ImageNet dataset. The exploration process, spanning 30 epochs, revealed comparative insights between PiMO-NAS and MSuNAS. It is important to highlight that in our implementation within MSuNAS, GPs were utilized as surrogate models. We emphasize that the choice of surrogate model type used in MSuNAS is kept consistent with our framework.

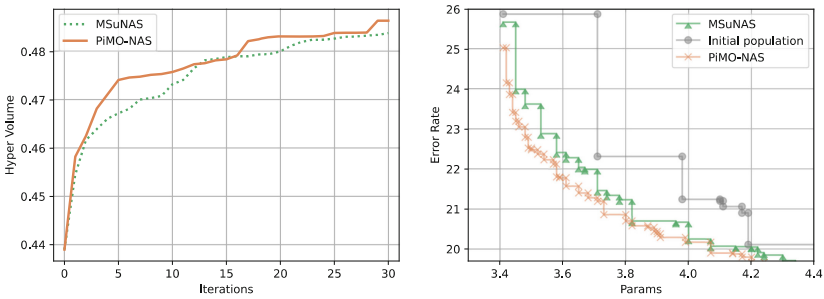


Fig. 3. Left: Hypervolume curves. The reference point was set at [8.426, 0.284], representing the worst values for each objective in the initial population. **Right:** Solutions obtained at the end of the search.

As shown in Fig. 3, an early observation was that PiMO-NAS exhibited a faster rate of hypervolume increase compared to MSuNAS. This indicates PiMO-NAS's efficient exploration and utilization of the surrogate model's estimated space. Specifically, after approximately 20 iterations (corresponding to 160 samples), the hypervolume achieved by PiMO-NAS approaches the results obtained by MSuNAS after 30 iterations. Furthermore, the Pareto solution set associated with PiMO-NAS not only exhibits a significantly larger hypervolume but also a denser distribution compared to that of MSuNAS at 30 iterations. Therefore, even when accounting for the overhead of training the solution generator, PiMO-NAS achieves a 1.24x speedup compared to the MSuNAS method for equivalent hypervolume results. Additionally, as the cost per solution evaluation increases, the relative expenditure of the Solution Generator in the total search time will further diminish. This reduction is expected to further extend the performance gap between PiMO-NAS and MSuNAS.

We also analyzed a total of 240 candidate solutions sampled by PiMO-NAS and MSuNAS as Fig. 4. We found that, compared with the nearly normal distribution sampling of different parameter-volume architectures by MSuNAS in the search space, PiMO-NAS, guided by GP evaluations, can sample more points with lower parameter volumes, effectively enhancing the dominance of the obtained solutions, approaching the true Pareto front. After introducing the

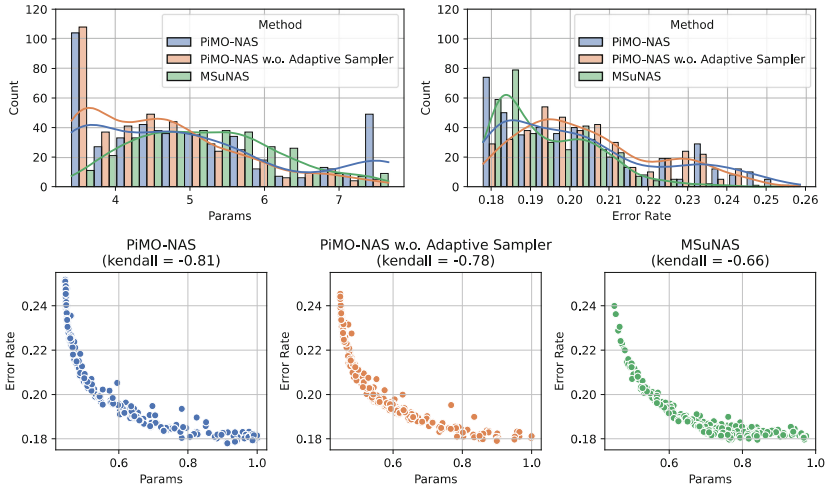


Fig. 4. Comparison of the search results of PiMO-NAS and MSuNAS. **Upper:** Distribution of Params and Error Rate of candidates sampled by PiMO-NAS and MSuNAS. **Lower:** The Kendall's Tau of points sampled by PiMO-NAS and MSuNAS.

adaptive sampler, PiMO-NAS can obtain a smoother sampling curve and significantly increase the sampling of large-volume models, compensating for the shortcoming in sampling density in the high parameter-volume region to some extent. Notably, we calculated the Kendall's rank correlation coefficient of the sampled solutions and found that the samples obtained by the PiMO-NAS method more purposefully explore trade-off solutions.

4.2 AutoFormer-Based Search Space

We further explored the AutoFormer-based search space using the PiMO-NAS method, while employing the genetic algorithm based AutoFormer with a predefined parameter range from the original paper of AutoFormer as a strong baseline. Following the genetic algorithm's design in the original study, we used a population of 50 and conducted 20 iterations. For PiMO-NAS, we started with a sample size of 200, sampling 20 instances per iteration over 40 rounds, a design balancing sampling efficiency and the cost of GP. Both our approach and the AutoFormer's genetic algorithm sampled 1000 instances, with the results depicted in Fig. 5.

In our observations, after 1000 samples, the PiMO-NAS method effectively covered the search results of the genetic algorithm within the predefined parameter range. Notably, in both Supernet-Small and Supernet-Base spaces, PiMO-NAS managed to cover the optimal solutions obtained by three rounds of the genetic algorithm within its 1000 samples, while AutoFormer itself requires 3,000 samples in total across these iterations.

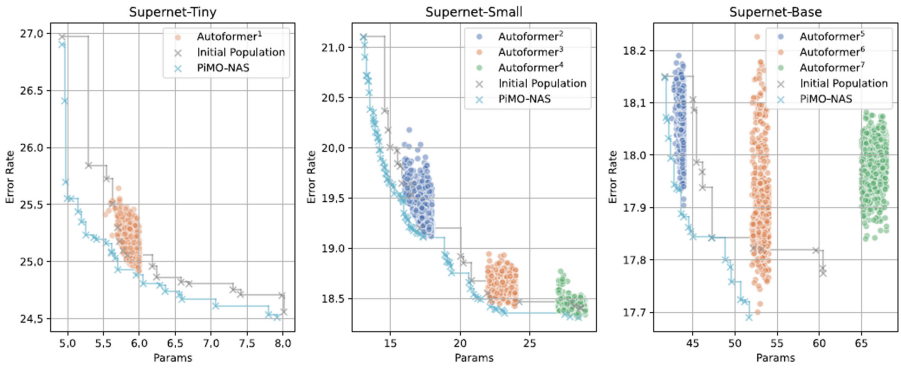


Fig. 5. Solutions obtained by PiMO-NAS on the AutoFormer-based search space. The variants of AutoFormer with distinct superscripts (i.e., 1-7) indicate the search results obtained under different parameter range restrictions, as detailed in supplementary materials [28] In Supernet-Small space, the PF obtained by PiMO-NAS (with only 1,000 samples) can efficiently cover the results obtained by AutoFormer^2+AutoFormer^3+AutoFormer^4 (each of which consumes 1,000 samples, thus producing a combined total of 3,000 samples). A similar phenomenon also occurs in Supernet-Base space.

4.3 Performance Comparisons of PiMO-NAS *versus* Existing SOTA Methods/Models

Finally, we conducted 20 iterations on the same space as MSuNAS with 100 random samples. In each iteration, we generated and evaluated eight candidate solutions. After completing the search, we fine-tuned the obtained solutions on the ImageNet dataset for 200 epochs. These solutions were named PiMO-NAS-O-T/S/B where 'O' denotes the OFA-based search space. The solutions were then ranked according to their FLOPs. Table 1 displays the performance of our models on the ImageNet dataset. The networks discovered by PiMO-NAS demonstrated strong competitiveness with the MSuNAS method, while requiring significantly less searching time.

For the results on the AutoFormer-based search space as presented in Table 2, we named the search results PiMO-NAS-A-T/S/B, where 'A' denotes the AutoFormer-based search space, categorized according to the search space divisions. We successfully achieved results comparable to those obtained in the original AutoFormer paper, which uses a genetic algorithm with a restricted parameter range. This corroborates that PiMO-NAS has effectively approximated the Pareto frontier in the AutoFormer space after 1,000 samples .It is important to note that while PiMO-NAS utilized slightly more searching time, it achieved dozens to hundreds of Pareto solutions in one round of search on the AutoFormer search space, as opposed to the original algorithm where each solution is obtained with one round of search involving 1,000 samplings. This implies that when multiple deployments are needed, PiMO-NAS won't exhibit

Table 1. Comparison results on OFA-based search space for the ImageNet classification task. The time required for supernet training and fine-tuning is not included.

Model	Type	Search Cost (GPU days)	Top-1 Acc. (%)	#Params (M)	Flops (M)
MobileNetV3[19]	combined	-	75.2	5.4	219
ShuffleNetV2[29]	manual		72.6	-	299
RCNet [46]	auto	9	74.7	4.7	471
SPOSNAS [16]	auto	13	74.8	5.3	465
MnasNet-A1 [35]	auto	3800	75.2	3.9	312
EEEA-Net-C1 [36]	auto	0.52	74.3	5.1	137
MOEA-PS [41]	auto	5.2	73.6	4.7	-
PiMO-NAS-O-T(Ours)	**auto**	**0.81**	**75.2**	**5.6**	**153**
FairNAS-A [9]	auto	12	77.5	5.9	392
MixPath-B [8]	auto	10.3	77.2	5.1	378
NSGANetV2-m [26]	auto	1	78.3	7.7	312
PiMO-NAS-O-S(Ours)	**auto**	**0.81**	**77.7**	**6.1**	**362**
FP-DARTS [39]	auto	2.44	76.3	7.2	598
PNAG-170 [15]	auto	0.7	80.3	10	606
NSGANetV2-xl [26]	auto	1	80.4	8.7	593
PiMO-NAS-O-B(Ours)	**auto**	**0.81**	**80.2**	**7.44**	**576**

Table 2. Comparison results on AutoFormer-based search space for the ImageNet classification task. The time required for supernet training and fine-tuning is not included. † indicates the results we reproduce using open-source code provided by the authors of AutoFormer.

Model	Type	Search Cost (GPU days)	Top-1 Acc. (%)	#Params (M)	Flops (G)
ViT-Ti[12]	manual	-	74.5	5.7	1.4
PVT-Tiny [38]	manual	-	75.1	13.2	1.9
DeiT-Tiny [37]	manual	-	72.2	5.7	1.2
ViTAS-C [33]	auto	32	74.7	5.6	1.3
AutoFormer-Tiny†[6]	auto	1.83	75.1	6	1.4
PiMO-NAS-A-T(Ours)	**auto**	**2.16**	**75.1**	**5.7**	**1.6**
T2T-VIT-14 [43]	manual	-	81.7	21.5	6.1
SWIN-T [25]	manual	-	**82.3**	29	4.5
GLiT-Small [5]	manual	-	80.5	24.6	4.4
CrossFormer-T [40]	manual	-	81.5	27.7	2.9
PoolFormer-S24 [42]	manual	-	80.3	21.3	3.4
TF-TAS-S [45]	auto	0.5	80.5	24.6	3.2
As-ViT-S [7]	auto	0.5	81.2	29	5.3
ViTAS-F [33]	auto	32	80.5	27.6	6.0
AutoFormer-Small†[6]	auto	1.83	81.6	23.9	4.8
PiMO-NAS-A-S(Ours)	**auto**	**2.16**	81.6	**21.4**	**4.3**
T2T-VIT-19 [43]	manual	-	82.2	39.2	9.8
EAT-B [44]	manual	-	82	86.6	14.8
ConViT-S+ [13]	manual	-	82.2	48	10.0
GLiT-Base [5]	manual	-	82.3	96.1	17.0
PoolFormer-M48 [42]	manual	-	**82.5**	73.4	11.6
AutoFormer-Base†[6]	auto	1.83	82.3	52.7	10.3
PiMO-NAS-A-B(Ours)	**auto**	**2.16**	82.3	**51.7**	**10.8**

a significant time disadvantage compared to zero-shot NAS methods like TF-TAS[45].

5 Conclusion

This work introduced the Pareto-Informed Multi-Objective Neural Architecture Search (PiMO-NAS) method, offering a significant stride in the domain of multi-objective NAS. By integrating the technique of Pareto set learning with a one-shot NAS framework, our approach adeptly balances multiple conflicting objectives, such as accuracy and model size, within neural architecture search. Our experiments within the OFA-based and AutoFormer-based search spaces showcased that PiMO-NAS could efficiently cover the search results of genetic algorithms, exemplifying its efficacy in discovering Pareto-optimal solutions.

PiMO-NAS was observed to outperform baselines in terms of hypervolume metric, indicating its proficiency in exploring the search space. Moreover, the adaptive sampler introduced in our methodology promoted diversity and circumvented local optima, further solidifying the robustness of PiMO-NAS.

Ultimately, the models discovered by PiMO-NAS achieved competitive accuracy on the ImageNet dataset, underlining the practicality of our method in real-world scenarios. The innovative blend of techniques within PiMO-NAS paves the way for future research in the efficient and effective exploration of complex neural architecture landscapes. Our findings and the developed method contribute a valuable addition to the field of NAS, promising to streamline the design of high-performing neural networks for diverse computational budgets.

This work has certain limitations, similar to those observed in studies like MSuNAS, wherein the precision of GP-based landscape estimation of objective functions may vary across distinct datasets, potentially impacting the quality of solution generation in PiMO-NAS. Additionally, the complexity of GP regression is tied to the number of samples, typically exhibiting a computational complexity of $O(n^3)$. This results in progressively longer iteration times for PiMO-NAS. In future research, we plan to optimize the cost of the Gaussian Process in this context and explore the potential of hybrid or adaptive surrogate models within the PiMO-NAS framework.

Disclosure of Interest. The author declares no competing interests as defined by Springer Nature, or other interests that might be perceived to influence the results and/or discussion reported in this manuscript.

References

1. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning (2016). arXiv preprint arXiv:1611.02167
2. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Smash: one-shot model architecture search through hypernetworks (2017). arXiv preprint arXiv:1708.05344
3. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-all: Train one network and specialize it for efficient deployment (2019). arXiv preprint arXiv:1908.09791

4. Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware (2018). arXiv preprint arXiv:1812.00332
5. Chen, B., et al.: Glit: neural architecture search for global and local image transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 12–21 (2021)
6. Chen, M., Peng, H., Fu, J., Ling, H.: Autoformer: searching transformers for visual recognition. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 12270–12280 (2021)
7. Chen, W., Huang, W., Du, X., Song, X., Wang, Z., Zhou, D.: Auto-scaling vision transformers without training (2022). arXiv preprint arXiv:2202.11921
8. Chu, X., Lu, S., Li, X., Zhang, B.: Mixpath: a unified approach for one-shot neural architecture search. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5972–5981 (2023)
9. Chu, X., Zhang, B., Xu, R.: Fairnas: rethinking evaluation fairness of weight sharing neural architecture search. In: Proceedings of the IEEE/CVF International Conference on computer vision, pp. 12239–12248 (2021)
10. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). arXiv preprint arXiv:1412.3555
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. IEEE (2009)
12. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale (2021)
13. d'Ascoli, S., et al.: Improving vision transformers with soft convolutional inductive biases. In: International conference on machine learning, pp. 2286–2296. PMLR (2021)
14. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
15. Guo, Y., et al.: Pareto-aware neural architecture generation for diverse computational budgets. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2247–2257 (2023)
16. Guo, Z., et al.: Single path one-shot neural architecture search with uniform sampling. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16, pp. 544–560. Springer (2020)
17. Han, K., et al.: A survey on vision transformer. IEEE Trans. Pattern Anal. Mach. Intell. **45**(1), 87–110 (2022)
18. 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)
19. Howard, A., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 1314–1324 (2019)
20. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700–4708 (2017)
21. Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., Xing, E.P.: Neural architecture search with Bayesian optimisation and optimal transport. Adv. Neural Inf. Process. Syst. **31** (2018)
22. Knowles, J.: ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evol. Comput. **10**(1), 50–66 (2006)
23. Lin, X., Yang, Z., Zhang, X., Zhang, Q.: Pareto set learning for expensive multiobjective optimization. Adv. Neural. Inf. Process. Syst. **35**, 19231–19247 (2022)

24. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search (2018). arXiv preprint arXiv:1806.09055
25. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 10012–10022 (2021)
26. Lu, Z., Deb, K., Goodman, E., Banzhaf, W., Boddeti, V.N.: Nsganetv2: evolutionary multi-objective surrogate-assisted neural architecture search. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16, pp. 35–51. Springer (2020)
27. Lu, Z., et al.: Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference, pp. 419–427 (2019)
28. Luo, G., Li, H., Chen, Z., Zhou, Y.: Supplementary materials of "pareto-informed multi-objective neural architecture search" (2024). https://github.com/SYSU22214881/PiMO-NAS
29. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV), pp. 116–131 (2018)
30. Peng, Y., Song, A., Ciesielski, V., Fayek, H.M., Chang, X.: PRE-NAS: predictor-assisted evolutionary neural architecture search. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1066–1074 (2022)
31. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: International conference on machine learning, pp. 4095–4104. PMLR (2018)
32. Real, E., et al.: Large-scale evolution of image classifiers. In: International conference on machine learning, pp. 2902–2911. PMLR (2017)
33. Su, X., et al.: Vitas: Vision transformer architecture search (2021). arXiv e-prints pp. arXiv–2106
34. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826 (2016)
35. Tan, M., et al.: Mnasnet: platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2820–2828 (2019)
36. Termritthikun, C., Jamtsho, Y., Ieamsaard, J., Muneesawang, P., Lee, I.: EEEA-Net: an early exit evolutionary neural architecture search. Eng. Appl. Artif. Intell. **104**, 104397 (2021)
37. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning, pp. 10347–10357. PMLR (2021)
38. Wang, W., et al.: Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 568–578 (2021)
39. Wang, W., Zhang, X., Cui, H., Yin, H., Zhang, Y.: FP-DARTS: fast parallel differentiable neural architecture search for image classification. Pattern Recogn. **136**, 109193 (2023)
40. Wang, W., et al.: Crossformer: a versatile vision transformer hinging on cross-scale attention. In: International Conference on Learning Representations (2021)
41. Xue, Y., Chen, C., Słowik, A.: Neural architecture search based on a multi-objective evolutionary algorithm with probability stack. IEEE Trans. Evol. Comput. **27**(4), 778–786 (2023)

42. Yu, W., et al.: Metaformer is actually what you need for vision. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10819–10829 (2022)
43. Yuan, L., et al.: Tokens-to-token vit: training vision transformers from scratch on imagenet. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 558–567 (2021)
44. Zhang, J., et al.: Analogous to evolutionary algorithm: designing a unified sequence model. Adv. Neural. Inf. Process. Syst. **34**, 26674–26688 (2021)
45. Zhou, Q., et al.: Training-free transformer architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10894–10903 (2022)
46. Zong, Z., Cao, Q., Leng, B.: RCNet: reverse feature pyramid and cross-scale shift network for object detection. In: Proceedings of the 29th ACM International Conference on Multimedia, pp. 5637–5645 (2021)

A Variable-Length Fuzzy Set Representation for Learning Fuzzy-Classifier Systems

Hiroki Shiraishi[1(✉)], Rongguang Ye[2], Hisao Ishibuchi[2], and Masaya Nakata[1]

[1] Department of Electrical Engineering and Computer Science, Yokohama National University, Yokohama 240-8501, Japan
shiraishi-hiroki-yw@ynu.jp, nakata-masaya-tb@ynu.ac.jp
[2] Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
yerg2023@mail.sustech.edu.cn, hisao@sustech.edu.cn

Abstract. This paper introduces a novel Learning Fuzzy-Classifier System (LFCS) that incorporates variable-length fuzzy sets in rule antecedents to enhance classification accuracy and mitigate overfitting in real-world data scenarios. Traditional LFCSs utilize fixed-length fuzzy sets, which can limit their performance, especially when the rule set size is restricted in high-dimensional input space. The proposed algorithm, Fuzzy-UCSv (i.e., the Fuzzy-UCS classifier system with a variable-length fuzzy set representation), addresses these limitations by allowing the number of fuzzy sets per dimension in rule-antecedents to vary. Fuzzy-UCSv aims to tackle two primary challenges identified in LFCS: the unnecessary optimization of membership functions for irrelevant features and the difficulty in forming optimal classification boundaries with a single membership function per feature. By optimizing the number of membership functions for each rule using an evolutionary algorithm, Fuzzy-UCSv acquires rules that ignore non-contributing features and effectively cover complex input spaces, significantly improving test accuracy without increasing the risk of overfitting. Experimental results demonstrate that Fuzzy-UCSv outperforms conventional Fuzzy-UCS and other machine learning techniques in terms of test accuracy.

Keywords: Learning Fuzzy-Classifier Systems · Supervised Learning · Membership Functions · Variable-Length Representation

1 Introduction

Learning Fuzzy-Classifier Systems (LFCSs) [40] represent a paradigm in evolutionary machine learning that employs an evolutionary algorithm to search for an accurate and general set of fuzzy rules [11]. An LFCS works as a classifier that adaptively divides the input space into multiple subspaces using fuzzy sets in the rule-antecedents, with individual fuzzy rules making local classification decisions within each subspace. This divide-and-conquer approach is particularly effective

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 386–402, 2024.
https://doi.org/10.1007/978-3-031-70071-2_24

for tasks with large and complex input spaces, such as those encountered in real-world data mining [14,31,35].

In LFCSs, setting appropriate membership functions in the rule-antecedents is crucial for effectively dividing the input space. Increasing the number of parameters and complicating the shape of membership functions allows for the production of more flexible classification boundaries. However, this flexibility enhancement increases the risk of overfitting to the training data and typically requires more time for parameter optimization [22,31]. Consequently, simple membership functions such as interval-based [36], triangular [27], and Gaussian [37] are commonly used. This paper uses the triangular membership function, which consists of three real-valued parameters (left, center, and right vertices [27]).

For a d-dimensional classification task, each input in the antecedent part of a fuzzy rule generated by LFCS has one fuzzy set (i.e., a triangular membership function). Accordingly, the rule-antecedent of a rule is always represented by a fixed-length real-valued string of length $3d$. Two problems can be identified in this regard:

Problem 1. Because an LFCS requires membership functions to be specified even for features that do not contribute to classification, it is necessary to optimize the parameters of the membership functions for unnecessary features as well. This can degrade the performance of LFCS, especially for high-dimensional tasks.

Problem 2. A rule can cover only one subspace. Therefore, if the training data points for a class are distributed across multiple subspaces, at least as many rules as there are subspaces must be optimized. This limitation can degrade the performance of LFCS, especially in scenarios where the rule set size is restricted (i.e., where no more rules than the number of subspaces can be allocated).

To address these problems, this paper proposes an LFCS using rule antecedents with variable-length fuzzy sets for the first time and aims to verify its effectiveness. Specifically, the number of membership functions assigned to each dimension i of the rule-antecedent of fuzzy rule k is extended from the conventional single function to $m_i^k \in \mathbb{N}_0$ functions. An evolutionary algorithm optimizes the number of membership functions for each rule. The system aims to acquire rules that do not place membership functions for non-contributing dimensions i ($m_i^k = 0$, i.e., equivalent to *Don't Care*) and place multiple membership functions for difficult-to-divide dimensions i ($m_i^k > 1$), thus addressing the problems 1 and 2 simultaneously. This paper extends the *sUpervised Fuzzy-Classifier System* (Fuzzy-UCS) [28], designed for single-label classification tasks, to the case of the variable-length fuzzy sets in rule-antecedents. We call the extended system Fuzzy-UCSv. It is important to note that this approach can also be applied to other LFCS variants, such as those for function approximation tasks [9] and multi-label classification tasks [26].

The paper is organized as follows: Sect. 2 reviews relevant literature. Section 3 outlines Fuzzy-UCS. Section 4 details our proposed algorithm. Section 5 presents comparative experiments using real-world classification tasks. The results of the

experiments are then evaluated and discussed. Finally, Sect. 6 concludes the paper.

2 Related Work

Bacardit et al. [3] proposed a rule representation to improve scalability in high-dimensional real-valued input tasks within BioHEL [4], a variant of *Learning Classifier Systems* (LCSs) [15] that uses non-fuzzy (i.e., crisp) rules. They introduced an *attribute list* in each rule, which holds the indices of dimensions important for classification. By optimizing only the parameters of the interval-shaped membership functions corresponding to these indices, BioHEL achieved improved learning efficiency and classification accuracy. Later, Urbanowicz et al. [39] extended this rule representation to handle classification tasks involving both discrete and real values.

Arif et al. [2] introduced a rule representation in an LCS known as XCSR [41], where two real-valued parameters that define the interval-shaped membership function (the lower limit l_i and the upper limit u_i) are set to the lower and upper ends of the input space domain, respectively. This approach creates a pseudo *Don't Care* condition. The system, termed XCSR#, exhibited improved performance in high-dimensional text data classification tasks with sparse data compared to traditional XCSR. The performance of XCSR#, based on *learning optimality theory* [23], was evaluated by Nakata and Browne [23] and succeeded in completely solving a 37-dimensional real-valued multiplexer problem [41] for the first time.

Arif et al. [1] proposed XCSRCFC, a method that replaces the interval-based representation in XCSR with a tree-like structure based on genetic programming called *code fragments*. These code fragments in XCSRCFC are binary tree structures with a maximum depth of two, where each terminal node is assigned an interval $[l_i, u_i]$ and the function set is {NOT, OR, NOR, AND, NAND}. Consequently, XCSRCFC, similar to the rule representations by Bacardit and Urbanowicz, can avoid encoding unnecessary dimensions as intervals. Experiments demonstrated that XCSRCFC outperformed XCSR in convergence performance on high-dimensional text data classification tasks. However, due to the complexity of the code fragment structure, it required more learning iterations than XCSR to converge.

In summary, previous research has addressed only Problem 1 mentioned in Sect. 1 and has not resolved Problem 2. Furthermore, all the aforementioned studies focused on LCS with interval-shaped non-fuzzy rules; to our knowledge, no studies have explored LFCS with fuzzy rules.

3 Fuzzy-UCS

Fuzzy-UCS [28] is a LFCS that integrates supervised learning with a steady-state genetic algorithm (GA) [13] to evolve fuzzy rules online. This system alternates between two phases: the training phase and the test phase. Within the training

phase, Fuzzy-UCS searches for accurate and maximally general rules. Conversely, during the test phase, the system applies these acquired rules to infer a class for a new unlabeled data point. This section briefly explains Fuzzy-UCS for a d-dimensional n-class single-label classification task with a class label set $\mathcal{C} = \{c_i\}_{i=1}^{n}$. For detailed explanations, kindly refer to [27,28].

3.1 Knowledge Representation

A d-dimensional fuzzy rule k is expressed by:

$$\textbf{IF } x_1 \text{ is } A_1^k \text{ and } \cdots \text{ and } x_d \text{ is } A_d^k \textbf{ THEN } c^k \textbf{ WITH } w^k, \tag{1}$$

where $\mathbf{A}^k = (A_1^k, ..., A_d^k)$ is an *rule-antecedent vector*, $c^k \in \mathcal{C} = \{c_1, ..., c_n\}$ is a *rule-consequent class*, and $w^k \in [0, 1]$ is a *rule-weight*. Each variable x_i is conditioned by a fuzzy set A_i for $\forall i \in \{1, ..., d\}$. A fuzzy set A_i^k is defined as $A_i^k = (a_i^k, b_i^k, c_i^k)$ by vertices of a triangle $a_i^k, b_i^k, c_i^k \in \mathbb{R}$ $(a_i^k \leq b_i^k \leq c_i^k)$.

The *membership degree* $\mu_{\mathbf{A}^k}(\mathbf{x}) \in [0, 1]$ of an input vector $\mathbf{x} \in [0, 1]^d$ with the rule k is computed using $\mu_{\mathbf{A}^k}(\mathbf{x}) = \prod_{i=1}^{d} \mu_{A_i^k}(x_i)$, where $\mu_{A_i^k} : [0, 1] \to [0, 1]$ is the membership function of the fuzzy set A_i^k. If the value of x_i is unknown, the system handles the missing value by specifying $\mu_{A_i^k}(x_i) = 1$.

Each rule k has five primary parameters: (i) a *fitness* $F^k \in (-1, 1]$, reflecting the classification accuracy of rule k; (ii) a *weight vector* $\mathbf{v}^k = (v_i^k)_{i=1}^{n} \in [0, 1]^n$, indicating the confidence with which rule k predicts each class c_i for a matched input (the largest element of $\mathbf{v}^k$ is used as w^k in Eq. (1)); (iii) a *correct matching vector* $\mathbf{cm}^k = (\mathrm{cm}_i^k)_{i=1}^{n} \in (\mathbb{R}_0^+)^n$, where each element cm_i^k is the sum of the matching degrees for training data points from class c_i; (iv) an *experience* $\mathrm{exp}^k \in \mathbb{R}_0^+$, calculating the accumulated contribution of rule k in classifying training data points; and (v) a *numerosity* $\mathrm{num}^k \in \mathbb{N}_0$, indicating the number of copies of rule k present in the population, which is a necessary parameter because the GA may produce rules with duplicate rule-antecedent and consequent pairs. These parameters are continuously updated throughout the training phase.

3.2 Training Phase

At the beginning of the iteration, the *population* $[P]$ is initialized as an empty set. All rules are stored in $[P]$. At the time t, the system receives an input $\mathbf{x}$ from a training dataset that belongs to class c^*. Subsequently, a *match set* $[M]$ is formed as $[M] = \{k \in [P] \mid \mu_{\mathbf{A}^k}(\mathbf{x}) > 0\}$.

After $[M]$ is formed, the system constructs a *correct set* $[C] = \{k \in [M] \mid c^k = c^*\}$. If $\sum_{k \in [C]} \mu_{\mathbf{A}^k}(\mathbf{x}) < 1$, the *covering* operator generates a new rule k_{cov} such that $\mu_{\mathbf{A}^{k_{\mathrm{cov}}}}(\mathbf{x}) = 1$. After forming $[C]$, the parameters of all rules k in $[M]$ are updated as follows: the experience exp^k is updated using $\mathrm{exp}^k \leftarrow \mathrm{exp}^k + \mu_{\mathbf{A}^k}(\mathbf{x})$; the correct matching vector $\mathbf{cm}^k$ for each class c_i is updated using $\mathrm{cm}_i^k \leftarrow \mathrm{cm}_i^k + \mu_{\mathbf{A}^k}(\mathbf{x})$ if $c_i = c^*$, otherwise it remains unchanged; the weight vector $\mathbf{v}^k$ for all c_i is updated using $v_i^k \leftarrow \mathrm{cm}_i^k / \mathrm{exp}^k$; and the fitness F^k

is updated as $F^k \leftarrow v^k_{\max} - \sum_{i|i\neq\max} v^k_i$ [16], where the system subtracts the values of the other weights from the maximum weight $v^k_{\max}$. Finally, the highest weight $v^k_{\max}$ in the weight vector $\mathbf{v}^k$ and its corresponding class label $c_{\max}$ are designated as the rule-weight w^k and the rule-consequent class c^k in Eq. (1), respectively.

After the rules are updated, a steady-state GA is applied to $[C]$. The GA is activated when the average time since the GA was last applied to a rule in $[C]$ exceeds the hyperparameter θ_{GA}. When the GA is activated, two parent rules from $[C]$ are selected through tournament selection [8], with the tournament size determined by the hyperparameter τ. In Fuzzy-UCS, tournament selection proceeds as follows: (i) A random sample of $\tau \times \sum_{k\in[C]|F^k\geq 0} \mathrm{num}^k$ rules is selected from $[C]$, excluding any rules with negative fitness; (ii) The rule k with the highest value of $(F^k)^\nu \cdot \mu_{\mathbf{A}^k}(\mathbf{x})$, where ν is a hyperparameter controlling selection pressure, is chosen as the parent rule from the sample obtained in (i). The two parent rules k_{p1} and k_{p2} are replicated as two child rules k_{ch1} and k_{ch2}, after which crossover and mutation are applied to the child rules with probabilities χ and p_{mut}, respectively. The two child rules are inserted into $[P]$, and two rules are deleted if the total number of rules in $[P]$, $\sum_{k\in[P]} \mathrm{num}^k$, exceeds the maximum ruleset size N.

3.3 Test Phase

When an input vector $\mathbf{x}$ from a test dataset is provided, all sufficiently trained rules participate in voting for the predicted class. First, the number of votes, vote_i, for each class c_i in the set of classes $\mathcal{C}$, is calculated using $\mathrm{vote}_i = \sum_{k\in[M]|c^k=c_i\wedge \exp^k>\theta_{\mathrm{exploit}}} v_k$, where $v_k = F^k \cdot \mu_{\mathbf{A}^k}(\mathbf{x}) \cdot \mathrm{num}^k$ represents the number of votes that rule k casts for the rule-consequent class c^k, and $\theta_{\mathrm{exploit}} \in \mathbb{R}^+$ is a hyperparameter. Finally, the class receiving the most votes is output as the system's inference result.

4 The Proposed Algorithm

This section introduces Fuzzy-UCSv, an enhanced version of Fuzzy-UCS that utilizes rule-antecedents with variable-length fuzzy sets. The key features of Fuzzy-UCSv are as follows:

- Unlike Fuzzy-UCS, where the rule-antecedent comprises a fixed-length fuzzy set vector (i.e., each variable x_i is conditioned by a single fuzzy set, A^k_i), Fuzzy-UCSv uses a variable-length fuzzy set vector. Here, the variable x_i is conditioned by $m^k_i \in \mathbb{N}_0$ fuzzy sets, represented as $\mathbf{A}^k_i = \left(A^k_{i,1}, ..., A^k_{i,m^k_i}\right)$.
- The length of each rule's antecedent is optimized by the GA.

The following subsections provide a detailed description of the modifications made to Fuzzy-UCS. It is important to note that while Fuzzy-UCS(v) in this paper uses a triangular-shaped membership function for the rule-antecedent, it can incorporate any type of membership function, such as interval-shaped [36], trapezoidal-shaped [33] and Gaussian-shaped [37].

4.1 Knowledge Representation

In Fuzzy-UCSv, a d-dimensional fuzzy rule k is expressed by:

$$\textbf{IF } x_1 \text{ is } \mathbf{A}_1^k \text{ and } \cdots \text{ and } x_d \text{ is } \mathbf{A}_d^k \textbf{ THEN } c^k \textbf{ WITH } w^k. \tag{2}$$

The notation in the rule-antecedent of Eq. (2) is as follows:

- The rule-antecedent vector $\mathbf{A}^k = \left(\mathbf{A}_1^k, \mathbf{A}_2^k, ..., \mathbf{A}_d^k\right)$ is represented by a fixed-length vector of length $d \in \mathbb{N}$.
- Each variable x_i is conditioned by a variable-length vector that we call the *rule-antecedent fuzzy set vector*, $\mathbf{A}_i^k = \left(A_{i,1}^k, A_{i,2}^k, ..., A_{i,m_i^k}^k\right)$, where $m_i^k \in \mathbb{N}_0$ is the length of $\mathbf{A}_i^k$.
- The j-th rule-antecedent fuzzy set in the i-th dimension is defined by $A_{i,j}^k = \left(a_{i,j}^k, b_{i,j}^k, c_{i,j}^k\right)$ where $a_{i,j}^k, b_{i,j}^k, c_{i,j}^k \in \mathbb{R}$ $(a_{i,j}^k \leq b_{i,j}^k \leq c_{i,j}^k)$ are the vertices of a triangle.

Therefore, while the number of rule-antecedent parameters in Fuzzy-UCS is $3d$, in Fuzzy-UCSv, it becomes $\sum_{i=1}^{d} 3m_i^k$.

Figures 1 and 2 show the rule landscapes for Fuzzy-UCS (with fixed-length fuzzy set vectors) and Fuzzy-UCSv (with variable-length fuzzy set vectors), respectively. The green and orange points represent training data points belonging to classes c_1 and c_2, respectively. In the tasks shown on the left side of Figs. 1 and 2, it is obvious that only x_1 is crucial for classifying class c_1. However, in Fuzzy-UCS, membership functions are also (unnecessarily) optimized for x_2, which is irrelevant for classification, highlighting Problem 1 outlined in Sect. 1. Conversely, Fuzzy-UCSv effectively addresses this problem by representing x_2 as a *Don't Care* condition. Additionally, in the tasks shown on the right side of Figs. 1 and 2, the data points for class c_1 are spread across six subspaces. While Fuzzy-UCS requires at least six rules to cover the data points for class c_1, since a single rule can only cover one subspace, indicating Problem 2 in Sect. 1. On the other hand, Fuzzy-UCSv can cover multiple subspaces with a single rule by incorporating multiple fuzzy sets for each variable.

Comparative experiments and discussions on the effectiveness of fixed- and variable-length fuzzy set vectors are given in Sect. 5.1.

4.2 Membership Degree Calculation

In Fuzzy-UCSv, the calculation of the membership degree between a given input $\mathbf{x}$ and a rule k, $\mu_{\mathbf{A}^k}(\mathbf{x})$, is given by $\mu_{\mathbf{A}^k}(\mathbf{x}) = \prod_{i=1}^{d} \mu_{\mathbf{A}_i^k}(x_i)$, where $\mu_{\mathbf{A}_i^k}(x_i)$ is given by:

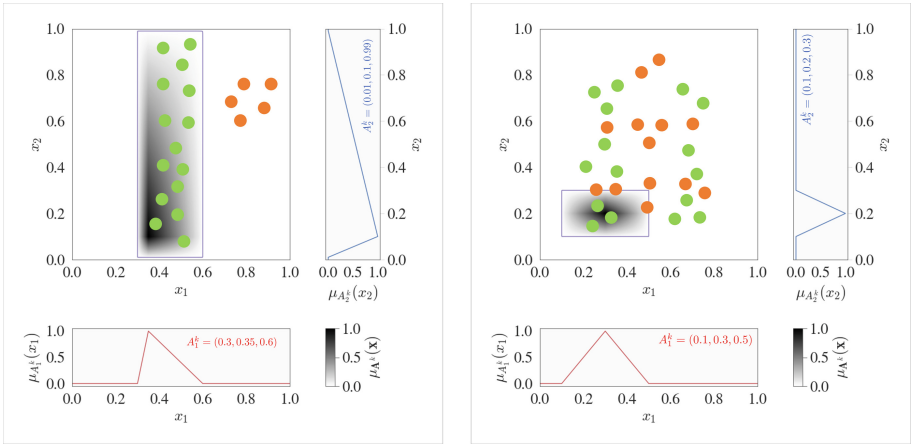


Fig. 1. Examples of a rule in Fuzzy-UCS (with fixed-length fuzzy set vectors).

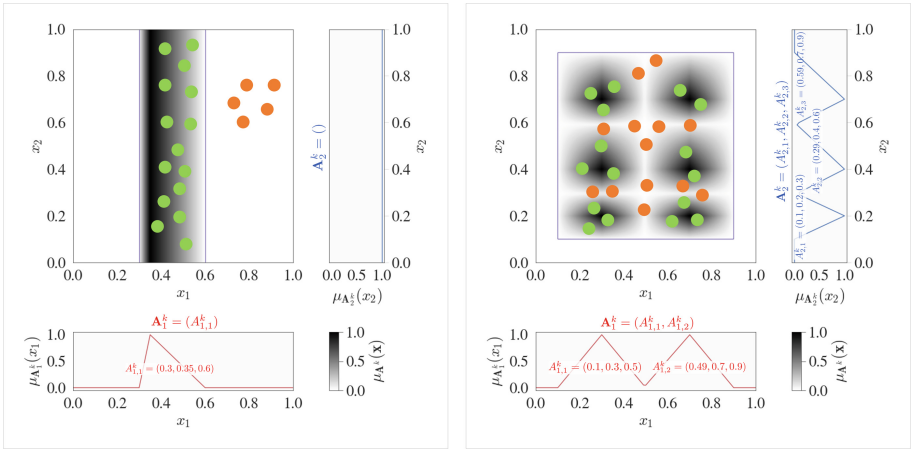


Fig. 2. Examples of a rule in Fuzzy-UCSv (with variable-length fuzzy set vectors).

$$\mu_{\mathbf{A}_i^k}(x_i) = \begin{cases} 1 & \text{if } \mathbf{A}_i^k = (), \\ \min\left\{1, \sum_{j=1}^{m_i^k} \mu_{A_{i,j}^k}(x_i)\right\} & \text{otherwise,} \end{cases} \tag{3}$$

where $\mu_{A_{i,j}^k} : [0,1] \to [0,1]$ is the membership function of the fuzzy set $A_{i,j}^k$. $\mathbf{A}_i^k = ()$ indicates that $\mathbf{A}_i^k$ is a zero-length vector, which is equivalent to a vector of empty sets.

As indicated by Eq. (3), if there is no fuzzy set in the i-th dimension of the rule-antecedent (i.e., $m_i^k = 0$ is satisfied), then that dimension is treated as *Don't Care*, and the membership degree $\mu_{\mathbf{A}_i^k}(x_i)$ becomes 1, which is the maximum

possible value. On the other hand, if there are fuzzy sets present (i.e., $m_i^k \geq 1$ is satisfied), the membership degree $\mu_{\mathbf{A}_i^k}(x_i)$ is set to the value aggregated by the *Łukasiewicz T-conorm* operator (i.e., $\min\{1, a+b\}$) [28] from the membership degrees $\mu_{A_{i,j}^k}(x_i)$ of each fuzzy set $A_{i,j}^k$ for $\forall j \in \{1, \ldots, m_i^k\}$.

4.3 Covering Operator

In Fuzzy-UCSv, the covering operator generates a new rule k_{cov} with rule-antecedent vector $\mathbf{A}^{k_{\text{cov}}} = \left(\mathbf{A}_1^{k_{\text{cov}}}, \mathbf{A}_2^{k_{\text{cov}}}, ..., \mathbf{A}_d^{k_{\text{cov}}}\right)$ for an input vector $\mathbf{x} \in [0,1]^d$ defined by:

$$\forall i \in \{1, ..., d\} : \mathbf{A}_i^{k_{\text{cov}}} = \begin{cases} () & \text{if } \mathcal{U}[0,1) < P_{\#} \text{ or } x_i \text{ is a missing value,} \\ \left(A_{i,j}^{k_{\text{cov}}}\right)_{j=1}^{n_{\text{init}}} & \text{otherwise,} \end{cases} \tag{4}$$

where $\mathcal{U}[0,1)$ is a uniformly distributed random number in the range $[0,1)$; $P_{\#}$ is a hyperparameter that indicates the probability of treating each dimension of the rule-antecedent as *Don't Care*; and $n_{\text{init}} \in \mathbb{N}$ is a hyperparameter that determines the number of fuzzy sets for each input i of the created fuzzy rule.

The j-th rule-antecedent fuzzy set in the i-th dimension of rule k_{cov}, $A_{i,j}^{k_{\text{cov}}} = \left(a_{i,j}^{k_{\text{cov}}}, b_{i,j}^{k_{\text{cov}}}, c_{i,j}^{k_{\text{cov}}}\right)$, is given by:

$$\forall j \in \{1, \ldots, n_{\text{init}}\} : a_{i,j}^{k_{\text{cov}}} = \mathcal{U}[-0.5, x_i],\ b_{i,j}^{k_{\text{cov}}} = x_i,\ c_{i,j}^{k_{\text{cov}}} = \mathcal{U}[x_i, 1.5]. \tag{5}$$

Comparative experiments and discussions on the effects of the value of n_{init} are given in Sect. 5.2.

4.4 Rule Discovery

In Fuzzy-UCSv, as in Fuzzy-UCS, crossover and mutation operators are applied to the child rules k_{ch1} and k_{ch2}. During these operations, the system optimizes the parameters of the rule-antecedent fuzzy sets (vertices of a triangle) associated with each rule, along with the length of the rule-antecedent fuzzy set vector.

Crossover Operator. In Fuzzy-UCSv, the system applies uniform crossover to the rule-antecedent fuzzy set vectors $\mathbf{A}_i^k$ (where $k \in \{k_{\text{ch1}}, k_{\text{ch2}}\}$).

If $\left|\mathbf{A}_i^{k_{\text{ch1}}}\right| = \left|\mathbf{A}_i^{k_{\text{ch2}}}\right|$, meaning $m_i^{k_{\text{ch1}}} = m_i^{k_{\text{ch2}}}$, then for each j-th fuzzy set $A_{i,j}^k$, the parameters $a_{i,j}^{k_{\text{ch1}}}$ and $a_{i,j}^{k_{\text{ch2}}}$, $b_{i,j}^{k_{\text{ch1}}}$ and $b_{i,j}^{k_{\text{ch2}}}$, $c_{i,j}^{k_{\text{ch1}}}$ and $c_{i,j}^{k_{\text{ch2}}}$ are swapped with a probability of 0.5.

If $m_i^{k_{\text{ch1}}} \neq m_i^{k_{\text{ch2}}}$, the uniform crossover is similarly applied up to the minimum length of $\mathbf{A}_i^k$. For indices j beyond this range, elements $A_{i,j}^{k_{\text{ch1}}}$ and $A_{i,j}^{k_{\text{ch2}}}$ (one of them is an empty set) from $\mathbf{A}_i^{k_{\text{ch1}}}$ and $\mathbf{A}_i^{k_{\text{ch2}}}$ are swapped with a probability of 0.5. For instance, if $A_{i,j}^{k_{\text{ch1}}} = \emptyset$ and $A_{i,j}^{k_{\text{ch2}}} = (a, b, c)$, then with a probability of 0.5, $A_{i,j}^{k_{\text{ch1}}}$ becomes (a, b, c), and $A_{i,j}^{k_{\text{ch2}}}$ becomes $\emptyset$.

Mutation Operator. In Fuzzy-UCSv, the mutation operator is applied to the rule-antecedent fuzzy set vectors $\mathbf{A}_i^k = \left(A_{i,1}^k, A_{i,2}^k, ..., A_{i,m_i^k}^k\right)$ for rule $\forall k \in \{k_{\text{ch1}}, k_{\text{ch2}}\}$.

With a probability p_{mut}, the system mutates the parameters $a_{i,j}^k$, $b_{i,j}^k$, and $c_{i,j}^k$ of the triangular membership function for each antecedent fuzzy set $A_{i,j}^k (\forall j \in \{1, ..., m_i^k\})$. The mutation of the center vertex $b_{i,j}^k$ follows the following equation:

$$b_{i,j}^k \leftarrow \mathcal{U}\left[b_{i,j}^k - (b_{i,j}^k - a_{i,j}^k) \cdot m_0, b_{i,j}^k + (c_{i,j}^k - b_{i,j}^k) \cdot m_0\right], \tag{6}$$

where $m_0 \in (0, 1]$ is a hyperparameter defining the maximum mutation extent. Subsequently, if rule k is accurate (i.e., $F^k > F_0$) and not generated by crossover, the left vertex $a_{i,j}^k$ and the right vertex $c_{i,j}^k$ are mutated using Eq. (7):

$$a_{i,j}^k \leftarrow \mathcal{U}\left[a_{i,j}^k - \frac{b_{i,j}^k - a_{i,j}^k}{2} m_0, a_{i,j}^k\right],\ c_{i,j}^k \leftarrow \mathcal{U}\left[c_{i,j}^k, c_{i,j}^k + \frac{c_{i,j}^k - b_{i,j}^k}{2} m_0\right]. \tag{7}$$

Otherwise, Eq. (8) is used:

$$\begin{cases} a_{i,j}^k & \leftarrow \mathcal{U}\left[a_{i,j}^k - \dfrac{b_{i,j}^k - a_{i,j}^k}{2} m_0, a_{i,j}^k + \dfrac{b_{i,j}^k - a_{i,j}^k}{2} m_0\right], \\ c_{i,j}^k & \leftarrow \mathcal{U}\left[c_{i,j}^k - \dfrac{c_{i,j}^k - b_{i,j}^k}{2} m_0, c_{i,j}^k + \dfrac{c_{i,j}^k - b_{i,j}^k}{2} m_0\right]. \end{cases} \tag{8}$$

Equation (7) is intended to enforce the generalization of rule k when the rule is guaranteed to be accurate by the system. If not, Eq. (8) is used to execute either generalization or specialization. This setting enhances the evolutionary pressure to acquire a set of rules that are both accurate and general [27,31].

Furthermore, with a probability p_{mut}, a newly generated fuzzy set $A_{\text{new}} = (a_{\text{new}}, b_{\text{new}}, c_{\text{new}})$ is added to $\mathbf{A}_i^k$. Specifically, the system first establishes the parameters $(a_{\text{new}}, b_{\text{new}}, c_{\text{new}})$ of the triangular membership function representing A_{new} using Eq. (5). Then, the system updates $\mathbf{A}_i^k$ by appending A_{new}:

$$\mathbf{A}_i^k \leftarrow \left(\mathbf{A}_i^k, A_{\text{new}}\right). \tag{9}$$

Lastly, if $\mathbf{A}_i^k$ is not empty, then with a probability p_{mut}, a fuzzy set is randomly removed from $\mathbf{A}_i^k = \left(A_{i,1}^k, A_{i,2}^k, ..., A_{i,m_i^k}^k\right)$.

In summary, our mutation operator applies equal probabilistic pressure to either increase or decrease the number of fuzzy sets. This dynamic adaptation helps prevent the excessive growth of the number of fuzzy sets and *Don't Care* conditions, ensuring they do not become larger than necessary.

5 Experiments

This section utilizes 20 real-world datasets in Table 1 from the *UCI Machine Learning Repository* [12] and *Kaggle Dataset* collection. These datasets are selected due to their inherent challenges in data classification, such as the presence of missing values and class imbalance issues [24].

Table 1. Properties of the 20 real-world datasets. The columns describe: the identifier (ID.), the name (Name), the number of instances ($|\mathcal{D}|$), the total number of features (d), the number of classes (n), and the source (Ref.).

ID.	Name	$\|\mathcal{D}\|$	d	n	Ref.	ID.	Name	$\|\mathcal{D}\|$	d	n	Ref.
brs	Breast cancer Wisconsin	699	9	2	[12]	pdy	Paddy Leaf	6000	3	4	[18]
can	Cancer	599	30	2	[17]	pis	Pistachio	2148	16	2	[34]
col	Column 3C weka	310	6	3	[12]	pmp	Pumpkin	2499	12	2	[20]
dbt	Diabetes	768	8	2	[12]	rsn	Raisin	900	7	2	[10]
ecl	Ecoli	336	7	8	[12]	seg	Segment	2310	19	7	[12]
frt	Fruit	898	34	7	[19]	soy	Soybean	683	35	19	[12]
gls	Glass	214	9	6	[12]	tae	Teaching assistant evaluation	151	5	3	[12]
hcl	Horse colic	368	22	2	[12]	wne	Wine	178	13	3	[12]
irs	Iris	150	4	3	[12]	wpb	Wisconsin prognostic breast cancer	198	33	2	[12]
mam	Mammographic masses	961	5	2	[12]	yst	Yeast	1484	8	10	[12]

5.1 Experiment 1: Fixed Vs. Variable-Length Fuzzy Sets

We compare the performance of Fuzzy-UCS and Fuzzy-UCSv to investigate the impact of variable-length fuzzy sets on the performance of LFCS in situations where the maximum number of rules N is limited ($N = 500$) and in situations where a sufficient number of rules is provided ($N = 2000$).

Experimental Setup. The hyperparameters for Fuzzy-UCS are set to the same values as those in previous studies [27,38,39]: $N \in \{500, 2000\}$, $F_0 = 0.99$, $\nu = 1$, $\chi = 0.8$, $p_{\text{mut}} = 0.04$, $\delta = 0.1$, $m_0 = 0.1$, $\theta_{\text{GA}} = 50$, $\theta_{\text{del}} = 50$, $\theta_{\text{exploit}} = 10$, $\tau = 0.4$, and $P_{\#} = 0.33$. Fuzzy-UCSv retains the same parameter settings as Fuzzy-UCS, except for $n_{\text{init}} = 1$. Uniform crossover is employed in the GA. The number of training epochs is fixed at 50, and each data attribute value is normalized to a real number in the range $[0, 1]$. The performance metric is the average classification accuracy on the training and test data across 30 trials, utilizing shuffle-split cross-validation with 90% of the data for training and 10% for testing. Additionally, to verify statistical significance, the Wilcoxon signed-rank test is applied for each dataset at a significance level of 0.05.

Results and Discussion. Table 2 presents the summary of results.

From Table 2, it is evident that Fuzzy-UCSv achieves improvements in test accuracy across many datasets without significantly deteriorating test accuracy compared to Fuzzy-UCS, for both $N \in \{500, 2000\}$. Therefore, irrespective of the value of N, variable-length fuzzy sets contribute to enhancing the test accuracy of LFCSs. Particularly in high-dimensional datasets with greater than or equal to 30 dimensions, at the setting of $N = 500$, Fuzzy-UCSv achieves significant improvements in test accuracy (25.32% in `can`, 75.60% in `frt`, 52.61% in `soy`, and 3.50% in `wpb`). This indicates that variable-length fuzzy sets are particularly effective in high-dimensional datasets and when the maximum number of rules is constrained.

On the other hand, in some cases, Fuzzy-UCS demonstrates significantly higher training accuracy than Fuzzy-UCSv. Specifically, with $N = 2000$, Fuzzy-

Table 2. Summary of results from Experiment 1, displaying average training and test accuracy (%). Statistical results of the Wilcoxon signed-rank test are summarized with symbols wherein "+", "−", and "∼" represent that the classification accuracy of the conventional Fuzzy-UCS is significantly better, worse, and competitive compared to that obtained by the proposed Fuzzy-UCSv, respectively. Green-shaded values denote the best values. The "p-value" is derived from the Wilcoxon signed-rank test.

	$N = 500$				$N = 2000$			
	Train		Test		Train		Test	
ID.	Fuzzy-UCS	Fuzzy-UCSv	Fuzzy-UCS	Fuzzy-UCSv	Fuzzy-UCS	Fuzzy-UCSv	Fuzzy-UCS	Fuzzy-UCSv
brs	95.52 −	96.56	94.43 −	96.33	96.44 ∼	96.63	94.29 −	96.10
can	68.00 −	93.60	67.72 −	93.04	94.38 +	92.66	89.12 −	91.99
col	84.25 ∼	83.99	80.54 ∼	79.46	81.61 ∼	82.09	78.92 ∼	78.49
dbt	77.12 ∼	77.35	73.94 ∼	74.85	78.50 +	77.87	73.90 ∼	74.94
ecl	88.31 ∼	88.07	84.51 ∼	84.71	87.60 −	88.34	85.00 ∼	86.08
frt	8.391 −	85.12	8.333 −	83.93	81.79 −	86.15	76.59 −	82.04
gls	80.56 +	78.87	65.00 ∼	67.42	80.49 ∼	81.65	65.00 −	70.30
hcl	60.05 −	73.23	57.30 −	70.18	86.61 +	85.57	67.66 ∼	70.45
irs	95.26 −	95.75	94.67 ∼	95.56	95.04 −	95.83	94.00 ∼	95.78
mam	82.35 ∼	82.55	80.96 −	81.99	81.22 −	81.93	81.13 ∼	81.96
pdy	84.60 ∼	84.30	84.44 ∼	83.95	87.77 ∼	87.73	87.40 ∼	87.74
pis	87.10 +	86.74	86.33 ∼	85.97	87.68 +	87.21	86.59 ∼	86.19
pmp	87.61 ∼	87.45	86.87 ∼	86.41	87.74 ∼	87.64	86.71 ∼	86.81
rsn	85.78 ∼	85.78	85.48 ∼	85.89	85.93 ∼	85.90	85.93 ∼	85.89
seg	91.20 −	93.74	90.79 −	93.35	95.36 −	95.88	93.82 −	95.15
soy	5.434 −	61.30	5.072 −	57.68	67.39 −	95.07	60.58 −	87.15
tae	69.95 ∼	68.35	53.33 ∼	54.58	64.30 +	62.35	51.46 ∼	51.46
wne	98.08 +	96.50	94.26 ∼	92.22	99.29 +	97.77	95.00 ∼	95.37
wpb	74.85 −	79.64	72.67 −	76.17	91.03 −	96.82	61.50 −	74.17
yst	54.01 −	56.06	51.83 −	53.62	63.09 ∼	63.19	59.53 ∼	59.73
+/ − / ∼	3/9/8	-	0/9/11	-	6/7/7	-	0/7/13	-
p-value	0.123	-	0.00558	-	0.411	-	0.000420	-

UCS exhibits significantly higher training accuracy in six datasets and significantly lower in seven datasets compared to Fuzzy-UCSv. This indicates that when N is sufficiently large, Fuzzy-UCS can achieve training accuracy comparable to that of Fuzzy-UCSv. However, Fuzzy-UCS did not significantly outperform Fuzzy-UCSv in test accuracy in any of the datasets. This could be attributed to scenarios like those depicted on the right side of Figs. 1 and 2, where class data points are distributed across multiple subspaces. Fuzzy-UCS attempts to cover one subspace with one specific fuzzy rule, as illustrated in Fig. 1. When N is sufficiently large, Fuzzy-UCS can generate rules with optimal rule-weights and rule-fitness for each subspace, enabling precise classification of training data. However, this specificity can lead to overfitting, as indicated in Table 2. For instance, while Fuzzy-UCS shows significantly higher training accuracy than Fuzzy-UCSv in the `can` problem, its test accuracy is significantly lower. Conversely, Fuzzy-UCSv, by aiming to classify multiple subspaces with a single general fuzzy rule, might not classify training data as accurately as Fuzzy-UCS but effectively avoids overfitting, thereby enhancing generalization performance on test data.

Table 3. Summary of results from Experiment 2. Symbols "+", "−", and "∼" represent that the classification accuracy of the variant of Fuzzy-UCSv is significantly better, worse, and competitive compared to that obtained by Fuzzy-UCS, respectively. "Rank" and "Position" denote each system's overall average rank obtained by using the Friedman test and its position in the final ranking, respectively. Green- and peach-shaded values denote the best and the worst, respectively. The "p_{Holm}-value" is derived from the Holm-adjusted Wilcoxon signed-rank test.

	Train				Test			
	Fuzzy-UCS	Fuzzy-UCSv			Fuzzy-UCS	Fuzzy-UCSv		
		$n_{\text{init}} = 1$	$n_{\text{init}} = 2$	$n_{\text{init}} = 3$		$n_{\text{init}} = 1$	$n_{\text{init}} = 2$	$n_{\text{init}} = 3$
Rank	*1.90*	*1.65*	*2.80*	*3.65*	*2.68*	*1.73*	*2.60*	*3.00*
Position	*2*	*1*	*3*	*4*	*3*	*1*	*2*	*4*
$+/-/\sim$	-	7/6/7	2/13/5	2/15/3	-	7/0/13	5/3/12	5/4/11
p-value	-	0.411	0.0192	0.00639	-	0.000420	0.571	0.498
p_{Holm}-value	-	0.411	0.0385	0.0192	-	0.00126	0.996	0.996

5.2 Experiment 2: On the Effects of n_{init} for Fuzzy-UCSv

We compare the performance of Fuzzy-UCS and Fuzzy-UCSv to investigate the impact of the hyperparameter n_{init}, which controls the number of fuzzy sets held by the rules generated by the covering operator, in Fuzzy-UCSv.

Experimental Setup. For Fuzzy-UCS and Fuzzy-UCSv, the hyperparameters are consistent with those in Experiment 1, except that $N = 2000$. For Fuzzy-UCSv, $n_{\text{init}} \in \{1, 2, 3\}$. The number of training epochs for all systems is fixed at 50, and statistical significance is verified using the same procedure as in Experiment 1 (cf. Sect. 5.1). Additionally, to compare the overall performance, a Friedman test with Holm's post-hoc test is conducted at a significance level of 0.05.

Results and Discussion. Table 3 presents the summary of results.

Table 3 indicates that the only n_{init} setting that significantly improved test accuracy over Fuzzy-UCS was $n_{\text{init}} = 1$. The settings of $n_{\text{init}} \in \{2, 3\}$ yielded significantly worse training accuracy compared to Fuzzy-UCS, and they did not show any significant differences in test accuracy. Furthermore, there were no datasets where the test accuracy was significantly worse than Fuzzy-UCS for the $n_{\text{init}} = 1$ setting; however, there were three and four datasets, respectively, for the $n_{\text{init}} = 2$ and $n_{\text{init}} = 3$ settings where the test accuracy was worse. In other words, the $n_{\text{init}} \in \{2, 3\}$ settings tend to cause underfitting of the training data for some datasets, which, in turn, reduces test accuracy. This underfitting is attributed to an *over-generalization* of the rules [21], caused by some dimensions of the rule-antecedent holding more fuzzy sets than necessary. As mentioned in Sect. 4.4, the Fuzzy-UCSv mutation operator applies equal pressure to increase and decrease the number of fuzzy sets. Additionally, as highlighted in recent research [32], many L(F)CSs, including Fuzzy-UCS, lack a mechanism to remove over-general rules explicitly. Therefore, if n_{init} is set too high (e.g., 3), it requires extensive learning to optimize the number of fuzzy sets in the rule to an appropriate value via the GA (i.e., rule specialization).

Table 4. Summary of results from Experiment 3. Symbols "+", "−", and "∼" represent that the existing algorithm is significantly better, worse, and competitive compared to Fuzzy-UCSv, respectively. "Rank", "Position", "p-value", "p_{Holm}-value", and green- and peach-shaded values should be interpreted as in Table 3.

	Train				Test			
ID.	Scikit-MLP	UCS	Fuzzy-UCS	Fuzzy-UCSv	Scikit-MLP	UCS	Fuzzy-UCS	Fuzzy-UCSv
brs	96.90 +	97.62 +	96.44 ∼	96.63	96.24 ∼	95.86 ∼	94.29 −	96.10
can	93.50 +	97.43 +	94.38 +	92.66	92.98 ∼	91.93 ∼	89.12 −	91.99
col	67.63 −	77.19 −	81.61 ∼	82.09	66.88 −	72.26 −	78.92 ∼	78.49
dbt	72.89 −	77.88 ∼	78.50 +	77.87	70.35 −	72.38 −	73.90 ∼	74.94
ecl	52.89 −	77.69 −	87.60 −	88.34	52.84 −	74.51 −	85.00 ∼	86.08
frt	77.36 −	93.51 +	81.79 −	86.15	77.04 −	83.44 +	76.59 −	82.04
gls	39.17 −	65.47 −	80.49 ∼	81.65	36.52 −	56.36 −	65.00 −	70.30
hcl	82.74 −	89.99 +	86.61 +	85.57	77.12 +	63.51 −	67.66 ∼	70.45
irs	76.67 −	84.59 −	95.04 −	95.83	74.44 −	82.44 −	94.00 ∼	95.78
mam	79.31 −	83.13 +	81.22 −	81.93	79.42 −	80.72 ∼	81.13 ∼	81.96
pdy	86.99 −	89.27 +	87.77 ∼	87.73	87.12 −	88.22 ∼	87.40 ∼	87.74
pis	86.67 −	87.33 ∼	87.68 +	87.21	85.88 ∼	85.81 ∼	86.59 ∼	86.19
pmp	86.74 −	87.63 ∼	87.74 ∼	87.64	86.21 −	86.31 ∼	86.71 ∼	86.81
rsn	85.77 ∼	84.88 −	85.93 ∼	85.90	85.67 ∼	83.81 −	85.93 ∼	85.89
seg	89.02 −	95.42 −	95.36 −	95.88	89.35 −	93.46 −	93.82 −	95.15
soy	88.52 −	95.28 ∼	67.39 −	95.07	86.86 ∼	65.41 −	60.58 −	87.15
tae	56.64 −	59.80 −	64.30 +	62.35	55.62 ∼	50.00 ∼	51.46 ∼	51.46
wne	94.50 −	94.23 −	99.29 +	97.77	90.19 −	87.78 −	95.00 ∼	95.37
wpb	76.39 −	95.45 −	91.03 −	96.82	76.00 +	68.00 −	61.50 −	74.17
yst	52.86 −	60.32 −	63.09 ∼	63.19	52.93 −	55.86 −	59.53 ∼	59.73
Rank	*3.70*	*2.20*	*2.10*	*2.00*	*2.95*	*3.05*	*2.48*	*1.53*
Position	*4*	*3*	*2*	*1*	*3*	*4*	*2*	*1*
+/ − / ∼	2/17/1	6/10/4	6/7/7	-	2/12/6	1/12/7	0/7/13	-
p-value	3.62E-5	0.330	0.411	-	0.0153	0.000105	0.000420	-
p_{Holm}-value	0.000109	0.660	0.660	-	0.0153	0.000315	0.000839	-

Based on these observations, n_{init} should be set to 1 to balance between rule generalization and specialization.

5.3 Experiment 3: Comparison of Fuzzy-UCSv to Several Machine Learning Techniques

To comprehensively evaluate the performance of Fuzzy-UCSv, we compare it with a representative classifier from the *scikit-learn* library [29], the *Multi-Layer Perceptron Neural Network* (Scikit-MLP), along with the *sUpervised Classifier System* (UCS) [5], which is currently the most successful LCS in supervised learning [25,31], and Fuzzy-UCS. While Scikit-MLP classifies data points using a single global model (a neural network) that covers the entire input space, UCS and Fuzzy-UCS(v) classify data points using multiple local models (rules).

Experimental Setup. The hyperparameters for Scikit-MLP are set to the default values provided by the scikit-learn library. For UCS, the hyperparameters are configured as follows [27,38,39]: $N = 2000$, $\text{acc}_0 = 0.99$, $\beta = 0.2$, $\nu = 1$, $\chi = 0.8$, $p_{\text{mut}} = 0.04$, $\delta = 0.1$, $m_0 = 0.1$, $r_0 = 1.0$, $\theta_{\text{GA}} = 50$, $\theta_{\text{del}} = 50$, $\tau = 0.4$, $P_{\#} =$

0.33, with the *unordered bound hyperrectangular representation* [36] employed for the rule-antecedent. For Fuzzy-UCS and Fuzzy-UCSv, the hyperparameters are consistent with those in Experiment 2 (cf. Sect. 5.2). For Fuzzy-UCSv, $n_{\text{init}} = 1$. The number of training epochs for all systems is fixed at 50, and statistical significance is verified using the same procedure as in Experiment 2.

Results and Discussion. Table 4 presents the summary of results.

From Table 4, we can observe that our proposed Fuzzy-UCSv significantly outperforms the three conventional systems in terms of test accuracy. In addition, Table 4 shows that among all systems, only Fuzzy-UCSv avoids recording the worst test accuracy, i.e., no peach cells. Furthermore, 10 out of 20 cells on test accuracy are depicted in green, i.e., Fuzzy-UCSv records the best test accuracy. This result underscores the effectiveness of Fuzzy-UCSv across various real-world datasets. An interesting observation is that the non-fuzzy system, UCS, achieves the best training accuracy in seven datasets, which is matched by Fuzzy-UCSv. However, UCS only records the best test accuracy for two datasets and significantly underperforms compared to Fuzzy-UCSv in terms of test accuracy. This result is consistent with previous research [31], which shows that non-fuzzy rules are more prone to overfitting on training data. Furthermore, as shown in a recent study [30], constructing a single global model that covers the entire input space for a problem can be time-consuming both in terms of model training and evaluation. This tendency is even more pronounced for complex models such as neural networks, as shown in Table 4.

In summary, Fuzzy-UCSv exploits the advantages of fuzzy rules, such as the suppression of overfitting and the robustness against uncertainty [31], while adaptively constructing multiple simple local models (i.e., fuzzy rules) using the divide-and-conquer principle. This approach demonstrates that Fuzzy-UCSv can effectively solve complex tasks such as real-world data mining.

6 Concluding Remarks

This paper introduced Fuzzy-UCSv, an LFCS that utilizes variable-length fuzzy sets in rule-antecedents. We aimed to develop a set of fuzzy rules capable of disregarding irrelevant features for classification while effectively covering training data points distributed in multiple subspaces with a single rule. Fuzzy-UCSv demonstrated superior performance on real-world data classification problems, exhibiting significantly better test accuracy than Fuzzy-UCS, which employs conventional fixed-length fuzzy sets, as well as two other machine learning techniques while mitigating overfitting on training data.

In future work, we plan to explore the efficacy of variable-length fuzzy sets in a broader range of tasks beyond single-label classification problems, such as regression, reinforcement learning, and multi-label classification problems, using different LFCS variants besides Fuzzy-UCS, e.g., [6,9,26]. Additionally, while this study employed the Łukasiewicz T-conorm operator for membership

degree aggregation, future research should examine the impact of using alternative operators, such as the *Maximum T-conorm* operator [7], on the performance of Fuzzy-UCSv.

Acknowledgements. This work was supported by Japan Society for the Promotion of Science KAKENHI (Grant No. JP23KJ0993), National Natural Science Foundation of China (Grant No. 62250710163, 62376115), and Guangdong Provincial Key Laboratory (Grant No. 2020B121201001).

Disclosure of Interests. The authors declare that they have no conflict of interest.

References

1. Arif, M.H., Iqbal, M., Li, J.: Extracting and reusing blocks of knowledge in learning classifier systems for text classification: a lifelong machine learning approach. Soft. Comput. **23**, 12673–12682 (2019)
2. Arif, M.H., Li, J., Iqbal, M., Peng, H.: Optimizing XCSR for text classification. In: 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE), pp. 86–95. IEEE (2017)
3. Bacardit, J., Burke, E.K., Krasnogor, N.: Improving the scalability of rule-based evolutionary learning. Memetic Comput. **1**, 55–67 (2009)
4. Bacardit, J., Stout, M., Hirst, J.D., Sastry, K., Llora, X., Krasnogor, N.: Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 346–353 (2007)
5. 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)
6. Bishop, J.T., Gallagher, M., Browne, W.N.: A genetic fuzzy system for interpretable and parsimonious reinforcement learning policies. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1630–1638 (2021)
7. Buckley, J., Siler, W.: A new t-norm. Fuzzy Sets Syst. **100**, 283–290 (1998)
8. Butz, M.V., Sastry, K., Goldberg, D.E.: Tournament selection: stable fitness pressure in XCS. 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.) Genetic and Evolutionary Computation — GECCO 2003, pp. 1857–1869. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). https://doi.org/10.1007/3-540-45110-2_83
9. Casillas, J., Carse, B., Bull, L.: Fuzzy-XCS: a Michigan genetic fuzzy system. IEEE Trans. Fuzzy Syst. **15**(4), 536–550 (2007)
10. Çinar, İ, Koklu, M., Taşdemir, Ş: Classification of raisin grains using machine vision and artificial intelligence methods. Gazi Mühendislik Bilimleri Dergisi **6**(3), 200–209 (2020)
11. Debie, E., Shafi, K.: Implications of the curse of dimensionality for supervised learning classifier systems: theoretical and empirical analyses. Pattern Anal. Appl. **22**(2), 519–536 (2019)

12. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
13. Goldberg, D.E.: The design of innovation: lessons from and for competent genetic algorithms, vol. 1. Springer (2002). https://doi.org/10.1007/978-1-4757-3643-4_12
14. Guendouzi, W., Boukra, A.: A new differential evolution algorithm for cooperative fuzzy rule mining: application to anomaly detection. Evol. Intel. **15**(4), 2667–2678 (2022)
15. Holland, J.H.: Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. Mach. Learn. Artif. Intell. Approach **2**, 593–623 (1986)
16. Ishibuchi, H., Yamamoto, T.: Rule weight specification in fuzzy rule-based classification systems. IEEE Trans. Fuzzy Syst. **13**(4), 428–435 (2005)
17. Kaggle: Cancer data. https://www.kaggle.com/datasets/erdemtaha/cancer-data. Accessed 22 June 2024
18. Kaggle: Paddy leaf images (aman). https://www.kaggle.com/datasets/torikul140129/paddy-leaf-images-aman. Accessed 22 June 2024
19. Koklu, M., Kursun, R., Taspinar, Y.S., Cinar, I.: Classification of date fruits into genetic varieties using image analysis. Math. Probl. Eng. **2021**, 1–13 (2021)
20. Koklu, M., Sarigil, S., Ozbek, O.: The use of machine learning methods in classification of pumpkin seeds (cucurbita pepo l.). Genetic Resources Crop Evol. **68**(7), 2713–2726 (2021)
21. Lanzi, P.L.: An analysis of generalization in the XCS classifier system. Evol. Comput. **7**(2), 125–149 (1999)
22. Lanzi, P.L., Wilson, S.W.: Using convex hulls to represent classifier conditions. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 1481–1488 (2006)
23. Nakata, M., Browne, W.N.: Learning optimality theory for accuracy-based learning classifier systems. IEEE Trans. Evol. Comput. **25**(1), 61–74 (2020)
24. Nakata, M., Takadama, K.: An empirical analysis of action map in learning classifier systems. SICE J. Contr. Measure., Syst. Integr. **11**(3), 239–248 (2018)
25. Nazmi, S., Homaifar, A., Anwar, M.: An effective action covering for multi-label learning classifier systems: a graph-theoretic approach. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 340–348 (2021)
26. Omozaki, Y., Masuyama, N., Nojima, Y., Ishibuchi, H.: Evolutionary multi-objective multi-tasking for fuzzy genetics-based machine learning in multi-label classification. In: 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–8. IEEE (2022)
27. Orriols-Puig, A., Casillas, J.: Fuzzy knowledge representation study for incremental learning in data streams and classification problems. Soft. Comput. **15**(12), 2389–2414 (2011)
28. Orriols-Puig, A., Casillas, J., Bernadó-Mansilla, E.: MFuzzy-UCS: a Michigan-style learning fuzzy-classifier system for supervised learning. IEEE Trans. Evol. Comput. **13**(2), 260–283 (2008)
29. Pedrefosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
30. Preen, R.J., Wilson, S.W., Bull, L.: Autoencoding with a classifier system. IEEE Transactions on Evolutionary Computation (2021)
31. Shiraishi, H., Hayamizu, Y., Hashiyama, T.: Fuzzy-UCS revisited: self-adaptation of rule representations in Michigan-style learning fuzzy-classifier systems. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 548–557 (2023)

32. Shiraishi, H., Hayamizu, Y., Sato, H., Takadama, K.: Absumption based on overgenerality and condition-clustering based specialization for XCS with continuous-valued inputs. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 422–430 (2022)
33. Shoeleh, F., Hamzeh, A., Hashemi, S.: Towards final rule set reduction in XCS: a fuzzy representation approach. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 1211–1218 (2011)
34. Singh, D., et al.: Classification and analysis of pistachio species with pre-trained deep learning models. Electronics **11**(7), 981 (2022)
35. Stein, A., Nakata, M.: Learning classifier systems: from principles to modern systems. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 498–527 (2021)
36. Stone, C., Bull, L.: For real! XCS with continuous-valued inputs. Evol. Comput. **11**(3), 299–336 (2003)
37. Tadokoro, M., Sato, H., Takadama, K.: XCS with weight-based matching in VAE latent space and additional learning of high-dimensional data. In: 2021 IEEE Congress on Evolutionary Computation (CEC), pp. 304–310. IEEE (2021)
38. Tzima, F.A., Mitkas, P.A.: Strength-based learning classifier systems revisited: effective rule evolution in supervised classification tasks. Eng. Appl. Artif. Intell. **26**(2), 818–832 (2013)
39. Urbanowicz, R.J., Moore, J.H.: ExSTraCS 2.0: description and evaluation of a scalable learning classifier system. Evol. Intell. **8**(2), 89–116 (2015)
40. Valenzuela-Rendón, M.: The fuzzy classifier system: motivations and first results. In: Schwefel, H.-P., Männer, R. (eds.) Parallel Problem Solving from Nature: 1st Workshop, PPSN I Dortmund, FRG, October 1–3, 1990 Proceedings, pp. 338–342. Springer Berlin Heidelberg, Berlin, Heidelberg (1991). https://doi.org/10.1007/BFb0029774
41. Wilson, S.W.: Get Real! XCS with continuous-valued inputs. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) Learning Classifier Systems: From Foundations to Applications, pp. 209–219. Springer Berlin Heidelberg, Berlin, Heidelberg (2000). https://doi.org/10.1007/3-540-45027-0_11

Evolvable Hardware and Evolutionary Robotics

Exploring Proprioceptive Feedback in the Evolution of Modular Robots

Babak Hosseinkhani Kargar(✉), Karine Miras, and A. E. Eiben

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, Netherlands
{b.hosseinkhanikargar,k.dasilvamirasdearaujo,a.e.eiben}@vu.nl

Abstract. We investigate an evolvable robot system where the body provides proprioceptive sensory signals to the controller (brain) about the positions of the joints. The key aspect we consider is whether all joints should be sensed or if sensing fewer joints would be better. We research this matter based on a test suite of twenty-two robots with various shapes and sizes and implement a system where the controller and the sensory signal system evolve together. Experiments with this system show that the evolved solutions use signals only from a fraction of the joints (25–51%) and perform better than the baseline, where all signals are used. This effect was observed across the majority of the test suite.

Keywords: Evolutionary Robotics · Proprioception · Efficiency

1 Introduction

Evolutionary Robotics (ER) is transforming the field of robotics by harvesting the potential of natural evolution principles [11], enabling robot adaptation in dynamic environments. ER plays a crucial role in pushing the boundaries of autonomous systems and, in the long term, should allow for robots that present high levels of adaptability. Nevertheless, one current shortcoming within ER studies is that when the body is also evolving with the controller (brain), the use of open-loop controllers is abundant [9,14–16,18,21]. Such controllers are usually composed of Central Pattern Generators (CPGs), which can be modeled as equations that produce an oscillatory output [13]. Because of their rhythmic nature, CPGs present the potential to address simple tasks like locomotion regardless of the lack of sensory input.

When contemplating the challenges posed by the simultaneous evolution of the body and controller due to frequent search space shifting, the appeal of CPGs is evident. On the other hand, the use of open-loop controllers is regrettable given that a *robot* is defined as "an artificial device that can sense its environment and purposefully act on or in that environment" [31]. Although CPGs have been shown to be effective for simple tasks in static environments [17,19], their use has led to the creation of robots that lack awareness about the environment and also about their own body in the environment. This limitation is critical

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 405–418, 2024.
https://doi.org/10.1007/978-3-031-70071-2_25

because robots that are "blind" to environmental conditions would be unable to cope with any dynamism in the environment.

In biology, the awareness of an organism about its environment is referred to as *exteroception* [20], and it involves processing signals of sensors that capture the properties of what is external to the organism. Exteroception has been investigated in ER, for instance, focusing on sensory layout [12,29].

Regarding the internal states of an organism, there are two main sensing categories: *interoception*, which comprises the reading of physiological signals [7], and *proprioception*, which regards the awareness that an organism has about the positions of its movable body parts [24,32]. While proprioception is useful for fine-tuning motor skills, e.g., coordination, it can also be useful to perceive the environment. Although proprioception does not allow the robot controller to sense the external environment directly, proprioceptive information allows the robot controller to infer environmental states because of how these states change in response to certain actions. For these reasons, proprioception can be very useful for ER and has been explored in the literature [2–4,19,23,25]. For instance, demonstrating the benefits of proprioception complementary to exteroception [23], and including proprioceptive feedback in the study of ontogenetic development [4]. However, we are unaware of any investigation that specifically addressed the effects of filtering proprioceptive signals.

In traditional machine learning approaches, it is common knowledge that including unnecessary variables in a model can reduce performance due to noise introduction [8]. Nevertheless, because such models are not embodied, proprioception does not apply. Conversely, embodied machines like robots can sense their own bodies through proprioception. Unlike exteroceptive signals, which could involve aspects of the environment that do not affect the robot, proprioceptive signals describe the robot itself. Therefore, it is not obvious whether any data should be ignored when dealing with proprioceptive signals.

Such consideration leads to questioning the need for thorough proprioceptive sensing. To address this matter, the current study evolves populations of robots that can proprioceptively sense their joints. In one of the experiments, robots can sense all of their joints; in another experiment, not all joints may be sensed. Through these experiments, we investigate the following hypothesis: *Applying proprioceptive sensing in a limited number of joints is more effective than its application across all joints.*

2 Methodology

2.1 Robot Framework and Morphology

Revolve [28] is a ER framework developed at the Vrije Universiteit Amsterdam, which forms the foundation of our experimental setup. Revolve utilizes the morphological (body) design space of modular robots from RoboGen [1], but simulated using MuJoCo [30] physics engine. A robot body morphology configures three different types of modules (Fig. 1).

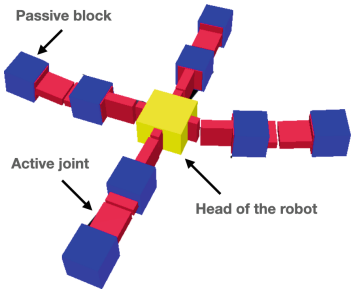


Fig. 1. Example of a robot body morphology. The yellow module is the head of the robot, the blue blocks are passive body parts, and the red modules (made of two parts) are active joints (motors). (Color figure online)

To isolate effects from body evolution, the body morphologies utilized in all current experiments are not evolvable, but fixed. These morphologies (Fig. 2) were derived from a previous study that produced a diverse set of morphologies [10]. Their selection process focused on two primary criteria: morphological viability - ensuring that the robots were capable of effective performance, and diversity in design - to cover a broad range of morphological traits.

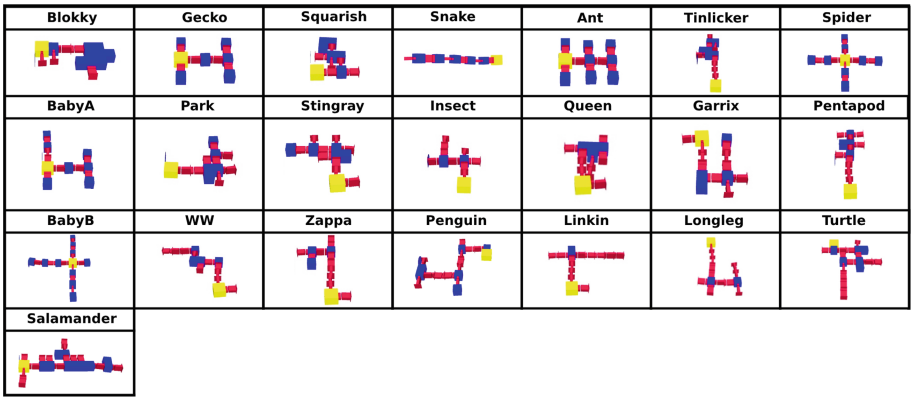


Fig. 2. The robot morphologies test suite. Each picture shows one of the 22 robot morphologies used in the experiments.

2.2 Control Architecture

Artificial Neural Network. We utilize a feedforward Artificial Neural Network (ANN) as a closed-loop controller (brain). Both the weights and topology of the networks are subject to evolution. We chose to evolve the topologies

instead of using fixed ones because the most suitable ANN topology for each body morphology was not known beforehand. The networks receive proprioceptive sensory signals (joint rotations) as inputs. The output neurons utilize 'tanh' as their activation function, ranging from −1 to 1, and are used to actuate the robot by setting new joint rotations.

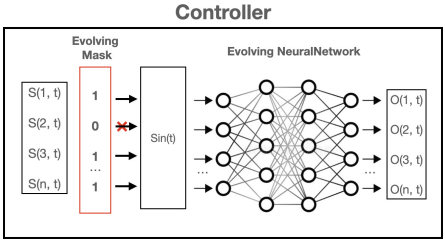


Fig. 3. Schematic representation of the neural network controller and signal filtering mask application. S represents sensors, O represents outputs, and t is the time-step.

Signal Filtering Mask. To experiment with different levels of proprioceptive signal usage, we implemented a binary mask that filters out proprioceptive sensory signals. This mask determines which signals are provided to the network. The mask size is the same as the number of joints (potential signals), and it is applied by multiplying the value of the signal by the value of the mask (0 or 1). Therefore, when the value of the mask is 0, the signal has no effect on the controller.

This way, the controller phenotype is the network, and this network is adjusted by the mask, defining calculations to be performed inside the input nodes.

Oscillators. Because CPGs produce rhythmic patterns, they are expected to produce smooth gaits. Therefore, we decided to modulate the proprioceptive signals with signals derived from oscillators. This was achieved by summing the proprioceptive signal to the signal of a sine function. This way, proprioceptive signals influenced the amplitude of the produced wave. The sinusoidal function is represented as $\sin(t)$. The calculation performed within each input node is defined by Eq. 1.

$$c_i = s_i \times mask_i + \sin(t) \tag{1}$$

where s_i is the original signal of sensor i, c_i is the transformed value of this sensor, $mask_i$ is the binary value that transforms or not s_i to zero, and t is the timestep of the robot life. The oscillatory patterns are introduced through the inputs of the network as opposed to outputs, to guarantee that evolution will not simply exploit the benefits of the oscillators while dodging the challenge of optimizing proprioceptive signals.

Figure 3 illustrates the controller phenotype, combining all three aspects described above: the neural network incorporating the mask and the oscillators.

2.3 Evolutionary Algorithm

The controller genotype is a direct representation formed by two different components: the ANN and the signal-filtering mask. Each one of them is represented using a distinct data structure, and they are used together in the mapping function to produce the final controller phenotype: the ANN utilizing a specific mask.

ANN. The evolution of the ANN involves a modified use of the MultiNEAT library [6]. Unlike the standard NeuroEvolution of Augmenting Topologies (NEAT) algorithm [27], our implementation does not include speciation nor fitness sharing. Instead, we utilize only the variation operators of NEAT (crossover and mutation), applied according to the minimality principle: minimal connections between inputs and outputs and then growing more neurons and connections incrementally.

Signal Filtering Mask. The signal filtering mask is a binary string. This mask is initialized by generating a random binary string (0 or 1) with a size corresponding to the number of active joints in the robot morphology. Crossover might be performed using single-point crossover, and mutation is applied using bit-flipping.

Selection. A number $n = 100$ of parent pairs is selected using tournament selection with $k = 2$ (one tournament per parent of the pair), and each pair generates one offspring. The union operation is performed on the sets of parents and offspring. The resulting set of individuals is considered for survival selection: a number $p = 100$ of individuals is selected by performing p tournaments with $k = 2$.

Robots are evaluated in the environment for a period of 20 seconds. The fitness function measures the speed (cm/second) of the robot in the y axis (positive values are better). This function is defined with Eq. 2:

$$f = \frac{y_t - y_0}{t} \tag{2}$$

where y_0 is s the position of the robot in the y axis at the beginning of the evaluation period, and y_t is the position of the robot in the y axis at the end of the evaluation period.

2.4 Experimental Setup

We evolved populations of controllers using two different experimental setups:

- *All signals*: The robot controller utilizes all sensory signals available. This serves as a baseline for the alternative setup.
- *Filtered signals*: The robot controller might utilize only part of the sensory signals available because it has a mask able to filter out signals. This filtering was described in Sect. 2.2.

Populations were evolved for 150 generations and experiments were repeated independently 10 times. For the artificial neural network (ANN), the mutation and crossover probabilities were set to 0.2 and 0.8 respectively, while for the mask, these probabilities were 0.004 for mutation and 0.64 for crossover. These hyperparameters were selected based on a combination of preliminary experiments and existing literature to achieve a balance between exploration and exploitation in the evolutionary process. A summary of all experimental parameters is provided in Table 1.

Table 1. Summary of Experimental Parameters

Parameter	Value
Evaluation in seconds	20
Experiment Repetitions	10
Population Size	100
Offspring Size	100
Number of Generations	150
Crossover Probability (Neural Network)	0.8
Mutation Probability (Neural Network)	0.2
Crossover Probability (Mask)	0.64
Mutation Probability (Mask)	0.004

3 Results and Discussion

3.1 How Filtering Affects Robot Performance

We start by analyzing the effects of filtering sensory signals on robot performance. Table 2 shows the metrics of performance and sensory signal usage for the two experimental setups. Supporting the main hypothesis, filtering brought about significant performance gains for most of the morphologies tested. The progression of the searches across the generations is shown for the smallest and largest morphologies in Figs. 4 and 5: there was no significant improvement to the robot with the lowest number of joints and a great improvement to the robot with the highest number of joints.

Table 2. Speed (performance) gain and sensory signal usage of different robot morphologies. Speed, gain, and usage were averaged over the robots in the last generation of each experiment. *Speed gain* measures the speed increase, comparing the results obtained from the *all signals* experiments to those obtained from the *filtered signals* experiments: $\frac{filtered-all}{all}$. *Ratio of used signals* refers to the percentage of non-filtered sensory signals in the *filtered signals* experiment. Significance is the *p-value* of the *t-test* that compares the speed between the two experimental setups in the final generation.

Morphology	Joints	Size	Speed (all signals)	Speed (filtered signals)	Speed gain (%)	Ratio of used signals (%)	Significance
blokky	5	15	1.99	2.07	4.00	51.00	0.216
gecko	6	13	3.82	4.45	16.00	42.00	0.111
squarish	6	13	2.00	2.62	31.00	38.00	0.001
snake	8	16	2.96	3.96	34.00	32.00	0.114
ant	8	17	3.22	4.31	34.00	35.00	0.007
tinlicker	8	15	2.80	2.98	7.00	39.00	0.476
spider	8	17	5.55	7.00	26.00	35.00	0.013
babya	8	16	3.70	4.32	17.00	29.00	0.163
park	8	14	2.50	2.69	8.00	25.00	0.147
stingray	9	15	2.25	2.74	22.00	35.00	0.014
insect	9	12	3.16	4.26	35.00	36.00	0.016
queen	9	14	2.76	3.14	14.00	39.00	0.032
garrix	10	15	2.37	3.28	38.00	32.00	0.000
pentapod	10	15	2.87	4.34	51.00	39.00	0.001
babyb	10	21	4.15	5.71	38.00	27.00	0.051
ww	10	15	3.72	5.29	42.00	30.00	0.013
zappa	11	15	3.85	5.35	39.00	34.00	0.016
penguin	12	19	7.43	10.04	35.00	31.00	0.063
linkin	12	15	3.03	6.13	103.00	26.00	0.000
longleg	12	15	2.44	4.03	66.00	26.00	0.004
turtle	13	20	1.82	2.64	45.00	27.00	0.016
salamander	14	25	2.58	3.98	54.00	26.00	0.027

We inspect performance further by correlating these different metrics to verify the relationship between performance, sensory signal usage, and the number of joints. Figure 6 shows that the usage of sensory signals is negatively correlated with performance gain and the number of joints (Fig. 6-a and 6-b, respectively). In other words, there was a greater performance gain when more filtering was applied, and evolution applied more filtering to morphologies with more degrees of freedom (more joints). Additionally, the performance gain is positively correlated with the number of joints (Fig. 6-c): morphologies with more degrees of freedom had more performance gain. These different correlations suggest that while filtering benefits performance for most of the morphologies, this benefit intensifies when robots have more joints. Importantly, while a higher number of joints also means a larger body size, there is a weak relationship between size and performance gain (Fig. 6-d) compared to the relationship between number of joints and performance gain (Fig. 6-c), corroborating the idea that the gains are related to greater degrees of freedom.

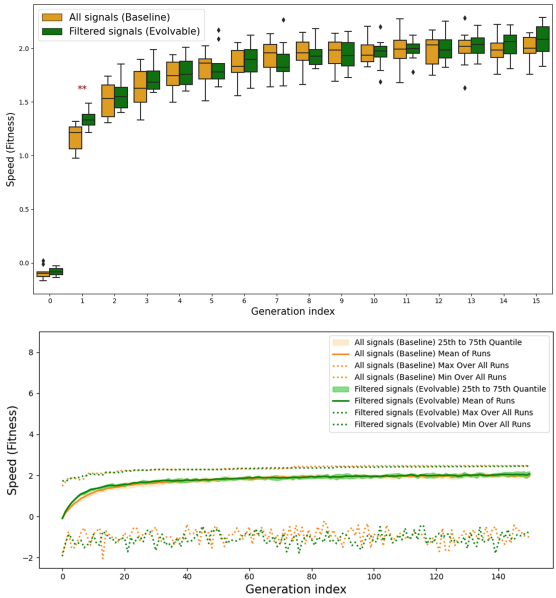


Fig. 4. Blokky - morphology with the lowest number of joints.

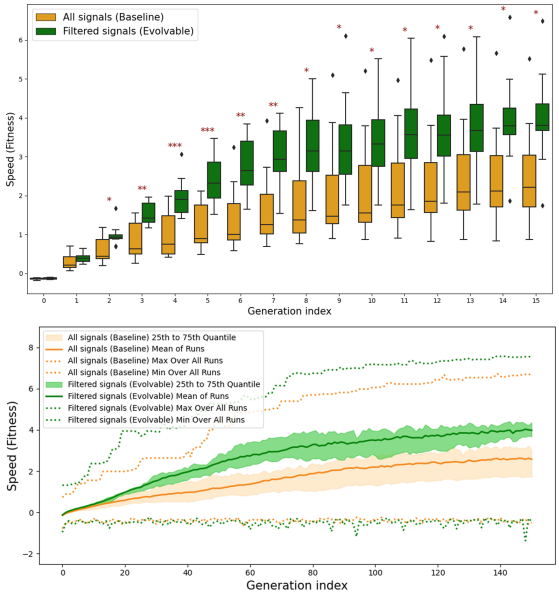


Fig. 5. Salamander - morphology with the highest number of joints.

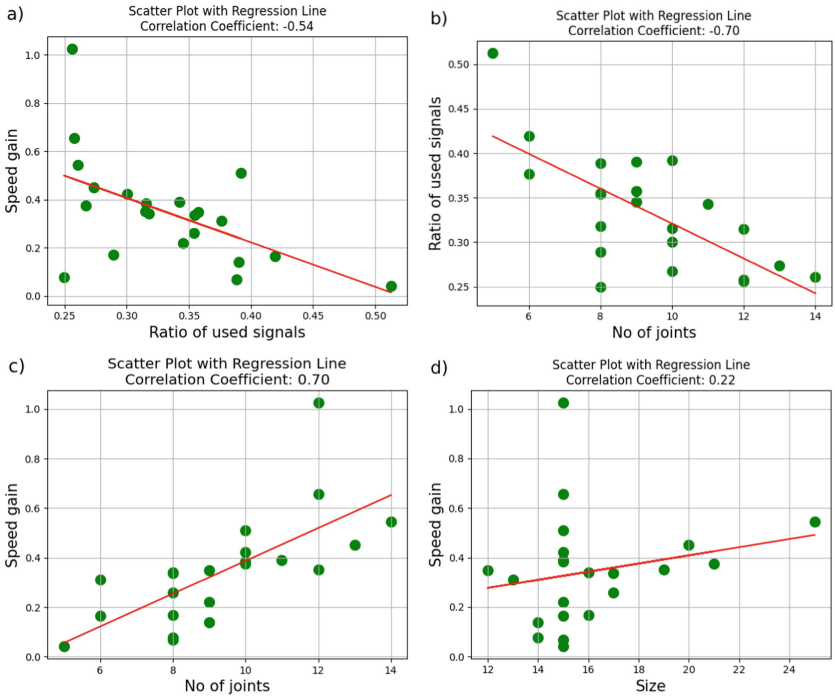


Fig. 6. Correlations between different metrics. a: negative correlation between performance gains and the ratio of used sensors. b: negative correlation between the number of joints and the ratio of used sensors. c: positive correlation between performance gains and the number of joints. d: weak positive correlation between performance gains and size of robots. Each dot regards each of the 22 morphologies. The different metrics are described in the caption of Table 2, where this same data is presented in more detail.

3.2 Impact of Oscillators on Proprioceptive Signal Filtering

Because the controllers utilized a combination of proprioceptive sensory signals with rhythmic patterns generated by oscillators, it is difficult to isolate how much of the gait success was derived from the rhythmic patterns and how much was derived from proprioceptive sensing. Considering that rhythmic patterns are commonly and successfully used to control robot locomotion [22], it is sensible to wonder if the rhythmic signals were responsible for gait success while proprioceptive signals were actually detrimental. This could have happened, for instance, because of determined flaws in the chosen architecture and algorithmic parameters. In this hypothetical scenario, reducing proprioceptive sensing could improve performance simply because proprioception does not help at all. Therefore, we conducted control experiments in which the oscillators were not included in the controller for the morphologies with the lowest and highest number of joints.

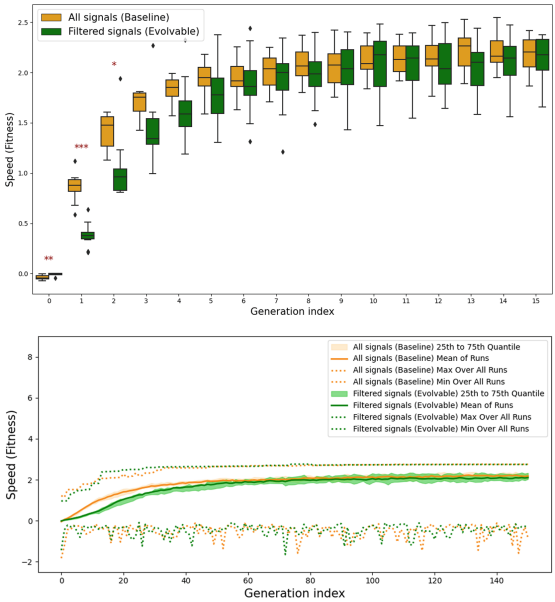


Fig. 7. Blokky without oscillators.

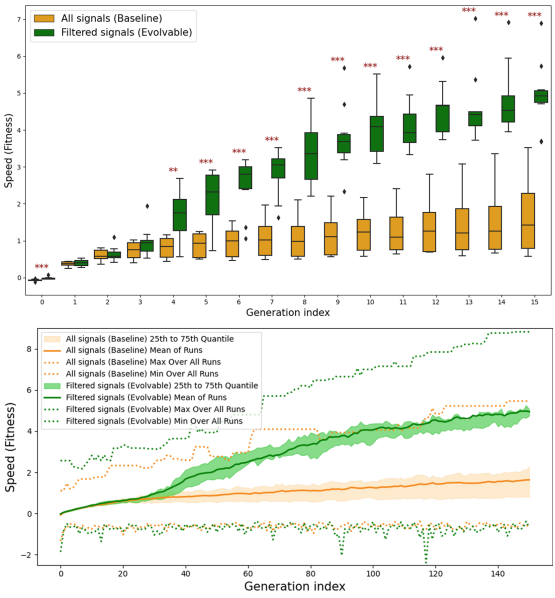


Fig. 8. Salamander without oscillators.

Results of these control experiments showed that the effects observed in the main experiments remain present: filtering was either neutral (Blokky) or beneficial (Salamander) even when the controller relied only on proprioceptive sensory signals (Figs. 7 and 8).

Beyond serving as a control, these results showed that the choice of not filtering proprioceptive sensory signals can be even more detrimental when controllers do not contain oscillators. While the speed of the salamander remained around 4cm/s in the setup that allows filtering, the speed dropped from around 2cm/s to around 1cm/s in the setup without filtering.

3.3 Sensory Signal Filtering Analysis

We analyzed the filtering evolved for all of the different body morphologies to verify if there was any pattern, e.g., a relationship between morphological traits and filtering. Nevertheless, we did not observe any patterns of filtering. Figure 9 shows the gecko morphology as an example: the diagram uses colors of variable intensity to indicate more or less usage of each sensor. The diagram includes all individuals in the final generation of all independent runs.

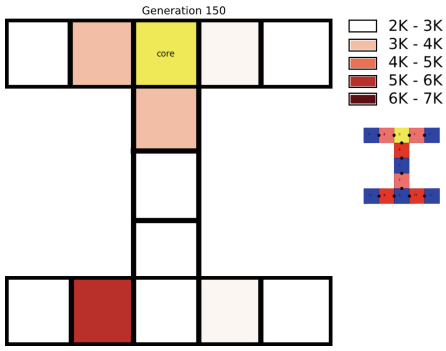


Fig. 9. Evolved pattern of used signals for the gecko morphology in the final generation. A higher number means more used (non-filtered) signals.

4 Conclusion

This study addressed the impact of filtering proprioceptive signals usage when evolving robot controllers. Considering that adding unnecessary variables to a model is commonly detrimental in traditional machine learning, the posited hypothesis intended to verify if this issue remains present in a context where the variables describe the robot itself.

The hypothesis proved partially true because applying proprioceptive sensing in a limited number of joints was more effective than its application across all

joints for most of the tested morphologies. In some cases, utilizing a limited number of joints was not more effective but equally effective. In general, robots with more degrees of freedom benefited more from sensory signal filtering – this was surprising because a robot with more degrees of freedom can produce more intricate motion and, therefore, might be expected to need more fine-tuning (more proprioceptive sensing). Finally, the detrimental effect of using too many proprioceptive signals was more severe when the controller did not include oscillators. One possible explanation for this is that oscillatory patterns reduce the dependency of the controller on the proprioceptive apparatus.

These results illuminate the potential of strategic sensor signal filtering, leading to a more effective and efficient evolutionary search. Filtering sensory signals decreases the risk of information overload, allowing the controller to make more accurate decisions. One relevant aspect driving these improvements is the concept of tailored sensory strategies [26]. The controllers can focus on the most relevant data by customizing the sensory filtering to each specific body configuration, enhancing decision-making. This approach to sensory filtering shares conceptual similarities with the feature selection process in machine learning [5,33], particularly in the context of preventing over-fitting and noise reduction. Although our method does not focus on speeding up training - remember that we control the number of parameters - it aligns with the fundamental goals of feature selection. In machine learning, feature selection aims to enhance model performance by choosing the most relevant data inputs. This is analogous to selectively filtering sensory signals in robotic systems. Such a parallel suggests a useful intersection between evolutionary robotics and machine learning, where techniques used in one domain can offer valuable insights and advancements in the other. A brief video summary of our study can be found at link online[1].

To conclude, the present study opens new avenues for robot design and control, demonstrating the potential benefits of proprioceptive sensory signal reduction. One limitation of the study is that the morphologies utilized were pre-determined. Future work should investigate if the observed effects are sustained when evolving the body morphology together with the controller and sensory signal usage. Moreover, we do not know if the observed effects would be the same for different tasks and environments. Finally, considering that the network topologies were evolvable in the baseline and alternative experiments, in both cases it was possible for evolution to prevent or weaken connections regarding detrimental signals. Nevertheless, the inclusion of a filtering mask in the alternative experiment proved more effective, though the reason for this is not yet fully understood.

[1] https://youtu.be/iwezrs7BcZI.

References

1. Auerbach, J., et al.: RoboGen: robot generation through artificial evolution. In: Artificial Life Conference Proceedings, pp. 136–137. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA Journals-Info ... (2014)
2. Bongard, J.C.: Evolved sensor fusion and dissociation in an embodied agent. In: Proceedings of the EPSRC/BBSRC International Workshop Biologically-Inspired Robotics: The Legacy of W. Grey Walter, pp. 102–109 (2002)
3. Bongard, J.C., Pfeifer, R.: A method for isolating morphological effects on evolved behaviour (2002)
4. Bongard, J.C., Pfeifer, R.: Evolving complete agents using artificial ontogeny. In: Hara, F., Pfeifer, R. (eds.) Morpho-functional Machines: The New Species. Springer, Cham (2003). https://doi.org/10.1007/978-4-431-67869-4_12
5. Cai, J., Luo, J., Wang, S., Yang, S.: Feature selection in machine learning: a new perspective. Neurocomputing **300**, 70–79 (2018)
6. Chervenski, P., Ryan, S.: Multineat, project website (2012). https://www.multineat.com
7. Craig, A.D.: Interoception: the sense of the physiological condition of the body. Curr. Opin. Neurobiol. **13**(4), 500–505 (2003)
8. D'Amario, V., Srivastava, S., Sasaki, T., Boix, X.: The data efficiency of deep learning is degraded by unnecessary input dimensions. Front. Comput. Neurosci. **16**, 760085 (2022)
9. De Carlo, M., Zeeuwe, D., Ferrante, E., Meynen, G., Ellers, J., Eiben, A.E.: Influences of artificial speciation on morphological robot evolution. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 2272–2279. IEEE (2020)
10. van Diggelen, F., Ferrante, E., Harrak, N., Luo, J., Zeeuwe, D., Eiben, A.: The influence of robot traits and evolutionary dynamics on the reality gap. IEEE Trans. Cogn. Dev. Syst. **15**(2), 499–506 (2021)
11. Doncieux, S., Bredeche, N., Mouret, J.B., Eiben, A.E.: Evolutionary robotics: what, why, and where to. Front. Robot. AI **2**, 4 (2015)
12. Heinerman, J., Rango, M.: Evolution, individual learning, and social learning in a swarm of real robots. In: 2015 IEEE Symposium Series on Computational Intelligence, pp. 1055–1062. IEEE (2015)
13. Huang, H.J., Ferris, D.P.: Computer simulations of neural mechanisms explaining upper and lower limb excitatory neural coupling. J. Neuroeng. Rehabil. **7**, 1–13 (2010)
14. Jelisavcic, M., Kiesel, R., Glette, K., Haasdijk, E., Eiben, A.: Analysis of Lamarckian evolution in morphologically evolving robots. In: Artificial Life Conference Proceedings, pp. 214–221. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA Journals-Info ... (2017)
15. Kriegman, S., Cheney, N., Bongard, J.: How morphological development can guide evolution. Sci. Rep. **8**(1), 13934 (2018)
16. Kriegman, S., et al.: Scalable sim-to-real transfer of soft robot designs. In: 2020 3rd IEEE International Conference on Soft Robotics (RoboSoft), pp. 359–366. IEEE (2020)
17. Liu, C., Chen, Y., Zhang, J., Chen, Q.: CPG driven locomotion control of quadruped robot. In: 2009 IEEE International Conference on Systems, Man and Cybernetics, pp. 2368–2373. IEEE (2009)
18. Luo, J., Miras, K., Tomczak, J., Eiben, A.E.: Enhancing robot evolution through Lamarckian principles. Sci. Rep. **13**(1), 21109 (2023)

19. Luo, J., Tomczak, J., Miras, K., Eiben, A.E.: A comparison of controller architectures and learning mechanisms for arbitrary robot morphologies. In: 2023 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1518–1525. IEEE (2023)
20. Marques, V., Ursi, S., Lima, E., Katon, G.: Environmental perception: notes on transdisciplinary approach. Sci. J. Biol. Life Sci. **1**(2), 1–9 (2020)
21. Miras, K.: Exploring the costs of phenotypic plasticity for evolvable digital organisms. Sci. Rep. **14**(1), 108 (2024)
22. Pasandi, V., Sadeghian, H., Keshmiri, M., Pucci, D.: An integrated programmable CPG with bounded output. IEEE Trans. Autom. Control **67**(9), 4658–4673 (2022)
23. Phillips, A., du Plessis, M.: Towards the incorporation of proprioception in evolutionary robotics controllers. In: 2019 Third IEEE International Conference on Robotic Computing (IRC), pp. 226–229. IEEE (2019)
24. Proske, U., Gandevia, S.C.: The proprioceptive senses: their roles in signaling body shape, body position and movement, and muscle force. Physiol. Rev. **92**, 1651–1697 (2012)
25. Röfer, T.: Evolutionary gait-optimization using a fitness function based on proprioception. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 310–322. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32256-6_25
26. Shamaiah, M., Banerjee, S., Vikalo, H.: Greedy sensor selection: leveraging submodularity. In: 49th IEEE Conference on Decision and Control (CDC), pp. 2572–2577. IEEE (2010)
27. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evol. Comput. **10**(2), 99–127 (2002)
28. Stuurman, A., Weissl, O., Chiang, T.C., Zeeuwe, D.: ci-group/revolve2: 1.0.1, January 2024. https://doi.org/10.5281/zenodo.10518564
29. Talamini, J., Medvet, E., Bartoli, A., De Lorenzo, A.: Evolutionary synthesis of sensing controllers for voxel-based soft robots. In: Artificial Life Conference Proceedings, pp. 574–581. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA Journals-Info ... (2019)
30. Todorov, E., Erez, T., Tassa, Y.: MuJoCo: a physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026–5033. IEEE (2012)
31. Winfield, A.: Robotics: A Very Short Introduction. OUP Oxford, Oxford (2012)
32. Yeon, J., et al.: A sensory-motor neuron type mediates proprioceptive coordination of steering in C. elegans via two TRPC channels. PLoS Biol. **16**(6), e2004929 (2018)
33. Zhao, T., Zheng, Y., Wu, Z.: Feature selection-based machine learning modeling for distributed model predictive control of nonlinear processes. Comput. Chem. Eng. **169**, 108074 (2023)

Author Index

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15150, pp. 419–420, 2024.
https://doi.org/10.1007/978-3-031-70071-2