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

LNCS 15149

Parallel Problem Solving from Nature – PPSN XVIII

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

Springer

Lecture Notes in Computer Science **15149**

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 II

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 978-3-031-70067-5 ISBN 978-3-031-70068-2 (eBook)
https://doi.org/10.1007/978-3-031-70068-2

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 II

Bayesian- and Surrogate-Assisted Optimization

Benchmarking and Performance Measures

Aggregated Partial Hypervolumes - An Overall Indicator for Performance Evaluation of Multimodal Multiobjective Optimization Methods

Ali Ahrari[1(✉)], Ruhul Sarker[1], and Carlos A. Coello Coello[2]

[1] School of Systems and Computing, University of New South Wales, Canberra, Australia
a.ahrari@unsw.edu.au, r.sarker@adfa.edu.au

[2] CINVESTAV-IPN, Departamento de Computación, Mexico City, Mexico
ccoello@cs.cinvestav.mx

Abstract. Multimodal multiobjective optimization (MMMOO) can be perceived as the combination of multiobjective optimization (MOO) and multimodal optimization (MMO). The performance of an MMMOO method should be thus assessed from both perspectives, leading to the prevalence of dual-metric indicators in the existing literature. This study first analyzes the ideal outcome of MMMOO for informed decision-making to determine the prerequisites of a theoretically and practically sound performance indicator. Then, it critically evaluates existing indicators, especially those that intend to measure success from the MMO perspective. Subsequently, it introduces Aggregated Partial Hypervolumes (APHVs) as a novel overall parametric performance indicator that not only addresses the drawbacks of existing ones but can also reflect the relative importance of MMO for the decision-maker. Finally, a few descriptive MMMOO examples are studied to verify that the optimal population according to APHVs matches our understanding of the ideal outcome of MMMOO, taking into account the relative importance of both the MMO and the MOO perspectives.

Keywords: Performance indicator · Multimodal multiobjective optimization · Hypervolume

1 Introduction

Multiobjective optimization (MOO) aims to find a set of diverse non-dominated solutions that approximate the Pareto front (PF). These solutions reveal the trade-off between the potentially conflicting objectives of the problem. The decision-maker can then determine the best overall trade-off among the objectives to select a single solution for its implementation using an *a posteriori* decision-making approach [1].

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 3–19, 2024.
https://doi.org/10.1007/978-3-031-70068-2_1

Quite often, the availability of distinct solutions for the selected trade-off can be beneficial. Such distinct solutions provide alternatives to support a reliable decision-making process [2]. The importance of such distinct solutions has already been analyzed and highlighted for several real-world multiobjective problems, such as path planning [3], space mission design [4], distillation plant layout [5], functional brain imaging [6], and diesel engine design [7].

Multimodal multiobjective optimization (MMMOO) aims to provide such distinct solutions. The goal of MMMOO is to find the whole Pareto set (PS), even though a part of the PS can represent the whole PF. Even solutions that are slightly dominated can be of interest [2]. MMMOO can be perceived as the integration of multimodal optimization (MMO) [8] with MOO, two relatively well-studied fields that can help to advance the knowledge in the less-establish and more complex field of MMMOO [2]. Evaluation of an MMMOO method requires assessing it from both the MOO and the MMO perspectives, resulting in the prevalence of dual-metric indicators in the existing literature:

- a metric that measures the success from the MOO perspective. Most studies used either hypervolume (HV) [9–11] [12], IGD [13–19], or both of them [20–23] [24,25] for this purpose.
- a metric that measures the success from the MMO perspective. Most existing studies used either IGDX ([10,12,16–19,21,25]), Pareto set proximity (PSP) [9,11,14,20] or both [13,15,22,23,26] for this purpose.

A significant drawback of dual-metric indicators arises when method A is better than method B according to one metric but worse according to the other one. In such cases, a dual-metric indicator cannot determine the superior method. Besides, existing metrics to measure the performance from the MMO perspective suffer from some theoretical shortcomings, which will be explained in Sect. 3.

To the best of our knowledge, IGDM [27] is the only overall indicator for MMMOO, which addresses some of the limitations of existing vdual-metric indicators. Nevertheless, it requires tuning a sensitive parameter. Besides, like IGD, it depends on the procedure used to generate uniformly distributed reference points on the PF, which can cause some biases in the comparison [28]. Therefore, developing other overall performance indicators that can overcome these shortcomings has been encouraged [28].

Another limitation of existing indicators is disregarding the relative importance of MMO and MOO for the decision-maker, which is referred to as *MMO-MOO trade-off* in this study. It implies that improving the performance from the MMO perspective comes at the cost of deteriorating it from the MOO perspective. There are two reasons for this claim. First, it is difficult and sometimes conflicting to efficiently address both MMO and MOO challenges at the same time since each demands certain strategies that can negatively affect the other one. This explains why MMMOO methods are not as good as well-known MOO methods when only MOO is pursued [2]. Second, for a given problem, the theoretically ideal outcome of MMMOO may have a worse HV or IGD than the ideal outcome of MOO. This means that regardless of the efficiency of the employed

method, improved diversity in solution space may necessitate some sacrifice in the diversity in the objective function space.

The MMO-MOO trade-off questions whether the added benefits of MMO can justify the decline in MOO capability. The answer to this question depends on how much the decision-maker is interested in the availability of diverse solutions, a feature of the problem that should be specified a priori. Accordingly, performance indicators should be able to reflect the relative importance of MMO for the decision-maker, a feature that is missing in existing ones.

The shortage of overall and pragmatic performance indicators is a major obstacle to the progress of this field since most developments in the field of evolutionary MMMOO rely on experimental evaluations and comparisons of heuristic and meta-heuristic methods and strategies. This study aims to mitigate this shortcoming by introducing a novel performance metric that can reliably measure the success of MMMOO methods. The contributions of this study are as follows:

- It analyzes different potential outcomes of MMMOO to determine prerequisites of a theoretically and practically sound performance indicator.
- It scrutinizes existing and popular performance indicators for MMMOO.
- It introduces an overall parametric indicator, called Aggregated Partial Hypervolumes (APHVs) to address the shortcomings of the existing ones.
- It analyzes APHVs on some distinct examples to confirm that indications of APHVs match our understanding of the optimal outcome of MMMOO for informed decision-making.

The rest of this study is organized as follows. Section 2 analyzes some potential outcomes of MMMOO. Section 3 reviews and analyzes relevant performance metrics for MMMOO. Section 4 introduces APHVs. Section 5 designs descriptive examples to study APHVs. Finally, our conclusions are drawn in Sect. 6.

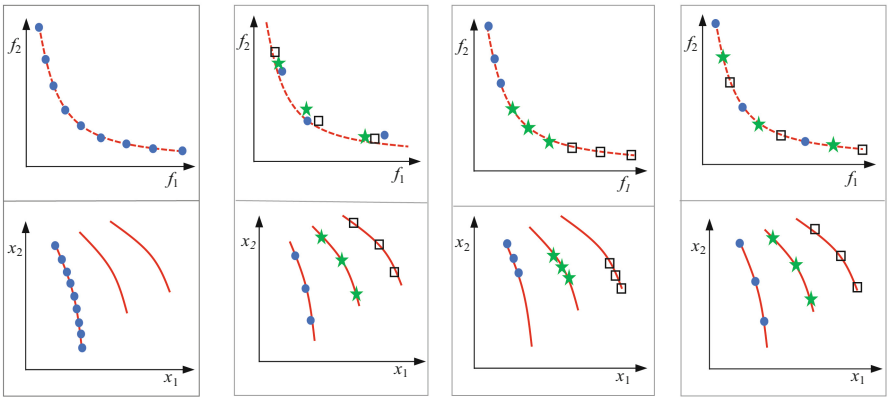
2 Qualitative Analysis of Potential MMMOO Outcomes

Figure 1 illustrates a typical MMMOO problem, in which the PS consists of three regions. Each of these regions can represent the whole PF. Four cases for the final population are considered:

- The population in Case I (Fig. 1a) has the ideal outcome from the MOO perspective, but a poor one from the MMO perspective since two regions of the PS have not been detected at all.
- In Case II (Fig. 1b), the population could detect all three regions and for each region, it has provided the three most important solutions, i.e., those that maximize the HV or IGD of that subpopulation. Such an outcome has generally been used in the MMMOO literature to represent a simple MMMOO problem where solutions from different PS regions (PSRs) map to the same point on the PF (e.g., in [11,14,25,29–31]). It provides the best approximation of individual regions of the PS. Once a trade-off among the objectives has

been selected, three distinct solutions are available for the decision-maker. However, we argue that this is not the ideal outcome when both the MOO and the MMO perspectives are important because all nine solutions almost map to three points on the PF, resulting in an inferior performance from the MOO perspective when compared to the population in Fig. 1a. Population in Case II is superior to that of Case I only if the MMO perspective was quite important for the decision-maker.

– In Case III (Fig. 1c), the population has nicely approximated the PF and detected all three regions of the efficient set. This outcome is indeed superior to the one in Case I; however, it is still not the ideal outcome since solutions from each part of the PF belong to one particular region of the PS. Once the decision-maker specifies the desired trade-off among the objectives, there is limited diversity in the available solutions.
– Case IV (Fig. 1d) shows the pragmatic ideal outcome of MMMOO when performance form both the MOO and the MMO perspectives is important. Convergence and diversity in the objective space are ideal. For each part of the PF, there are solutions from different regions of the PS, and by a slight deviation from the selected trade-off, three distinct solutions are available for the decision-maker.



(a) case I (b) case II (c) case III (d) case IV

Fig. 1. Potential outcomes of MMMOO. The problem has three distinguishable PSRs.

3 Critical Assessment of Existing Indicators

Hypervolume (HV) [32] and Inverted Generational Distance (IGD) [33] are two of the most popular metrics for assessing performance of MMMOO methods

form the MOO perspective. Given a set of solutions $\mathbb{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|\mathbb{X}|}\}$ with normalized objective values $\mathbf{0}$ $\mathbb{F} = \{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_{|\mathbb{X}|}\}$, HV is the size of the space that is dominated by solutions in $\mathbb{F}$ and dominates a reference point $\boldsymbol{r}$. vThe ideal and nadir points have normalized objective values of $\mathbf{0}$ and $\mathbf{1}$, respectively, and the reference point should be slightly dominated by the nadir point, e.g. $\boldsymbol{r} = \mathbf{1.1}$, as suggested in [34]. HV is the only known unary Pareto-compliant performance indicator.

IGD is another popular performance indicator for MOO:

$$IGD(\mathbf{F}^*, \mathbf{F}) = \frac{1}{|\mathbb{F}^*|} \sum_{i=1}^{|\mathbf{F}^*|} \min_{\boldsymbol{f}_j \in \mathbb{F}} d(\boldsymbol{f}_i^*, \boldsymbol{f}_j), \tag{1}$$

in which $d()$ calculates the Euclidean distance between two points, and $\mathbb{F}^* = \{\boldsymbol{f}_1^*, \boldsymbol{f}_2^*, \ldots, \boldsymbol{f}_{|\mathbb{F}^*|}^*\}$ is a set of uniformly distributed reference points on PF. The main drawback of IGD is that it is not Pareto-compliant [35]. Besides, the IGD value depends on the algorithm used to generate the reference points, which may be difficult to reproduce across studies. IGD^+ [36] is an enhanced version of IGD which is weakly Pareto compliant; however, it does not resolve the challenge of sampling reference points.

IGDX [21] measures the spread of solutions in the solution space to evaluate the success from the MMO perspective. Analogous to IGD, it samples a set of uniformly distributed solutions $\mathbf{X}^* = \{\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \ldots, \boldsymbol{x}_{|\mathbb{X}^*|}^*\}$ on PS. Then, IGDX is calculated as follow:

$$IGDX(\mathbf{X}^*, \mathbb{X}) = \frac{1}{|\mathbf{X}^*|} \sum_{i=1}^{|\mathbf{X}^*|} \min_{\boldsymbol{x}_j \in \mathbf{X}} d(\boldsymbol{x}_i^*, \boldsymbol{x}_j). \tag{2}$$

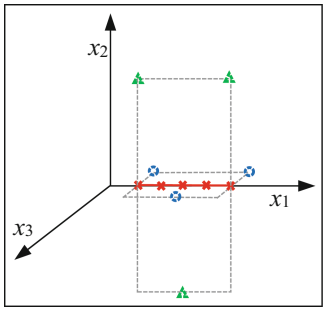


Fig. 2. Given Reference points (crosses), IGDX of population $\mathbf{P}_1$ (shown by circles) is much smaller than that of $\mathbf{P}_2$ (shown by triangles).

Like IGD, IGDX suffers from the dependency on the employed algorithm for generating the reference point. This dependency is more prominent for IGDX

since the solution space has generally a higher dimensionality than the objective space, which makes generating reference points that are uniformly distributed more challenging. Besides, directional sensitivity of the objective function(s) can result in misleading IGDX values. Figure 2 depicts such a situation in which at PS (solid line segment on x_1 axis), the fitness function is assumed to be more sensitive along x_3 than x_2. Two populations are considered: $\mathbf{P}_1$ (circles) has a slight deviation from PS along x_3 while $\mathbf{P}_2$ v(triangles) has a large deviation along x_2. The fitness values of the solutions can be identical, but population $\mathbf{P}_1$ has a much smaller IGDX, and thus should be regarded as a better one according to the IGDX metric. However, $\mathbf{P}_2$ provides a greater diversity in the solution space; therefore, it should be a better choice from the MMO perspective if diversity of solutions is desired.

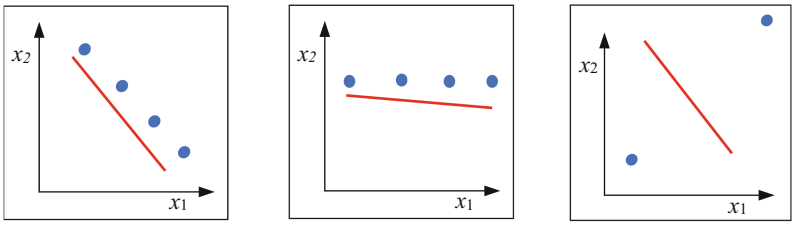
Like IGDX, PSP [9] has been frequently used to evaluate the success from the MMO perspective. It is the ratio of the Cover Rate (CR) and IGDX:

$$PSP = \frac{CR}{IGDX}, \; CR = \left(\prod_{D}^{k=1} \delta_k \right)^{1/(2D)}, \tag{3}$$

in which δ_k is the coverage of the PS for the k^{th} dimension. It is calculated as follows:

$$\delta_k = \begin{cases} 1 & \text{if } V_k^{\max} = V_k^{\min} \\ 0 & \text{if } v_k^{\min} \geq V_k^{\max} \text{ or } v_k^{\max} \leq V_k^{\min} \\ \frac{\min\{v_k^{\max}, V_k^{\max}\} - \max\{v_k^{\min}, V_k^{\min}\}}{V_k^{\max} - V_k^{\min}} & \text{otherwise} \end{cases}, \tag{4}$$

in which $v_k^{\min}$ and $v_k^{\max}$ are the minimum and maximum of the k^{th} element of the solutions in the population, and $V_k^{\min}$ and $V_k^{\max}$ are the minimum and maximum of the PS along the k^{th} coordinate.



(a) (b) (c)

Fig. 3. Actual PS (solid line) and final populations (circles) in for three different cases for problems with two decision parameters.

CR is inspired by the maximum spread [37], a less popular performance indicator for MOO. CR takes into account only the hypercube that contains

final populations, and compares it with a similarly defined hypercube for the actual PS. Comparing such limited information can easily result in misleading conclusions. For instance, PSP is sensitive to the orientation of PS. Figure 3a shows a situation in which the final population has approximated the PS quite successfully. The value of CR is about 0.9 in this case. Figure 3b represents the same problems subject to a linear rotation of the search space, and it is assumed that the population has converged to exactly the previous solutions after undergoing the exact same rotation; therefore, the approximation quality has remained unchanged. It is expected that an indicator assessing the convergence to the PS has identical values for the cases depicted in Figs. 3a and 3b; however, we have *CR=0* for the latter. Figure 3c reveals a more serious drawback of CR. The final population (two solutions here) could not approximate the PS properly, yet, we have $CR = 1$ in this case, which is the best possible outcome from a MMO perspective according to the CR indicator. Since PSP is proportional to CR, the drawbacks of CR are also present in PSP, even though combining CR with IGDX may mitigate these drawbacks to some extent.

So far, IGDM [27] is the only overall performance indicator for MMMOO [28]. Like IGD, IGDM requires a set of uniformly distributed reference points on the PF ($\boldsymbol{f}_i^*, i = 1, 2, \ldots, |\mathbf{F}^*|$). For each $\boldsymbol{f}_i^*$, it finds all the solutions in the PS that map to that reference point ($\boldsymbol{x}_{i,j}^*$s). Then, the population members are assigned to the closest reference solutions according to the minimum Euclidean distance criterion in the solution space. This means that for every $\boldsymbol{x}_{i,j}^*$, there is a subset of population members, denoted by $\mathbf{P}_{ij}$. Then, the distance between all $\boldsymbol{x}_{ij}^*$s and all the population members in $\mathbf{P}_{ij}$ in the objective space is calculated, and the smallest one is considered d_{ij}. IGDM is the mean of all these d_{ij} values.

IGDM addresses some of the drawbacks of dual-metric indicators. Most importantly, it is an overall indicator which facilitates comparison of MMMOO methods. However, it has the following drawbacks:

- It depends on the algorithm used for generating reference solutions on the PF.
- Some of $\mathbf{P}_{ij}$'s can be empty, for which IGDM sets $d_{ij} = d_{\max}$, which is a default value for reference solutions with no matching population member. This makes the relative values of IGDM sensitive to $d_{\max}$, particularly knowing that IGDM is the mean of d_{ij}s, and the mean is not a robust statistical measure.
- It cannot reflect the relative importance of MOO and MMO for the decision-maker. Based on our analysis, the ideal population with minimal IGDM resembles the one depicted in Case II (1b), which is not generally the best outcome when both MMO and MMO are important.

4 Aggregated Partial Hypervolumes

Our proposed performance measure is based on the summation of partial hypervolumes (PHVs). Let us assume that the PS, consists of K distinct regions, each of which represents the whole or a part of the PF, i.e., $\mathbf{PS} = \cup_{k=1}^{K} \mathbf{PSR}_k$,

in which $\mathbf{PSR}_k$ is the k^{th} distinct region of $\mathbf{PS}$. Let $\mathbf{P}_k$ be a subset of population $\mathbf{P}$ for which the closest PSR is $\mathbf{PSR}_k$. It is assumed that $\mathbf{P}_k$ approximates $\mathbf{PSR}_k$. PHV_k is the partial hypervolume that corresponds to $\mathbf{P}_k$, and THV is the total hypervolume that corresponds to $\mathbf{P}$. Aggregated Partial Hypervolumes (APHVs) is defined as follows:

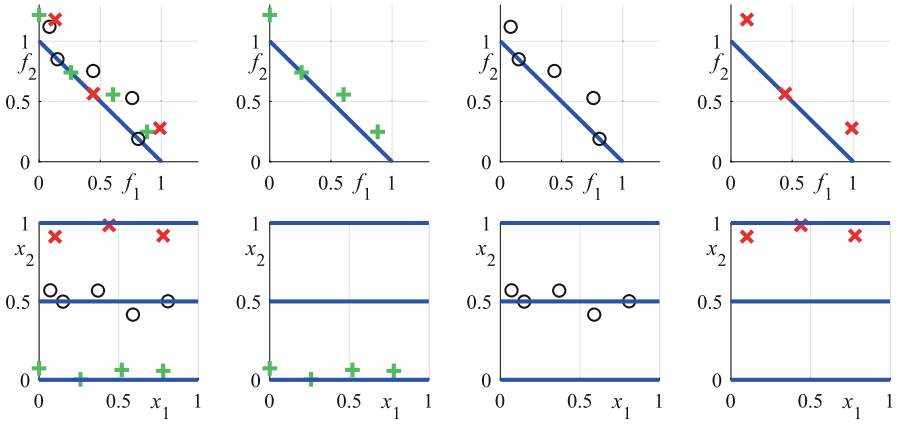
$$\begin{aligned} APHVs &= \alpha_t \, THV + (1 - \alpha_t) MPHVs, \\ MPHVs &= \frac{1}{K} \sum_{k=1}^{K} PHV_k, \; 0 \leq \alpha_t \leq 1 \end{aligned} \tag{5}$$

APHVs is the weighted average of two terms. The first term, THV, measures the quality of $\mathbf{P}$ from the MOO perspective (convergence and spread), disregarding which PSRs have been approximated. In contrast, the second term ($MPHVs$), measures how good every PSR has been approximated on average, indicating the quality of $\mathbf{P}$ from the MMO perspective. Parameter α_t specifies the relative importance of MOO and MMO for the decision-maker, i.e., the MMO-MOO trade-off. $\alpha_t = 1$ means MMO has no value for the decision maker, whereas for $\alpha_t = 0$, the importance of finding every region of the PS is maximal.

Figure 4 illustrates how APHVs is calculated for a simple bi-objective minimization problem. The PS consists of three distinct regions ($\mathbf{PSR}_1, \mathbf{PSR}_2$, $\mathbf{PSR}_3$), and each region may represent the whole PF. The population $\mathbf{P}$ has successfully approximated all these regions, and can be divided into $\mathbf{P}_1$ (pluses), $\mathbf{P}_2$ (circles), and $\mathbf{P}_3$ (crosses), based on the distance of its members to the each PSR. For this example, Fig. 4a calculates $THV = 0.557$ given the reference point $[1.1, 1.1]^T$. Figures 4b, 4c, and 4d illustrate how PHVs are calculated for each subpopulation. For this example, $APHVs = 0.557\alpha_t + (0.460 + 0.476 + 0.384)(1 - \alpha_t)/3$.

When compared with existing indicators for MMMOO, APHVs has the following advantages:

- Like IGDM, APHVs is an overall indicator, which facilitates comparison of MMMOO methods.
- Although APHVs is the weighted average of two terms, these two terms have the same nature. They are all HVs calculated in the objective space with respect to one fixed reference point.
- APHVs uses a distance metric in the solution space only to group population members based on the PSR that they are approximating. Unlike IGDX, MPHVs (or APHVs when $\alpha_t = 0$) does not use any distance metric in the solution space to quantify the performance from the MMO perspective.
- MPHVs implicitly takes the quality of solutions into account. It can deal with potential differences in the sensitivity of the objective function at different parts of the PS or to certain directions, whereas IGDX ignores such information.
- Unlike IGDX, IGD, and IGDM, APHVs does not require uniformly distributed reference points on the PS or the PF. The reference point for the



(a) $THV = 0.557$ (b) $PHV_1 = 0.460$ (c) $PHV_2 = 0.476$ (d) $PHV_3 = 0.384$

Fig. 4. Calculation of different terms in APHVs for a typical problem with three distinct PSRs, represented by solid lines. The reference for the calculation of APHVs is $[1.1, 1.1]^T$.

calculation of APHVs can be easily set (e.g. **1.1** when the objectives are normalized [35]).
- Unlike IGDM, APHVs can easily reflect the relative importance of the MMO perspective for the decision maker by setting α_t at the problem level.
- Unlike IGDM, APHVs does not have any additional and sensitive parameter to account for regions of the PS that have not been detected. These subregions simply have a zero PHV.

Quite often, the PSRs are distinguishable given the mathematical description of the PS. This is the case with most existing mathematically defined benchmark problems for MMMOO, such as those proposed in the CEC 2019 special session on MMMOO [38], in which the PS consists of either end-joined or disjoint continuous PSRs. For a PS with a complex geometry, a mathematically meaningful procedures to divide the PS into PSRs is required at the problem level, which can be regarded as a limitation of APHVs.

5 Descriptive Examples

This section designs descriptive examples to highlight distinctive features of APHVs, especially its capability to reflect the relative importance of MMO and MOO for the decision-maker. The examples are simple but distinct so that the search-space can be exhaustively searched and indications of APHVs can be visually compared with our intuition of the ideal outcome of MMMOO for informed decision-making. For the same reason, only bi-objective problems with two decision parameters ($0 \leq x_1, x_2 \leq 1$) are considered. Both objectives should be minimized, and the reference point for the calculation of APHVs is $\boldsymbol{r} = \mathbf{1.1}$.

For each problem, a set of Pareto optimal solutions of size N_P is provided. The optimal population of size n_P is exhaustively sought such that APHVs is maximized for the predefined α_t. Population members are a subset of the N_P provided Pareto optimal solutions. For each α_t, we report THV and MPHVs and their relative ratios to the maximum possible values in parenthesis. It is possible that multiple of such optimal populations exist in the problem; therefore, for each value of α_t, N_{best} indicates the number of different populations with the maximum APHVs. In such cases, the first population with the maximal APHVs is illustrated. The outcomes are distinct for the selected values of α_t.

5.1 Example 1

This example analyzes APHVs in a relatively simple but insightful scenario. The problem objectives are:

$$\begin{cases} f_1(\boldsymbol{x}) = x_1 \exp(g), f_2(\boldsymbol{x}) = (1 - x_1) \exp(g) \\ g = \sin^2(2\pi x_2) \end{cases}. \tag{6}$$

The PS consists of three disjoint regions with $x_2 = 0, 0.5, 1$, respectively, and each can represent the whole front. For this problem, $N_P = 3 \times 17 = 51$ and $n_P = 6$. Figure 5 illustrates the ideal population that maximizes APHVs for selected values of α_t. As observed:

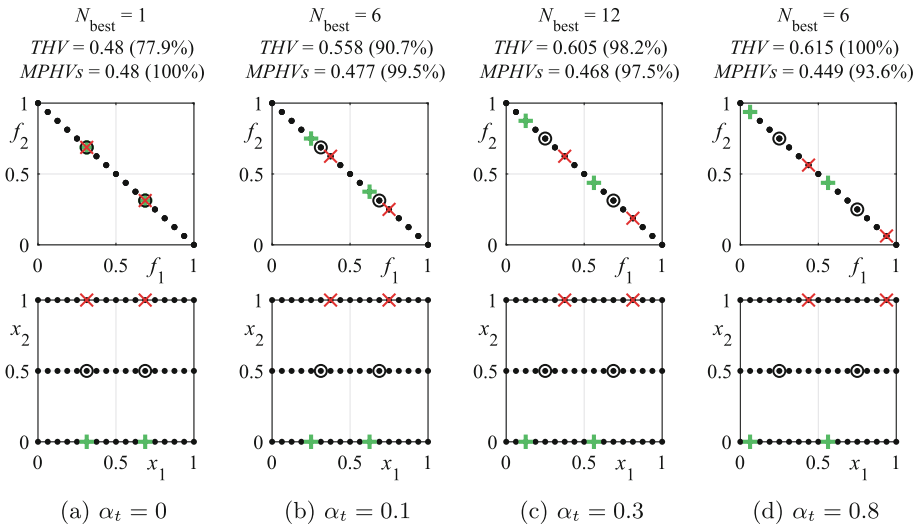


Fig. 5. The optimal MMMOO outcome for the first example according to the APHVs indicator with different values of α_t. The dots represent N_P solutions on the PS. Pluses, circles, and crosses are used to divide the population according to which region of the PS they are approximating.

- When $\alpha_t = 0$ (maximal importance of the MMO perspective), the optimal population resembles the one depicted in Fig. 1b, which is the ideal outcome from the MMO perspective. The HV is 22.1% less than the ideal case from the MOO perspective due to a lack of sufficient diversity of the whole population in the objective space. In fact, from the MOO perspective, the union of subpopulations is no better than each subpopulation alone.
- when $\alpha_t = 0.1$, subpopulations map to different points of the PF, even though they are still in two distinguishable regions of the PF. THV is now only 9.3% inferior to the ideal outcome form the MOO perspective. At the same time, MPHVs has reduced only 0.5%, which is practically negligible. There are 6 possibilities for the ideal population in this case, which are formed by swapping the relative solutions in each subpopulation.
- A greater α_t is used when the MMO perspective is less important for the decision-maker, e.g, when the formulated optimization problem is a more accurate model of the actual problem. A better approximation of the whole PF thus becomes more important. Comparing the plots in Fig. 5 demonstrates that APHVs nicely reflects this preference, where the optimal population, according to the APHVs indicator, should have a higher diversity in the objective space to maximize THV, in exchange for a weaker approximation of individual PSRs. The former becomes less important for a greater α_t.
- The ideal outcome for $0.3 \leq \alpha_t \leq 0.99$ resembles the outcome in Fig. 1d, which has a nice balance between the importance of MMO and MOO. This shows that the indications of APHVs are robust with respect to α_t.

5.2 Example 2

In the second test problem, PS consisted of two regions, $\mathbf{PSR}_1$ and $\mathbf{PSR}_2$ with $x_2 = 0$ and $x_2 = 1$, respectively. Each PSR can represent the whole front, but $\mathbf{PSR}_2$ is four times larger:

$$\begin{cases} f_1(\boldsymbol{x}) = g\, x_1/h, f_2(\boldsymbol{x}) = g\,|1 - x_1/h| \\ g = 2 + \cos(2\pi x_2 + \pi), h = (1 + Ax_2)/(A+1), A = 3 \end{cases}. \tag{7}$$

The set of available Pareto optimal solutions consists of 17 solutions on each PSR ($N_P = 2 \times 17 = 34$). For this problem $n_P = 8$. Figure 6 illustrates the optimal outcome according to the APHVs indicator for different values of α_t. As observed:

- When $\alpha_t = 0$, the population forms two equally sized subpopulations $\mathbf{P}_1$ and $\mathbf{P}_2$, which are uniformly distributed on $\mathbf{PSR}_1$ and $\mathbf{PSR}_2$, respectively, even though $\mathbf{PSR}_2$ is four times larger. Both subpopulations map to exactly the same points in the objective space to maximize their MPHVs, providing the decision-maker with the most important trade-off solutions for each PSR. Once a trade-off is chosen, two distinct solutions are available.

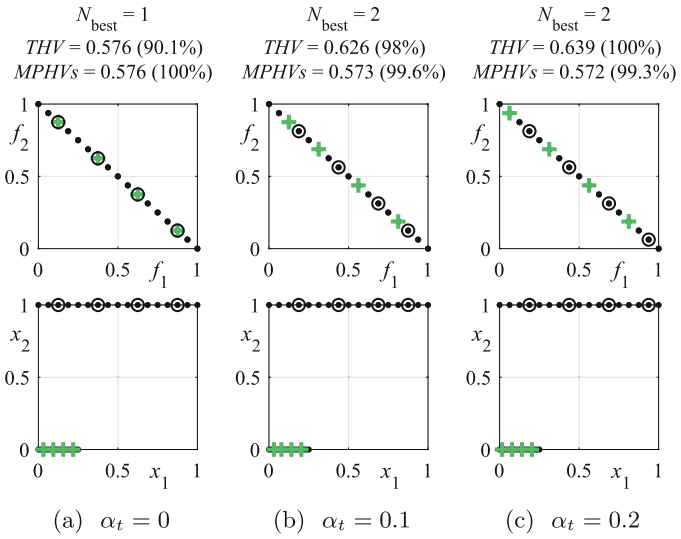
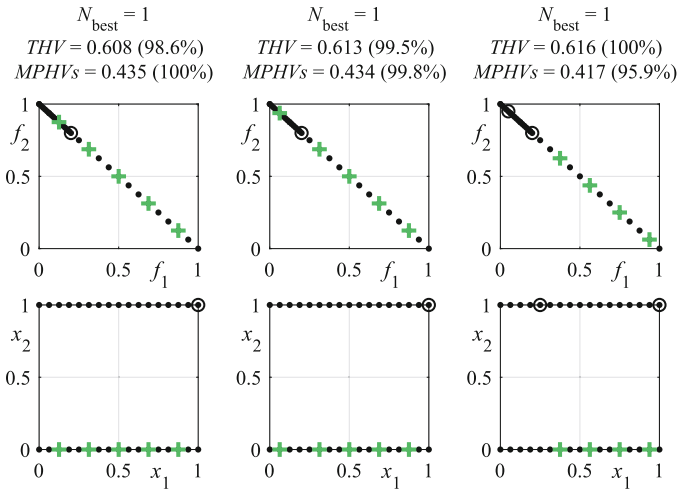


Fig. 6. The optimal MMMOO outcome for the second test problem according to the APHVs indicator with different values of α_t. The dots represent N_P solutions on the PS. Pluses and circles are used to divide the population according to the region of the PS that they are approximating.

- According to the IGDX indicator, the ideal outcome from the MMO perspective would have only one or two points on $\mathbf{PSR}_1$. The reason for this is that the number of reference points on $\mathbf{PSR}_2$ would be four times larger than the number of reference points on $\mathbf{PSR}_1$ since reference points for the calculation of IGDX are uniformly distributed on the PS. This observation reveals an intrinsic difference between IGDX and MPHVs (or APHVs when $\alpha_t = 0$): MPHVs and IGDX specify two fundamentally different populations when the ideal outcome from the MMO perspective is desired. From the decision-making perspective, the one specified by IGDX is inferior because distinct solutions might not be available for the selected trade-off.
- When α_t increases, the diversity of the optimal population in the objective space improves while the quality of the approximation of individual PSRs declines. Nevertheless, for the large range of $0.2 \leq \alpha_t \leq 0.99$, the ideal outcome does not change, indicating the robustness of the APHVs metric to the choice of α_t.

5.3 Example 3

The third example investigates a scenario which can be regarded as the opposite of the one in Example 2: $\mathbf{PSR}_1$ at $x_2 = 0$ and $\mathbf{PSR}_2$ at $x_2 = 1$ have equal lengths but the latter maps to a small region of the PF:



(a) $\alpha_t = 0$ (b) $\alpha_t = 0.2$ (c) $\alpha_t = 0.99$

Fig. 7. The optimal MMMOO outcome for the third example according to the APHVs indicator with different values of α_t. The dots represent N_P solutions on the PS. Pluses and circles are used to divide the population according to the region of the PS that they are approximating.

$$\begin{cases} f_1(\boldsymbol{x}) = g\, x_1/h, f_2(\boldsymbol{x}) = g\,(1 - x_1)/h \\ g = 2 + \cos(2\pi x_2 + \pi), h = (1 + Ax_2), \quad A = 4 \end{cases}. \tag{8}$$

2×17 uniformly distributed points are provided on the PS from which the optimal population of size $n_P = 6$, which consists of two subpopulations, $\mathbf{P}_1$ and $\mathbf{P}_2$, should be selected. Figure 7 illustrates the optimal outcome for different values of α_t. As observed:

- For $\alpha_t = 0$, the two subpopulations are not equally sized. Proper approximation of $\mathbf{PSR}_1$ is more important because the contribution of $\mathbf{PSR}_2$ to MPHVs is limited. For decision-making, detection of $\mathbf{PSR}_2$ is useful only if the decision-maker is interested in a trade-off on the upper left part of the PF. This characteristic of the problem is well-captured by the APHVs indicator. The only member of $\mathbf{P}_2$ is on the right corner of $\mathbf{PSR}_2$ to present the most important trade-off that can be offered by a solution in $\mathbf{PSR}_2$. In contrast, IGDX recommends a completely different outcome in which both subpopulations are equally sized and uniformly distributed on PSRs, disregarding the fact that all solutions in $\mathbf{PSR}_2$ are useless unless the decision-maker is interested in a small region of the PF on the top left.
- As expected, a larger α_t emphasizes more on diversity of the whole population in the objective space. Surprisingly, when $\alpha_t = 0.99$, $\mathbf{PSR}_2$ has two members in the optimal population. An extra member on $\mathbf{PSR}_2$ has improved THV

because the available points on $\mathbf{PSR}_2$ can provide a better approximation of the upper left part of PF.

6 Summary and Conclusions

This study introduced an overall indicator, called APHVs, which overcomes the discussed theoretical and practical shortcomings of existing indicators. APHVs is the weighted average of two terms: THV, which measures the success from the MOO perspective, and MPHVs, which quantifies the success from the MMO perspective. Both terms are inherently hypervolumes calculated on the objective space with respect to a fixed reference point. The weight parameter, $0 \leq \alpha_t \leq 1$, specifies the importance of the MMO perspective for the decision-maker, a feature that does not exist in the currently available indicators for MMMOO.

Our descriptive test problems presented evidence indicating that APHVs matches our understanding of the desirable outcome of MMMOO with an arbitrary MOO-MMO trade-off. Besides, the parameter α_t of APHVs can reliably reflect the relative importance of MOO and MMO perspectives when evaluating MMMOO methods. When $\alpha_t = 0$, the optimal population aims to make the best approximation of individual regions of the PS. By increasing α_t, the optimal population focuses more on providing a better approximation of the PF; nevertheless, the optimal population is not sensitive to the choice of α_t, and for a large range of α_t, the optimal population should have a good performance from both the MMO and the MOO perspectives.

When compared with existing dual-metric indicators, APHVs takes the relation between PS and PF into account when quantifying MMO success, resulting in a fundamentally different and practically more meaningful perception of the MMO success. For example, if a large region of PS maps to a small part of PF, APHVs gives less importance to that region of the PS. Besides, for APHVs, diverse solutions are those that belong to different regions of the PS, which might not necessarily be far from each other in the solution space.

The limitation of APHVs arises when PSRs cannot be easily determined given the PS; however, this case is scarce in existing benchmark problems for MMMOO. Formulating a mathematically sound procedure to divide the PS into PSRs can address this limitation, which can be the subject of future studies.

Acknowledgement. This research has been funded by the Australian Research Council Discovery Early Career Researcher Award DE230101281. Computational resources for this study were provided by the National Computational Infrastructure (NCI), which is supported by the Australian Government.

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

References

1. Purshouse, R.C., Deb, K., Mansor, M.M., Mostaghim, S., Wang, R.: A review of hybrid evolutionary multiple criteria decision making methods. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 1147–1154. IEEE (2014)
2. Tanabe, R., Ishibuchi, H.: A review of evolutionary multimodal multiobjective optimization. IEEE Trans. Evol. Comput. **24**(1), 193–200 (2019)
3. Liang, J., Yue, C., Li, G., Qu, B., Suganthan, P., Yu, K.: Problem definitions and evaluation criteria for the CEC 2021 on multimodal multiobjective path planning optimization (2020)
4. Schutze, O., Vasile, M., Coello, C.A.C.: Computing the set of epsilon-efficient solutions in multiobjective space mission design. J. Aerosp. Comput. Inf. Commun. **8**(3), 53–70 (2011)
5. Preuss, M., Kausch, C., Bouvy, C., Henrich, F.: Decision space diversity can be essential for solving multiobjective real-world problems. In: Ehrgott, M., Naujoks, B., Stewart, T., Wallenius, J. (eds.) Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems. LNEMS, vol. 634. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-04045-0_31
6. Sebag, M., Tarrisson, N., Teytaud, O., Lefevre, J., Baillet, S.: A multi-objective multi-modal optimization approach for mining stable spatio-temporal patterns. In: IJCAI, pp. 859–864 (2005)
7. Hiroyasu, T., Nakayama, S., Miki, M.: Comparison study of SPEA2+, SPEA2, and NSGA-II in diesel engine emissions and fuel economy problem. In: 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 236–242. IEEE (2005)
8. Das, S., Maity, S., Qu, B.-Y., Suganthan, P.N.: Real-parameter evolutionary multimodal optimization-a survey of the state-of-the-art. Swarm Evol. Comput. **1**(2), 71–88 (2011)
9. Yue, C., Qu, B., Liang, J.: A multiobjective particle swarm optimizer using ring topology for solving multimodal multiobjective problems. IEEE Trans. Evol. Comput. **22**(5), 805–817 (2017)
10. Ming, F., Gong, W., Yang, Y., Liao, Z.: Constrained multimodal multi-objective optimization: test problem construction and algorithm design. Swarm Evol. Comput. **76**, 101209 (2023)
11. Agrawal, S., Tiwari, A., Yaduvanshi, B., Rajak, P.: Differential evolution with nearest better clustering for multimodal multiobjective optimization. Appl. Soft Comput. **148**, 110852 (2023)
12. Wang, Y., Liu, Z., Wang, G.-G.: Improved differential evolution using two-stage mutation strategy for multimodal multi-objective optimization. Swarm Evol. Comput. **78**, 101232 (2023)
13. Sun, Y., Zhang, S.: A decomposition and dynamic niching distance-based dual elite subpopulation evolutionary algorithm for multimodal multiobjective optimization. Expert Syst. Appl. **231**, 120738 (2023)
14. Zhang, X., Liu, H., Tu, L.: A modified particle swarm optimization for multimodal multi-objective optimization. Eng. Appl. Artif. Intell. **95**, 103905 (2020)
15. Li, W., Zhang, T., Wang, R., Ishibuchi, H.: Weighted indicator-based evolutionary algorithm for multimodal multiobjective optimization. IEEE Trans. Evol. Comput. **25**(6), 1064–1078 (2021)
16. Li, W., Yao, X., Li, K., Wang, R., Zhang, T., Wang, L.: Coevolutionary framework for generalized multimodal multi-objective optimization. IEEE/CAA J. Automatica Sinica **10**(7), 1544–1556 (2023)

17. Lv, Z., Li, S., Sun, H., Zhang, H.: A multimodal multi-objective evolutionary algorithm with two-stage dual-indicator selection strategy. Swarm Evol. Comput. **82**, 101319 (2023)
18. Ji, J., Wu, T., Yang, C.: Multimodal multiobjective differential evolutionary optimization with species conservation. IEEE Trans. Syst. Man Cybern. Syst. **54**(2), 1299–1311 (2023)
19. Ding, Z., Cao, L., Chen, L., Sun, D., Zhang, X., Tao, Z.: Large-scale multimodal multiobjective evolutionary optimization based on hybrid hierarchical clustering. Knowl.-Based Syst. **266**, 110398 (2023)
20. Zhang, W., Li, G., Zhang, W., Liang, J., Yen, G.G.: A cluster based PSO with leader updating mechanism and ring-topology for multimodal multi-objective optimization. Swarm Evol. Comput. **50**, 100569 (2019)
21. Zhou, A., Zhang, Q., Jin, Y.: Approximating the set of pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. IEEE Trans. Evol. Comput. **13**(5), 1167–1189 (2009)
22. Liang, J., et al.: A clustering-based differential evolution algorithm for solving multimodal multi-objective optimization problems. Swarm Evol. Comput. **60**, 100788 (2021)
23. Yue, C., Qu, B., Yu, K., Liang, J., Li, X.: A novel scalable test problem suite for multimodal multiobjective optimization. Swarm Evol. Comput. **48**, 62–71 (2019)
24. Zhou, T., Han, X., Wang, L., Gan, W., Chu, Y., Gao, M.: A multiobjective differential evolution algorithm with subpopulation region solution selection for global and local pareto optimal sets. Swarm Evol. Comput. **83**, 101423 (2023)
25. Yang, C., Wu, T., Ji, J.: Two-stage species conservation for multimodal multiobjective optimization with local pareto sets. Inf. Sci. **639**, 118990 (2023)
26. Xiong, M., Xiong, W., Liu, Z., Liu, Y., Han, C.: A multi-modal multi-objective evolutionary algorithm based on dual decomposition and subset selection. Swarm Evol. Comput. **84**, 101431 (2023)
27. Liu, Y., Yen, G.G., Gong, D.: A multimodal multiobjective evolutionary algorithm using two-archive and recombination strategies. IEEE Trans. Evol. Comput. **23**(4), 660–674 (2018)
28. Liu, Y., Xu, L., Han, Y., Zeng, X., Yen, G.G., Ishibuchi, H.: Evolutionary multimodal multiobjective optimization for traveling salesman problems. IEEE Trans. Evol. Comput. **28**(2), 516–530 (2023)
29. Zhou, T., Hu, Z., Su, Q., Xiong, W.: A clustering differential evolution algorithm with neighborhood-based dual mutation operator for multimodal multiobjective optimization. Expert Syst. Appl. **216**, 119438 (2023)
30. Zou, J., Deng, Q., Liu, Y., Yang, X., Yang, S., Zheng, J.: A dynamic-niching-based pareto domination for multimodal multiobjective optimization. IEEE Trans. Evol. Comput. (2023). https://doi.org/10.1109/TEVC.2023.3316723
31. Zhang, J., Zou, J., Yang, S., Zheng, J.: An evolutionary algorithm based on independently evolving sub-problems for multimodal multi-objective optimization. Inf. Sci. **619**, 908–929 (2023)
32. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)
33. Coello Coello, C.A., Reyes Sierra, M.: A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 688–697. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24694-7_71

34. Ishibuchi, H., Imada, R., Setoguchi, Y., Nojima, Y.: How to specify a reference point in hypervolume calculation for fair performance comparison. Evol. Comput. **26**(3), 411–440 (2018)
35. Ishibuchi, H., Imada, R., Masuyama, N., Nojima, Y.: Comparison of hypervolume, IGD and IGD$^+$ from the viewpoint of optimal distributions of solutions. In: Deb, K., et al. (eds.) EMO 2019. LNCS, vol. 11411, pp. 332–345. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12598-1_27
36. Ishibuchi, H., Masuda, H., Tanigaki, Y., Nojima, Y.: Modified distance calculation in generational distance and inverted generational distance. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) EMO 2015. LNCS, vol. 9019, pp. 110–125. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15892-1_8
37. Tsou, C.-S., Fang, H.-H., Chang, H.-H., Kao, C.-H.: An improved particle swarm pareto optimizer with local search and clustering. In: Wang, T.-D., et al. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 400–407. Springer, Heidelberg (2006). https://doi.org/10.1007/11903697_51
38. Liang, J.-J., Qu, B., Gong, D., Yue, C.: Problem definitions and evaluation criteria for the CEC 2019 special session on multimodal multiobjective optimization. Comput. Intell. Lab. 353–370 (2019). Zhengzhou University, Technical Report 201912

Empirical Analysis of the Dynamic Binary Value Problem with IOHprofiler

Diederick Vermetten[1(✉)], Johannes Lengler[2], Dimitri Rusin[3], Thomas Bäck[1], and Carola Doerr[3]

[1] LIACS, Leiden University, Leiden, The Netherlands
d.l.vermetten@liacs.leidenuniv.nl
[2] ETH Zürich, Zürich, Switzerland
[3] Sorbonne Université, CNRS, LIP6, Paris, France

Abstract. Optimization problems in dynamic environments have recently been the source of several theoretical studies. One of these problems is the monotonic Dynamic Binary Value problem, which theoretically has high discriminatory power between different Genetic Algorithms. Given this theoretical foundation, we integrate several versions of this problem into the IOHprofiler benchmarking framework. Using this integration, we perform several large-scale benchmarking experiments to both recreate theoretical results on moderate dimensional problems and investigate aspects of GA's performance which have not yet been studied theoretically. Our results highlight some of the many synergies between theory and benchmarking and offer a platform through which further research into dynamic optimization problems can be performed.

Keywords: Evolutionary Algorithms · Benchmarks · Dynamic Environment · Dynamic Binary Value

1 Introduction

Evolutionary and genetic algorithms (EAs, GAs) are an important family of randomized optimization heuristics. In order to better understand the behavior of these algorithms, we should take advantage of the wide range of perspectives from which they have been studied, including theoretical runtime analysis, systematic benchmarking studies, and practical experience. While each of these domains offers its own viewpoint, these should not stay isolated but rather benefit from and strengthen each other.

In the last years, there has been a substantial amount of theoretical work on two seemingly unrelated situations: optimization of monotonic functions [8], and optimization in dynamic linear environments [5,13] like dynamic binary value [11]. For both of these topics, theoretical analysis shows that even among simple EAs, performance can dramatically differ. Hence, these situations have high discriminative power between different EAs. To give a flavour of the theoretical results, we highlight two specific results here:

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 20–35, 2024.
https://doi.org/10.1007/978-3-031-70068-2_2

1. The $(1+1)$-EA with mutation rate $1/N$ finds the optimum of the dynamic binary value environment in time $O(N \log N)$, while the same algorithm with mutation rate $2/N$ needs exponential time [13].
2. The $(1+1)$-EA with mutation rate $1/N$ finds the optimum of every monotonic function in time $O(N \log^2 N)$ [10], while the $(\mu+1)$-EA with the same mutation rate and large population size μ needs exponential time on some hard instances [15].

These results illustrate the strengths, but also the limitations of theoretical analyses. They can identify interesting differences between similar algorithms, and may be able to prove results on whole classes of benchmarks like all monotonic functions. Moreover, they are able to provide a deep understanding of the given situation. On the other hand, they are often limited to rather simple algorithms, and to asymptotic statements like "$O(N \log N)$" or "exponential". Thus they leave important gaps that can be filled with systematic benchmarking studies.

In this paper we provide such a benchmarking study for two aforementioned situations of monotonic and of dynamic linear environments. To this end, we have several contributions:

- We develop practical variants of the theoretical benchmarks that have been used for runtime analyses. This requires non-trivial modifications, as the theoretical variants are not suited for efficient evaluations, see Sect. 1.2.
- We have integrated them into IOHexperimenter [22], a module of the open-source IOHprofiler [4] project that allows general access to the new benchmarks in a framework where they can be easily used to test other iterative optimization algorithms. Part of this integration is a generic extension of IOHexperimenter to support dynamic environments.
- We run a systematic evaluation of a large class of algorithms on the new benchmarks, identifying the parameters which are generally most crucial for the performance on these benchmarks.
- We investigate several of the theoretical results in a wider context, asking how big the effects are for small dimensionality N, how sensitive they are for other parameter settings and how well they transfer to modifications of the algorithms.

Since this paper brings together many different benchmarking aspects, we discuss them in more detail in the following sections.

1.1 Dynamic Environments

As mentioned, we have extended the IOHprofiler framework [4] to support dynamic environments, i.e., environments in which the fitness function f can change over time. In our experiments, whenever the environment changes, we re-evaluate the population with respect to the new environment, to avoid comparing fitnesses from different environments with each other. However, the implementation also supports keeping the old fitness values.

Dynamic environments can occur in various ways through a range of applications [1]. For example, part of the problem description may be uncertain or become only available over time [20,21]. Other work studies if algorithms are able to track slowly moving optima [3,6,7,16,17]. Our work is based on dynamic linear functions [5,11,13] and the dynamic binary value function [11,12], which maintain the same global optima and were motivated by shifting training data for a machine learning system, especially in the context of co-evolution [11]. This can also be seen for example as a simplified case of algorithm configuration, where different parameters contribute more on some instances than others [9].

1.2 Theory-Inspired Benchmarks

In recent years, there has been increasing interest and demand for composing a problem suite with theoretical benchmarks that are suitable for empirical testing. We provide a class of such benchmarks, based on two ideas from theory: dynamic binary value for dynamic environments and the HotTopic function as an instance of a hard monotonic function.

The dynamic binary value function DBV is based on the static linear binary value function $\mathrm{BV} : \{0,1\}^N \to \mathbb{R}; \mathrm{BV}(x) = \sum_{i=1}^{N} 2^{i-1} x_i$, which computes for a bit string x the integer encoded by x in binary representation. In the dynamic setting, we use the same set of weights, but shuffle them via a random permutation. Note that the same permutation is applied for all strings that are evaluated in the same environment. Hence, every environment is given by a linear function with positive weights. In particular, all environments share the all-one string as their common global optimum, which gives the optimization process a clear objective: it should produce solutions which are as close as possible to the invariant global optimum. Moreover, the functions are never deceptive in any coordinate: whenever we flip a zero-bit into a one-bit, then the fitness increases in all possible environments. This ensures that the benchmark is feasible and can be solved efficiently by some algorithms and parameter configurations, but not by all. This yields the high discriminatory power of the benchmark.

Practically, the BV function can not easily be implemented because the weights are so large that they cause numerical problems. We do provide an implementation via a lexicographic comparison, but since this requires an approach that is not compatible with the usual framework of fitness functions, it is less well-suited as a module to be used by other researchers. Instead, we use the observation from [11] that DBV can be obtained in the limit from dynamic linear functions where we draw all weights independently from some heavy-tailed distribution. In this work, we consider the Uniform, Power-of-two (with a maximum power) and capped Pareto distributions, and compare them to the version based on lexicographic comparison.

Moreover, our implementation leaves the freedom to either change the environment every generation, or to change it less frequently. This also gives a way to test a function similar to the HotTopic function introduced in [14] (named HotTopic in [8]). This function has served an important role as theoretical benchmark [8,15], but it is forbiddingly slow to evaluate practically. However, it

has been pointed out in [12] that HotTopic may be approximated by a dynamic linear function in which the environment changes every εn generations, for some constant ε.

Thus, we provide several practical implementations of theoretical benchmarks. We empirically validate our implementations in Sect. 3 by testing whether previous theoretical analyses can be recovered empirically, and find generally a good qualitative agreement.

1.3 Theoretical Results on the Benchmarks

Here we review very briefly the known theoretical results on the related benchmarks. For both DBV and HotTopic, the $(1+1)$-EA is very well understood. For this simple algorithm, the main parameter is the mutation rate. For both benchmarks, there is a sharp phase transition in the mutation rate χ/N. There is a constant χ_0, which is $\chi_0 = 2.13\ldots$ for DBV [12] and $\chi_0 = 1.59..$ for HotTopic [8,14]. If $\chi < \chi_0$ then the $(1+1)$-EA finds the optimum in time $O(N \log N)$, otherwise it needs exponential time in N. The same dichotomy also holds for other dynamic linear functions, where all weights are drawn independently and identically distributed. The threshold χ_0 depends on the distribution and is always larger than the threshold 1.59.. for DBV. For example, the threshold is $\chi_0 = 2$ for an exponential distribution and $\chi_0 = (2-p)/(1-p)$ for a geometric distribution with parameter p.

For HotTopic, it is known that this result generalizes to a large number of algorithms, including the $(1+\lambda)$-EA, the $(\mu+1)$-EA, the $(1+\lambda)$-fEA, the $(\mu+1)$-fEA, all without crossover, and the $(1+(\lambda,\lambda))$-GA, where for the last three the parameter χ must be substituted appropriately [8]. However, all these results come with an important caveat: they hold only sufficiently close to the optimum. This may seem like a harmless condition as usually the hardest part of optimization is close to the optimum. However, this is not generally the case for HotTopic. It was shown in [15] that for $(\mu+1)$-EA without crossover, the hardest part of HotTopic is not around the optimum, but that the algorithm instead gets stuck in a region "in distance εn" from the optimum, even though it would efficiently find the optimum if started in distance $\varepsilon n/2$ from the optimum [15]. This shows abstractly the surprising fact that progress from εn to $\varepsilon n/2$ is harder than from $\varepsilon n/2$ to 0. However, the ε could not be explicitly specified. One advantage of our benchmarking approach is that we are able to replace the abstract value of ε by some concrete numbers, see Sect. 3.

Another interesting result on HotTopic is that crossover helps dramatically close to the optimum: for every mutation rate χ there is μ_0 such that the $(\mu+1)$-GA using either mutation or crossover, each with probability 0.5, is efficient close to the optimum. It remained an open question how the behavior is further away from the optimum, where there are the opposing effects of being (counter-intuitively) in a harder region of the search space, and having the benefits of crossover.

For DBV, results other than for the $(1+1)$-EA are a bit more sparse. The benchmark has been introduced in [11]. Experimental results for a very limited

set of parameters indicated that also here the hardest part of optimization is not always next to the optimum, at least not for the $(2+1)$ EA and $(3+1)$ EA without crossover. For the $(2+1)$ EA this could also be shown formally in [12]. However, it can be ruled out that the reason is the same as for the $(\mu+1)$-EA on HotTopic, since the latter effect crucially depends on the fact that HotTopic functions maintain the order of weights for some period of time [11]. So even though there is a similar surprising behavior, the reasons must be different.

1.4 IOHprofiler

IOHprofiler [4] is a modular toolbox for benchmarking iterative optimization heuristics. It provides access to a variety of problem suites through a common interface in its IOHexperimenter [22] module. This allows for a flexible benchmarking pipeline, which includes data logging. This data can be processed and visualized in the IOHanalyzer [25] module.

By providing access to a wide variety of problem types with common logging infrastructure, IOHprofiler has enabled a range of different research questions, both from a theoretical and practical perspective. For example, recent works have shown that the star-discrepancy optimization problem provides a challenging environment for a large set of optimization heuristics [2], or created environments for competitions on several sets of submodular optimization problems [19].

2 Experimental Setup

2.1 DBV in IOHexperimenter

Our experimental setup is built upon the integration of several versions of the Dynamic BinVal problem class into the IOHexperimenter environment. To handle the dynamic nature of the problem, a way of progressing its internal state (importance of each variable), a *step*-function has to be added to the problem itself. Note that this means that in practice, the algorithm determines when the problem's state is updated, which is needed since the problem itself has no ability to detect when an evaluation belongs to a new generation. Note that the internal structure allows for problems with pre-determined update steps, but the problems we use in this paper all rely on a per-generation update controlled by the algorithm.

The Dynamic BinVal problem is based on the Binary Value problem defined via BinVal: $\{0,1\}^n \to \mathbb{R}; x \mapsto \sum_{i=1}^{n} 2^{i-1}x_i$. In the dynamic version, we draw a permutation π for each generation and compute all fitnesses with respect to the permuted weights $\mathrm{DBV}(x) = \mathrm{DBV}_{\pi(x)} = \sum_{i=1}^{n} 2^{\pi(i)-1}x_i$. This version of the problem can not be implemented directly, as the weights would become too large for the standard datatypes used within IOHexperimenter. As such, we follow the observation from [12] and draw weights independently from different distributions instead. In this context, we consider three distinct distributions:

Algorithm 1. Outline of the used Genetic Algorithm.

Inputs: Parameters from Table 1, function f to maximize
procedure GA
  Initialize Population $\mathcal{P}$ ▷ At random or with fixed distance to optimum
  while termination criterion not met **do**
    if Update required **then** ▷ Based on update frequency
      $f \rightarrow step()$
    end if
    for i in number of offspring **do**
      Perform crossover with probability p_c, creating $\mathcal{O}_i$ ▷ Uniformly
      if No crossover or Mutation after cross disabled **then**
        $n_{flip} \leftarrow max(n_{min}, Binom(\frac{\chi}{N}, N))$
        Flip n_{flip} bits to create $\mathcal{O}_i$
      end if
    end for
    Evaluate $\mathcal{O}$ (and $\mathcal{P}$ if required) ▷ Rank or evaluate based on function
    $\mathcal{P} \leftarrow (\mathcal{P}; \mathcal{O})$ ▷ Plus or comma selection
  end while
end procedure

- **Uniform**: Uniform Weights from $\mathcal{U}(0, 1)$
- **PowersOfTwo:** Uniformly from the set $\{2^i : 1 \leq i \leq 31 - \log_2(N)\}$ (to avoid overflow of the summed value)
- **Pareto**: Pareto distribution with a limited upper bound $((1 - \mathcal{U}(0, 0.75))^{-10}$, where 0.75 is chosen to avoid overflow of the summed value)

In addition to these 3 problem variants, we can also consider a 'true' version of the DBV (**Ranking**) by not implementing an evaluation, but a ranking function instead. This ranking function sorts the given individuals lexicographically according to the order of the weights, and thus does not require using the large powers of 2 directly. While this version is true to the problem, it requires any used optimization algorithm to be modified from objective-value evaluations to this ranking scheme, and thus does not give good modularity.

The integration of the 4 versions of DBV into IOHexperimenter has the additional benefit that we can utilize several transformation methods to validate algorithmic invariances. These transformations, originally proposed in the context of the 'PBO' suite, include changing the target string with a random bitstring, swapping the order of variables and translating/scaling the returned objective value. In our experiments, when we refer to a problem instance, this corresponds to a function combined with a setting of these transformations.

2.2 Used Algorithm

In this paper, we make use of a standard GA which is heavily parameterized to allow for a variety of experimental setups. An outline of this algorithm is given in Algorithm 1, while the available parameters are indicated in Table 1. For our termination criterion, we use a combination of function evaluation budget ($100 \cdot N$ by default) and optimality (to save computation time).

Table 1. Parameters available in the used GA version (top) and parameterization of the DBV functions (bottom). In our experiments, we use the default value unless stated otherwise.

Parameter	Range	Default	Notes
Mutation factor χ	$[0, N]$	1	Mutation rate is $\frac{\chi}{N}$
Number of offspring λ	$[1, \infty)$	1	
Number of parents μ	$[1, \infty)$	1	
Selection	plus, comma	plus	
Crossover rate p_c	$[0, 1]$	0	
Dynamic frequency	$[0, \infty)$	1	Number of generations
Minimum bits flipped n_{min}	$[0, N)$	0	
Mutation after crossover	yes, no	yes	Mutate crossover-result
Function Version	Rank, Uniform Power2, Pareto	Rank	Rank=lexicographic
Instance	$[0, \infty]$	$[1, 25]$	Determines transformations
Dimensionality N	$[1, \infty]$	1000	

2.3 Experimental Setup and Reproducibility

Throughout our experiments, we make use of $N = 1000$, and perform independent runs on 15 different instances for each configuration. Each run is given a budget of $1000 \cdot N$ evaluations. Our full experimental setup, including the data collection, processing and visualization, is available on our Zenodo repository [24]. This repository also includes the full datasets from our experiments in an IOHanalyzer-compatible format.

3 Results

3.1 Exploration of Used Algorithms

For our first set of experiments, we perform a coarse grid sampling of our parameter space to identify which factors most impact the algorithm's performance. For this setting, performance is measured as the percentage of correct bits after $100 \cdot N$ evaluations. To analyze the resulting data, we make use of the SHAP approach [18], which is a popular approach in the context of global explainability [23]. The resulting SHAP-values are shown in Fig. 1, sorted from most impactful (top) to least (bottom). In this figure, the color of each dot corresponds to the chosen option, e.g. for selection blue corresponds to a comma while red corresponds to a plus-strategy.

While the grid used to create Fig. 1 is rather coarse, it does provide an initial overview of the components of the algorithm which influence the selected performance measure in the most drastic ways. In particular, we notice that using a comma selection greatly deteriorates the ability to get close the the optimum, and that using a low number of parents and a large number of offspring is beneficial in this experimental setting. This generalizes the theoretical result in [15],

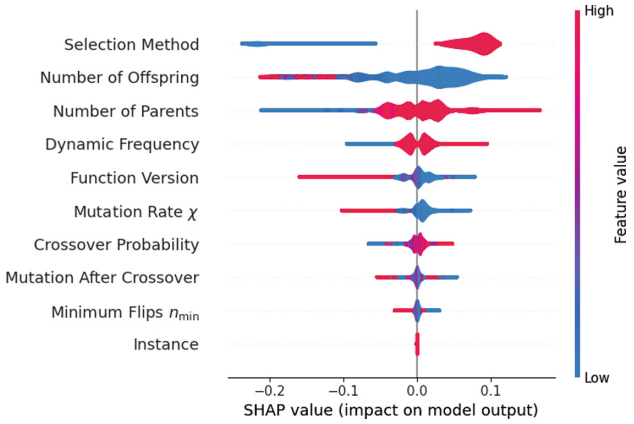


Fig. 1. Dimenionality 1000, SHAP-values for each of the varied parameter settings. For selection, red corresponds to plus-selection. For the function version, red corresponds to the rank-based version. (Color figure online)

where it was identified as a problem of the $(\mu + 1)$-EA that the population is exchanged only slowly, and the family trees within the population become deep. Those properties are generally mitigated if the ratio of offspring versus parents is large. Also of note is the fact that the version of the function has quite some impact on the algorithm's performance, while the instantiation of the function is negligible. This suggests that the used GA is invariant to the used instance transformations, but not to the used weight-distribution.

3.2 Mutation in 1+1

For our second experiment, we focus on the (1+1) GA and study the impact of mutation and the used function version on its performance. To achieve this, we vary the mutation rate between 0 and 6 (with increments of 0.1), as well as the minimum number of mutated bits between 0 and 1. This is repeated for each version of the DBV, with 25 independent runs of budget $1000 \cdot N$.

In Fig. 2, we show the number of function evaluations required to reach the optimum for each of the selected GA versions. In Fig. 3 we plot the fraction of correct bits after the evaluation budget has been exhausted. A dotted black line highlights the mutation rate of 1.6, which is the theoretical threshold for the ranking-based version, the solid blue line, in the case where it is allowed that mutation flips zero bits (the parent is duplicated). Also, theory predicts that there are no other versions with smaller thresholds [13]. Indeed, both theoretical predictions are confirmed in Figs. 2 and 3.

When comparing the different versions of DBV in Fig. 2, we observe quite a stark difference in algorithm performance between the Uniform-based DBV and the other settings. While for low mutation rates all considered versions result in the same behavior, which results from flipping 1 bit, which is always

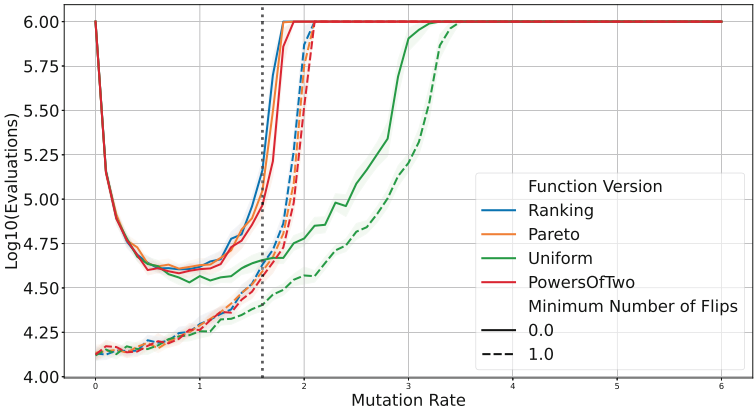


Fig. 2. Dimensionality 1000, number of evaluations (log-scaled) needed to reach optimum given different mutation rates.

accepted if it was a zero-bit and rejected otherwise, with larger mutation rates the uniform problem becomes visibly easier than the rest. This is visible both in the number of evaluations required to reach optimality, as well as in the number of correct bits after the budget threshold is reached (Fig. 3). We also see that both the PowersOfTwo and Pareto are very similar to the Rank-based DBV (with a slight preference for Pareto), so they seem to both be suitable to model the problem when function-evaluation-based approaches are required.

To understand the dashed lines, we observe that in the $(1+1)$ case, generations have no effect if the parent is duplicated (except for counting as an idle step). Hence, we should consider the number n_{flip} of flipped bits *conditioned* on flipping at least one bit. A bit counter-intuitively, the expectation $\mathbb{E}[n_{\text{flip}} \mid n_{\text{flip}} > 0]$ goes *down* by overwriting the case $n_{\text{flip}} = 0$ with $n_{\text{flip}} = 1$, which means that this change *decreases* the number of flipped bits in the conditional space. Effectively this change decreases the mutation rate, which is why the threshold is shifted to the right.

Figure 3 shows that the effect of a large mutation rate is quite dramatic. For example, for a mutation rate of $3/n$ the algorithm is only able to set less than 80% of the bits correctly after a very generous budget of $1000N$. This is still a large distance from the optimum. Even with mutation rate of $3/n$, the algorithm has a chance of roughly 15% (for $n_{\min} = 0$) to flip exactly one bit in a mutation, compared to 37% for mutation rate $1/n$. Since those are the mutations that bring most progress in this range, the poor performance can not be explained by the moderate slowdown (factor 2-3) that larger mutation rates entail on other benchmarks like OneMax. The large effect shown in Fig. 3 has not been quantified by theory before.

While the heavy aggregation of Figs. 2 and 3 shows the final performance of the considered algorithm configurations, it does not show the full behavior of the underlying runs. Since we recorded the full performance trajectory, we can

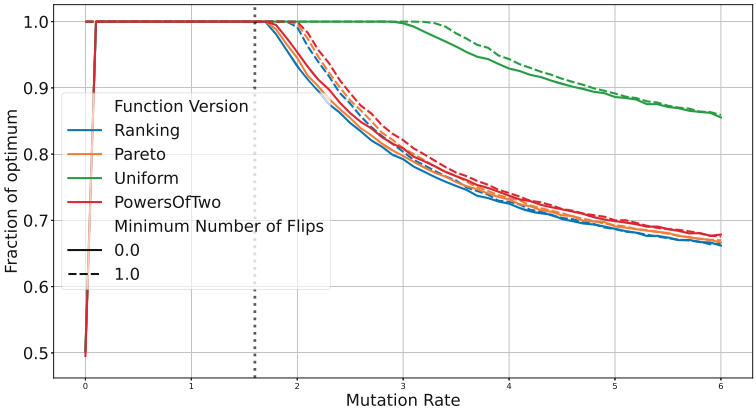


Fig. 3. Dimensionality 1000, relative number of correct bits after the budget (1000 times N) is used, given different mutation rates.

load this data into IOHanalyzer to generate more fine-grained visualization, such as ERT curves [25] illustrated in Fig. 4. The ERT gives some estimates for the expected running time under some pessimistic assumptions on the running time in which the optimum is not reached [25]. It tends to overestimate the actual running times [11]. Again, this shows how suddenly the performance drops at a given level, and how little is to be gained by increasing the computation budget.

3.3 Adding Multiple Parents: Crossover

For our next set of experiments, we investigate the effects of crossover (uniform) on the performance of the $(\mu+1)$-GA. We vary μ in $\{2, 3, 4, 8, 16\}$, and crossover rate in $\{0, 0.5, 0.9\}$. In addition to the crossover rate, we change whether an individual resulting from crossover still undergoes mutation or not. Mutation is not applied after crossover in the previous theory results.

To illustrate the impact of the different versions of crossover, we plot the number of evaluations until optimality for a given crossover rate 0.5 in Fig. 5. In this figure, we can see a transition between these two mechanisms, where for lower mutation rates it is beneficial to still perform mutation after crossover, while disabling mutation once crossover is performed allows for more stable behavior at higher mutation rates. One perspective on this is that there is something like an optimal amount of mutation. If the mutation rate is smaller, then it may be beneficial to add mutation also after crossover. If the mutation rate is larger, then it is beneficial to not apply it in every generation.

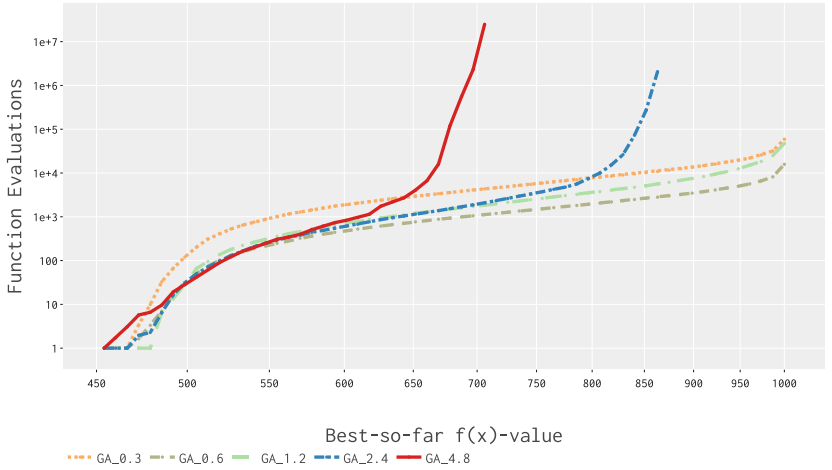


Fig. 4. Dimensionality 1000, some selected ERT curves for different mutation rates (rank-based function).

A similar trade-off can be observed in Fig. 5 for population sizes. While an increase in population size requires more function evaluations per iteration (each individual needs to be re-evaluated to perform selection), it also improves the ability of the algorithm to work with much higher mutation rates.

3.4 Difficulty of the Search Stages

A remarkable theoretical result is that the hardest region for optimization is not always around the optimum. This has been shown theoretically for the (2+1) EA on DBV [12] and for the $(\mu + 1)$ EA with large constant μ on HotTopic [15]. Notably, the underlying cause is very different in both settings. In this section we want to investigate how relevant this phenomenon for DBV in practice, which has also been shown in some narrow-focused experiments in [11], for example for the $(3 + 1)$ EA for mutation rate $2.7/N$. To this end, we seeded the algorithm with bit strings of a given Hamming distance from the optimum (fraction of correct bits). By seeding the algorithm with strings ranging from 0.5 to 0.95 and measuring the time needed to improve this fraction by 0.05, we get an overview of the relative hardness of each stage of the optimization procedure. The results of this experiment are visualized in Fig. 6.

In our experiments, we could not reproduce the finding that the hardest region was in some intermediate range. Comparing with the experiments in [11], we suspect that this effect only shows for very large budgets. Since we want to systematically explore different parameter setting in this paper, the required budget becomes forbiddingly large. We believe our results do show that the effect

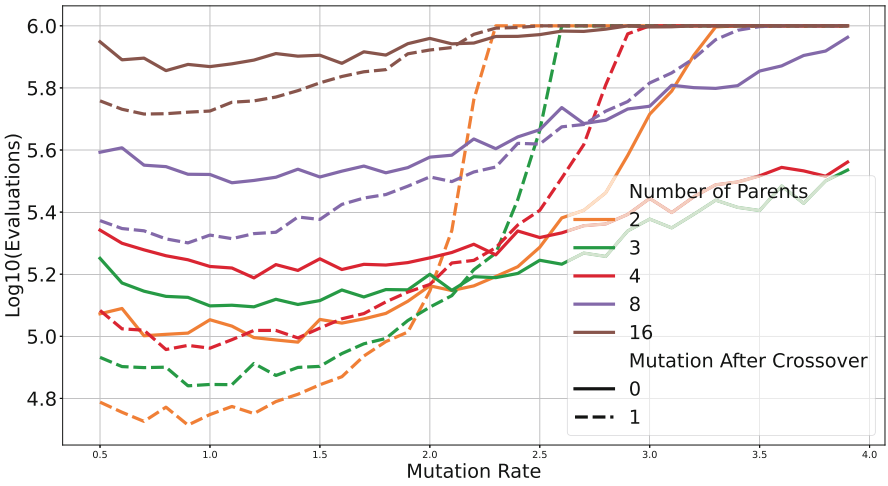


Fig. 5. Dimensionality 1000, number of evaluations (log-scaled) needed to reach optimum, on the rank-based functions, given different number of parents and mutation after crossover setting. Crossover probability is 0.5.

may not be very common in practical setting, and we leave it to future work to nail down the difference further. However, in Fig. 6 we see another interesting effect. As for the mutation rate, the number of parents has a trade-off between different regions of the search space. We generally see that larger population sizes are beneficial in "easy" regions of the search space, but are detrimental in "harder regions". This suggests that in the setting of dynamic optimization and DBV, there may not be a fixed population size that is optimal throughout all phases of the optimization process.

3.5 HotTopic: Impact of Update Frequency

While the DBV problem is defined as a fully dynamic problem where the environment changes every iteration of the algorithm, we may also change it less frequently. This has been conjectured to be related to the well-studied HotTopic problem, which is a *static* monotonic problem, but where different regions of the search space behave like different linear functions.

To illustrate the impact of the function update frequency, we again show the number of evaluations required by our $(\mu, 1)$-GA with different settings of μ. The results of this experiment, where the crossover rate is 0.9 and mutation after crossover is enabled, can be seen in Fig. 7. From this figure, we observe that for $\mu > 1$ the difficulty of the problem increases with the update frequency up until some point, after which the problem difficulty drops again. With a large

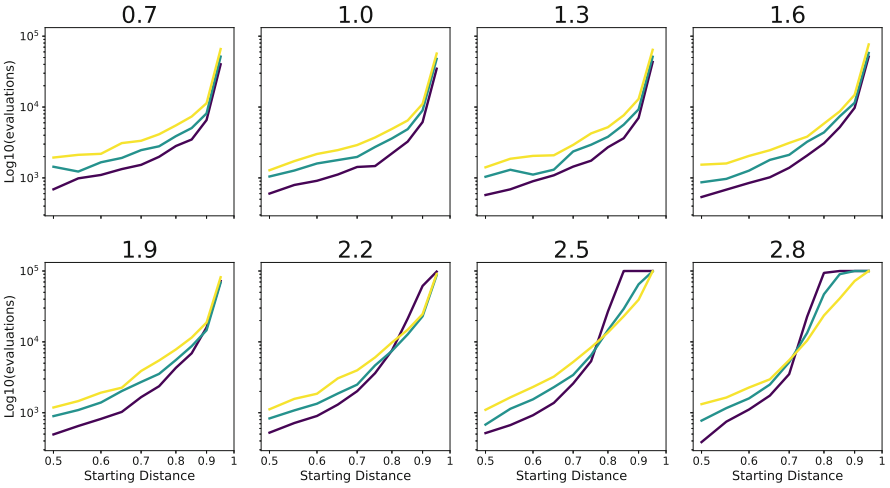


Fig. 6. Dimensionality 1000, number of evaluations (log-scaled) needed to improve the hamming distance to the optimum solution by 50, on the rank-based functions, given different number of parents (blue = 2, green = 3, yellow = 4) and mutation rate (plot titles). Crossover probability is 0. (Color figure online)

enough update frequency, the problem becomes essentially static, at which point the problem collapses to a static binary value problem. While this function is by folklore conjectured to be the hardest static linear function, the class of static linear functions is rather easy, and this includes the static binary value problem.

Interestingly, the intermediate range where the problem is hardest may correspond to HotTopic functions, where informally the "environment" changes after εN generations, where ε is a small constant. There is a remarkable analogy to the theoretical results in [15]. There it was also shown that an intermediate regime is the hardest, but not for the update frequency, but rather for the distance from the optimum. However, the two quantities may be related in this context. In the HotTopic analysis, the crucial observation was that if the weight structure is stable, then some bad family tree structure may evolve temporarily. However, in some distance from the optimum, improvements are found while this bad family tree structures dominate. Similarly, here the same bad family tree structures might evolve while the environment is constant, and new offspring may evolve while these bad structures are around. We hope that this hypothesis gives inspiration for future work, be it empirically or theoretically.

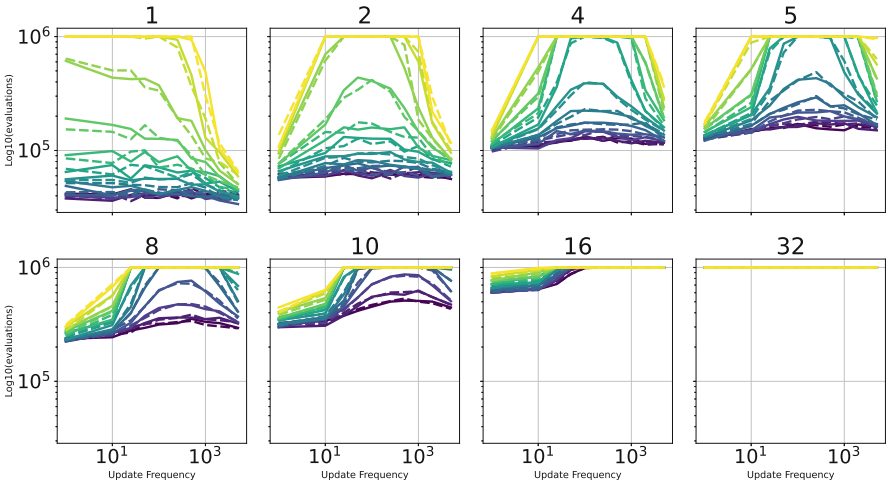


Fig. 7. Dimensionality 1000, number of evaluations (log-scaled) needed to reach optimum, on the rank-based functions, given different number of parents, mutation rates and mutation after crossover setting ($p_c = 0.9$). The x-axis corresponds to the frequency at which the weights are updated (in number of generations)

4 Conclusions and Future Work

Through a varied set of empirical experiments, we have highlighted several ways in which benchmarking can be used to build upon results from theoretical analysis in sometimes surprising ways. By integrating different versions of the Dynamic Binary Value problem into the IOHprofiler platform, we were able to compare several versions of DBV, highlighting that sampling-based versions, specifically based on the Pareto distribution, lead to similar performance as the original DBV formulation, without requiring impractically large weights.

To highlight the potential of the integration of DBV into different benchmarking pipelines, we analyzed the impact of a variety of algorithmic modifications to a standard GA, which allows for the exploration of research questions related to mutation rates, crossover methods and population sizes within the same setup. Perhaps most intriguing from a theory perspective is the non-monotone dependence on the update frequency in Fig. 7. For symmetry reasons, the update frequency makes no difference for population size $\mu = 1$. Hence, the observation that a lower update frequency makes the problem harder must be due to the structure of the population. It would be very interesting to understand how the structure of the evolving populations depends on the update frequency, and why a lower update frequency leads to more susceptible populations.

The experimental environment described in this paper can be further extended to look at different kinds of dynamic environments or different algorithmic ideas. Such new empirical studies might be used to determine interesting

aspects of these problems to further analyze theoretically, leading to a positive feedback loop between theory and practice.

Acknowledgments. This work was supported by CNRS Sciences informatiques via the AAP project IOHprofiler. It was initiated at the Dagstuhl seminar 23332.

References

1. Branke, J.: Evolutionary optimization in dynamic environments, vol. 3. Springer Science & Business Media (2012)
2. Clément, F., Vermetten, D., De Nobel, J., Jesus, A.D., Paquete, L., Doerr, C.: Computing star discrepancies with numerical black-box optimization algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1330–1338 (2023)
3. Dang, D.C., Jansen, T., Lehre, P.K.: Populations can be essential in tracking dynamic optima. Algorithmica **78**, 660–680 (2017)
4. Doerr, C., Ye, F., Horesh, N., Wang, H., Shir, O.M., Bäck, T.: Benchmarking discrete optimization heuristics with IOHprofiler. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1798–1806 (2019)
5. Janett, D., Lengler, J.: Two-dimensional drift analysis: optimizing two functions simultaneously can be hard. Theoret. Comput. Sci. **971**, 114072 (2023)
6. Kötzing, T., Molter, H.: ACO beats EA on a dynamic pseudo-Boolean function. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012. LNCS, vol. 7491, pp. 113–122. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32937-1_12
7. Lehre, P.K., Qin, X.: Self-adaptation can help evolutionary algorithms track dynamic optima. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1619–1627 (2023)
8. Lengler, J.: A general dichotomy of evolutionary algorithms on monotone functions. IEEE Trans. Evol. Comput. **24**(6), 995–1009 (2019)
9. Lengler, J.: Synergizing theory and practice of automated algorithm design for optimization (Dagstuhl Seminar 23332). Dagstuhl Rep. **13**(8), 46–70 (2024). https://doi.org/10.4230/DagRep.13.8.46
10. Lengler, J., Martinsson, A., Steger, A.: When does hillclimbing fail on monotone functions: an entropy compression argument. In: 2019 Proceedings of the Sixteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO), pp. 94–102. SIAM (2019)
11. Lengler, J., Meier, J.: Large population sizes and crossover help in dynamic environments. Nat. Comput. **23**(1), 1–15 (2022)
12. Lengler, J., Riedi, S.: Runtime analysis of the (μ+ 1)-EA on the dynamic binval function. Evol. Comput. Comb. Optim. **12692**, 84–99 (2021)
13. 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)
14. Lengler, J., Steger, A.: Drift analysis and evolutionary algorithms revisited. Comb. Probab. Comput. **27**(4), 643–666 (2018)
15. Lengler, J., Zou, X.: Exponential slowdown for larger populations: the (μ+ 1)-EA on monotone functions. In: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, pp. 87–101 (2019)

16. Lissovoi, A., Witt, C.: Runtime analysis of ant colony optimization on dynamic shortest path problems. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, pp. 1605–1612 (2013)
17. Lissovoi, A., Witt, C.: A runtime analysis of parallel evolutionary algorithms in dynamic optimization. Algorithmica **78**, 641–659 (2017)
18. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
19. Neumann, F., et al.: Benchmarking algorithms for submodular optimization problems using IOHProfiler. CoRR abs/2302.01464 (2023). https://doi.org/10.48550/arXiv.2302.01464
20. Neumann, F., Pourhassan, M., Roostapour, V.: Analysis of evolutionary algorithms in dynamic and stochastic environments. In: Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 323–357 (2020)
21. Nguyen, T.T., Yang, S., Branke, J.: Evolutionary dynamic optimization: a survey of the state of the art. Swarm Evol. Comput. **6**, 1–24 (2012)
22. de Nobel, J., Ye, F., Vermetten, D., Wang, H., Doerr, C., Bäck, T.: IOHexperimenter: Benchmarking platform for iterative optimization heuristics. CoRR abs/2111.04077 (2021). https://arxiv.org/abs/2111.04077
23. van Stein, N., Vermetten, D., Kononova, A.V., Bäck, T.: Explainable benchmarking for iterative optimization heuristics (2024). arXiv preprint arXiv:2401.17842
24. Vermetten, D., Lengler, J., Rusin, D., Bäck, T., Doerr, C.: Reproducibility files and additional figures (2024), code and data repository (Zenodo): https://doi.org/10.5281/zenodo.10964455 Figure repository (Figshare): https://doi.org/10.6084/m9.figshare.25592904
25. Wang, H., Vermetten, D., Ye, F., Doerr, C., Bäck, T.: IOHanalyzer: detailed performance analysis for iterative optimization heuristic. ACM Trans. Evol. Learn. Optim. **2**(1), 3:1–3:29 (2022). https://doi.org/10.1145/3510426, https://doi.org/10.1145/3510426, IOHanalyzer is available at CRAN, on GitHub, and as web-based GUI, see https://iohprofiler.github.io/IOHanalyzer/ for links

A Deep Dive Into Effects of Structural Bias on CMA-ES Performance Along Affine Trajectories

Niki van Stein[1(✉)], Sarah L. Thomson[2], and Anna V. Kononova[1]

[1] LIACS, Leiden University, Einsteinweg 55, 2333 CC Leiden, Netherlands
{n.van.stein,a.kononova}@liacs.leidenuniv.nl
[2] Edinburgh Napier University, Edinburgh, UK
s.thomson4@napier.ac.uk

Abstract. To guide the design of better iterative optimisation heuristics, it is imperative to understand how inherent structural biases within algorithm components affect the performance on a wide variety of search landscapes. This study explores the impact of structural bias in the modular Covariance Matrix Adaptation Evolution Strategy (modCMA), focusing on the roles of various modulars within the algorithm. Through an extensive investigation involving 435 456 configurations of modCMA, we identified key modules that significantly influence structural bias of various classes. Our analysis utilized the Deep-BIAS toolbox for structural bias detection and classification, complemented by SHAP analysis for quantifying module contributions. The performance of these configurations was tested on a sequence of affine-recombined functions, maintaining fixed optimum locations while gradually varying the landscape features. Our results demonstrate an interplay between module-induced structural bias and algorithm performance across different landscape characteristics.

Keywords: Structural Bias · benchmarking · performance analysis · algorithm behaviour

1 Introduction

In light of the rapid advancement of the field of heuristic black-box optimisation [1], a remarkable array of algorithms is now available to practitioners. The behaviour of most of these algorithms strongly depends on the settings of numerous hyperparameters, exploding the number of options further and making the choice of a well-performing algorithm configuration for a specific (real-world) problem even harder.

And yet, we still do not understand these algorithms well enough. One thing we can do is screen (a family of) algorithms or algorithm configurations against some unwanted characteristics. Although it is unrealistic to examine all settings across all characteristics, initial efforts are essential. Such screening is expensive

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 36–50, 2024.
https://doi.org/10.1007/978-3-031-70068-2_3

but it not only helps eliminate ineffective configurations but also aids in elucidating the internal dynamics that impedes performance. Modular algorithm designs [2,3], where each operator option can be selected independently of the choice for other operators, are particularly well-suited for such analyses.

While in many cases, the unwanted characteristics only manifest within specific function landscapes (and the correspondence between these landscapes and characteristics can be unknown), it is hypothesised that at least some characteristics can be assessed in general. One such aspect that has not been investigated sufficiently is structural bias (SB) [4] and especially its precise causes and influence on the algorithm's performance. SB is the algorithm's inherent limitation in locating optima in certain regions of the domain independently of the objective function's landscape. It stems from the iterative application of a limited set of algorithm operators and their interplay [4]. While some families of algorithms have been screened to some extent [2,5], no clear performance implications have been established so far. Unfortunately, even though such screening is done for very large algorithm configuration spaces, it necessarily remains limited due to the need to discretise numerous continuous hyperparameters, thus potentially overlooking some interactions. This paper is no exception (as we set the continuous parameters to a fixed value), however, its experimental design is structured to be as comprehensive as computationally feasible.

This paper aims to explore the impact of SB on algorithm performance by addressing the following questions: 1. How does the performance of structurally biased versus unbiased configurations change on sequences of functions where the landscape progressively shifts from rugged to smooth? 2. How does the location of the optima within the domain[1] of these functions affect the performance depending on the class of structural bias? The complete methodology of our investigation is summarised in Fig. 1.

2 Background

2.1 Modular CMA-ES

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [6] is a robust, state-of-the-art evolutionary algorithm used for solving non-linear, non-convex optimisation problems [1]. Central to its approach is the adaptation of a covariance matrix which determines the shape and scale of the search distribution, effectively learning the landscape of the problem space. This self-adaptive mechanism allows the algorithm to balance exploration of the search space with exploitation of known good regions, making it particularly effective in a wide range of practical applications, from machine learning parameter tuning to engineering design optimisation. The Modular CMA-ES (modCMA) [3] is a Python and C++ modular implementation of the CMA-ES algorithm and many of its variants, with module options and hyper-parameters that can be switched on and off independently of each other. In this work we investigate the full scale

[1] This paper focuses on box-constrained minimisation problems.

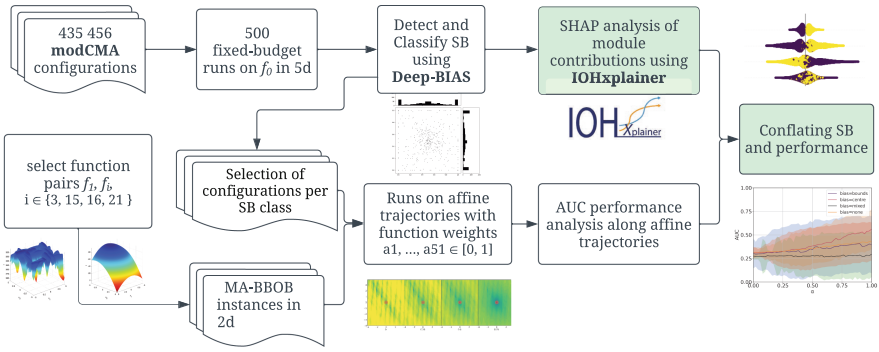


Fig. 1. The summary of the overall methodology used. Full details of all steps are provided in the corresponding sections. Green blocks highlight the contributions of this paper. (Color figure online)

of these module options, given in Table 1, leading to a total of 435 456 different CMA-ES configurations.

2.2 Structural Bias

Structural bias in iterative optimisation heuristics [4,7] refers to the tendency of certain algorithms or configurations of algorithms to favour specific regions of the search space over others, despite the absence of initial information indicating where high-quality solutions might reside. Ideally, an optimisation algorithm should explore the defined domain boundaries without preconceived preferences, allowing the data collected from sampled points to guide its progression towards areas with optimal objective function values. However, in practice, some algo-

Table 1. Considered modules of **modCMA** and their configurations.

Module name	Shorthand	Domain
Covariance adaptation	`covariance`	{false, true}
Elitism	`elitist`	{false, true}
Active update	`active`	{false, true}
Sequential selection	`sequential`	{false, true}
Base sampler	`base_sampler`	{Gaussian, Halton, Sobol}
Orthogonal sampling	`orthogonal`	{false, true}
Threshold convergence	`threshold`	{false, true}
Sample Sigma	`sigma`	{false, true}
Bound correction	`bound_correction`	{off, saturate, mirror, COTN, toroidal, uniform}
Mirrored sampling	`mirrored`	{off, mirrored, mirrored pairwise}
Recombination weights	`weights_option`	{default, equal, λ-decay}
Step size adaptation	`step_size_adaptation`	{CSA, PSR, TPA, MSR, XNES, MXNES, LPXNES}
Local restarts	`local_restart`	{none, IPOP, BIPOP}

rithms inherently exhibit a preference, such as a bias towards the centre of the domain, which can limit their effectiveness in universally discovering the best solutions across the entire feasible domain. This phenomenon, known as *structural bias*, can compromise the algorithm's ability to perform well in general situations. Detecting structural bias is hard, as the objective function and behaviour of the algorithm are always entangled. Using a special objective function (f_0), defined as a completely random fitness landscape, allows one to detect structural bias using statistical tests as introduced in [8]. There are a few related works on structural bias that either introduced a different detection method, like the generalised signature test [9], or analysed different groups of algorithms regarding SB [10–12].

2.3 SHAP

Shapley Additive Explanations (SHAP), as introduced by Lundberg et al. [13], is a popular *explainable AI* (XAI) method for attributing features in model predictions. SHAP quantifies the impact of a specific feature, f, by comparing model outputs with and without f. The difference in outputs, averaged across models, defines the SHAP value, which can be positive, negative, or zero, representing the feature's marginal contribution.

However, SHAP's application to large datasets is computationally intensive. To address this, the TreeSHAP method [14] employs tree-based model structures to streamline computations, using approximation techniques to enhance efficiency in scenarios with extensive feature sets. In this work, we use this XAI method to compute the contributions of different module settings towards specific structural bias classes. This is done by training XGboost regression models on the one-hot-encoded algorithm configurations as input and the predicted SB class label from Deep-BIAS as output. Using these models we can approximate the SHAP values of each module option per configuration. Note that SHAP is one way of approximating these contributions, there are other methods such as f-ANOVA that can also be deployed. However, SHAP has the advantage of providing also local explanations.

3 Structural Bias Classification

To assess the structural bias (SB) and in the end the interplay of structural bias with performance depending on landscape features and the location of the optima, the first step is to detect and classify structural bias per algorithm configuration. We conduct a configuration sweep for modCMA [3] using the MODCMA package in Python. In this extensive analysis, we use a full grid of all of the categorical module options in modCMA, as specified in Table 1. This resulted in a total of 435 456 configurations. For each of these configurations the population sizes are fixed to $\mu = 5, \lambda = 20$. The analysis in this paper is broader and more in-depth than previous analysis of structural bias for modCMA [12] in the sense that it contains not just a subset of modCMA module options but the complete

set of all categorical options (and a limited set of continues parameters). In addition, in this work we propose an explainable AI approach, similar to the approach used in [5], to analyze the different contributions of different module options to structural bias and to specific types of structural bias, leading to new insights. In [5], the XAI approaches was used to analyse the contributions of modules and hyper-parameters on the performance on different function landscapes, here we use it to assess the influence of modules on structural bias, and differently from the approach in [5] we one-hot-encode all categorical module options to see how each option affects the structural bias individually. We also used the approach to look at second order interactions in relation with structural bias, however, these second order interactions were marginal and we therefore do not include these results in this work.

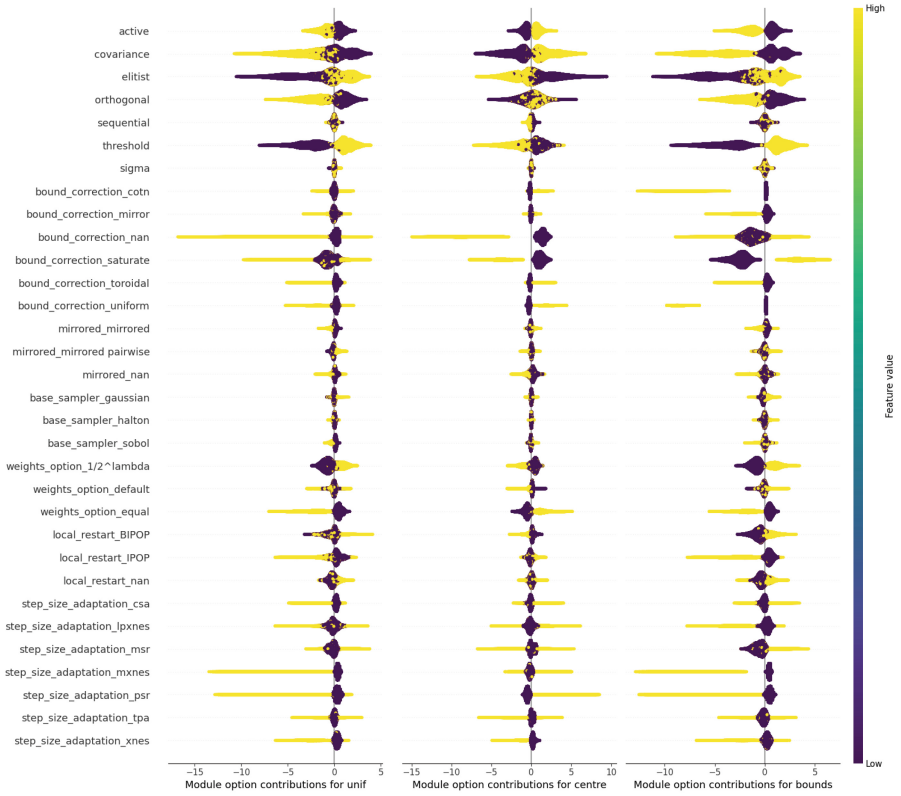
3.1 Methodology

For the assessment and classification of SB, we used the BIAS toolbox [8], available on [15]. The toolbox provides an SB detection mechanism based on the aggregation of the results of 39 statistical tests but also a Deep-learning approach [16] to detect and classify SB based on distributions of final points (found minima) of many independent runs on f_0. The three bias types we are looking at in this work are: SB towards the **centre** of the search space, towards the **bounds**, and **uniform** (no SB detected). SB is detected by first running an optimisation algorithm several times on the random objective function f_0. Here we used 100 independent runs with 10.000 function evaluations as budget per run to make the first classification of SB using the Deep-BIAS toolbox. Due to its speed and classification accuracy, we leveraged the Deep-BIAS model instead of the statistical methods in the BIAS toolbox. The Deep-BIAS method classifies the distribution of (in this case 100) final best points found by the algorithm. We do have to note that the Deep-learning model is not a perfect predictor. We therefore also verify the top 20 configurations per SB class, used later in our experiments, by visual inspection of the final point distributions.

Once we have classified each of the configurations automatically, we can use the confidence of each SB class to calculate approximate Shapley values using the TreeSHAP algorithm [14]. The calculated SHAP values for each module option per structural bias class are shown in Fig. 2. We can use these SHAP values to gain insights into which module options contribute to what kind of structural bias.

3.2 Module Contributions to SB

Based on the output of the Deep-BIAS package, most of the considered configurations of modCMA (82%) are classified as **Centre** biased, 9% as unbiased (uniform) and 5% as biased towards the bounds. The remaining fraction (3%) was classified as discretization bias, however after visual inspection, those configurations were mostly misclassified and should be either centre or unbiased and therefore we did not take them into account.



(a) No bias (b) Centre bias (c) Bounds bias

Fig. 2. SHAP values showing module contributions to (from left to right) no structural bias, centre bias and bounds bias classes, respectively. The baseline prediction of these classes are 0.094, 0.559, 0.054 respectively, meaning that a SHAP value of 0 would result in the given baseline class probability.

Given the SHAP values from Fig. 2 and taking into account the base value (mean classification confidence of each class), we see a few interesting patterns. Overall, the covariance, elitism, threshold, bound correction and step size adaptation modules mostly influence the structural bias classification. We can also observe that in general, option contributions are negatively correlated between centre SB and bounds SB, in other words, when a module option causes centre SB it lowers the probability of bounds SB and vice versa. Bounds SB and uniform (no SB) seem roughly aligned except for the bound correction methods. Let us discuss the major modules involved in centre, bounds and unbiased below.

Elitism when turned on, reduces centre SB according to the SHAP data. Since centre SB is the majority class, Elitism seems to reduce SB in general. When looking at the inner workings of modCMA and also the objective function

f_0, this could be explained due to the fact that with elitism the algorithm is more likely to get stuck in a (local) minimum on f_0 early in the optimisation run, effectively dampening the structural bias effects. Elitism by itself is however very likely not to be responsible for any structural biased behaviour. It is important to note that the SHAP values represent a correlation and not a causation.

Threshold convergence when turned on has a similar effect as elitism (though less profound). Again, threshold convergence is likely not causing any biased algorithm behaviour but amplifies (when turned off) or dampens (when turned on) the SB effects.

Bound correction saturate shows to have a large effect on bounds SB, which makes perfect sense and is in line with other research on structural bias [12]. Upon close inspection, all configurations that were classified with high confidence as bounds SB (confidence > 0.45), all used Saturate as the bound correction method.

Covariance matrix adaptation seems to play a large role in centre SB. In the context of a standard CMA-ES (so with the covariance module on), the search distribution is represented by a multivariate normal distribution. This distribution is characterized by its covariance matrix, which determines the shape and orientation of the points (solutions) that are sampled. Geometrically, the shape of this distribution resembles a hyper-ellipse. The search space, on the other hand, is typically a hyper-cube. This causes a mismatch in shapes being explored, likely leading to a structural bias towards the centre of the search space. The hyper-ellipse will naturally avoid sampling close to the edges and especially the corners of the hyper-cube because these areas are outside the maximum reach of the distribution whose radius is limited to the smaller distance from the center to an edge, rather than to a corner. This effect is amplified as the dimensionality of the space increases. In higher dimensions, the corners of the hyper-cube are exponentially further away from the centre compared to the edges. Thus, a spherical sampling distribution centred in the hyper-cube will leave vast regions in the corners significantly undersampled. This results in a higher concentration of sample points towards the centre of the search space, and relatively fewer near the boundaries and corners. It could potentially lead to suboptimal exploration of the search space, especially if the global optimum lies near the boundaries or corners of the domain.

Other module options seem to have a limited or mixed effect on structural bias in modCMA.

3.3 Limitations of Deep-BIAS and Mixed SB

While the deep-learning approach of the Deep-BIAS toolbox is very fast, and therefore allows the evaluation of 400 000+ configurations, it is not perfect. First of all, it is known [16] that the SB type 'Clusters' is often a misclassified 'Centre' SB. As a result of this, after visual inspection of (a large fraction) of the configurations that were initially classified here as Cluster SB, we found out that all of those actually belonged to the Centre class. We therefore discarded the

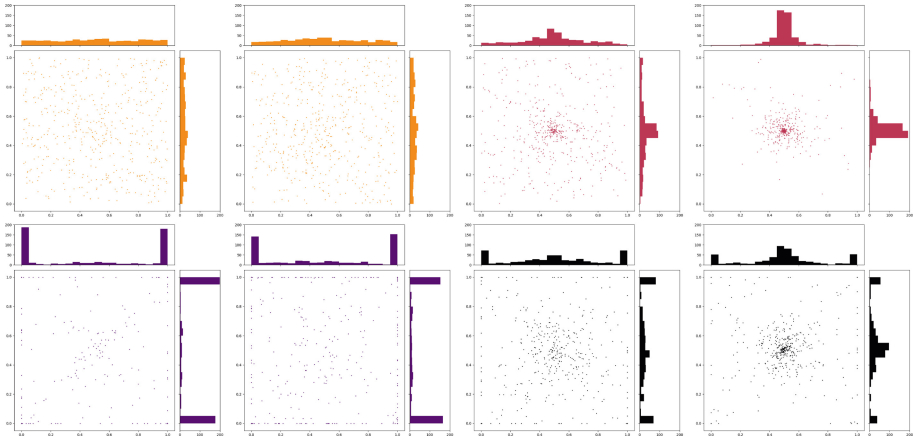


Fig. 3. Examples of the final best point distributions from CMA-ES configurations run on the SB test function f_0 in $2D$ from 500 independent runs (using different random seed) of configurations (re)classified as **no**, **centre**, **bounds** and **mixed** SB.

'Clusters' class in our analysis. In addition, after visual inspection of the configurations with highest confidence scores for each of the SB classes, we discovered a *mixed* class between centre and bounds bias. This mix of two bias directions was not discovered in earlier structural bias research and was also not taken into account when developing the (Deep)-BIAS Toolbox. For modCMA, this mixed SB behaviour seems to occur when there is a configuration that is normally centre biased and it also uses the Saturate bound correction method (inducing additional bounds SB). For further analysis of the effects of these different SB classes on performance, we decided to re-classify the top 20 configurations (sorted by confidence) for each SB class as identified by Deep-BIAS by visual inspection of the $2D$ final distributions into four SB classes: Uniform (no SB), Centre, Bounds and Mixed SB. See Fig. 3 for examples of each class of SB we took into consideration. The complete set of configurations and distributions of their final best points across runs can be found in the supplemental material [17].

4 Effects of Structural Bias on Performance

To analyse the effect of SB on algorithm performance, four distinct SB groups of algorithm configurations are evaluated on a range of affine function combinations. By gradually changing the function landscape properties, it is evaluated how the performance of these different groups changes under different conditions.

4.1 Affine Function Pairs

We include as original problems the SPHERE function (*f1* from BBOB, uni-modal) and four other BBOB functions: *f3* (separable RASTRIGIN, multi-modal), *f15* (non-

seperable Rastrign, multi-modal), *f16* (Weierstrass, multi-modal, adequate global structure), and *f21* (Gallagher's Gaussian 101-me Peaks Function, multi-modal, weak global structure). All of these are visualised in $2D$ in Fig. 1 of the supplemental material. The sphere function was chosen because we would like to *tune flatness into the affine combinations*, and the other four were chosen after visual inspection of their ruggedness (we would like to track structural bias along affine trajectories which begin at the flatness of the sphere function and *gradually become more rugged*).

The notion of affine combinations was first introduced in Dietrich and Mersmann [18]. We consider affine combinations between BBOB functions and use the generator later proposed by Vermetten *et al.* [19] which facilitates combinations of more than two functions—although we consider only pairs here. Their generator takes three objects as input in order to construct a function: 1. the desired location for the optimum, $\vec{X}_{opt}$; 2. a vector of length 24—for each of the 24 BBOB functions—indicating proportions, $\vec{W}$; and 3. a vector of length 24 indicating which instances of the BBOB functions should be used, $\vec{I}$. Of course, to obtain pairwise combinations then the proportions of 22 functions can be set to zero and the remaining two have non-zero weight. An affine combination Ξ is constructed by the generator according to the fitness *scaling functions*:

$$R_i(x) = \frac{max(\log_{10}(x), -8) + 8}{S_i} \tag{1}$$

and its inverse (to reverse back to the original fitness scale):

$$R_i^{-1}(x) = 10^{(S_i \cdot x - 8)} \tag{2}$$

S_i is a scale factor and is set at literature-recommended values [19] depending on the base function: 11.0 for *f1*, 12.3 for *f3* and *f15*, 10.3 for *f16*, and 10.7 for *f21*.

With these defined, we can formally state that Ξ can be obtained by the generator as such:

$$\Xi(\vec{W}, \vec{I}, \vec{X}_{opt}) = R^{-1}(\sum_{i=1}^{24} W_i.R_i(f_i, I_i(x - \vec{X}_{opt} + O_i, I_i) - f_i, I_i(O_i, I_i))) \tag{3}$$

where f_i, I_i is instance i of original function f_i and O_i, I_i is the location of the optimum for instance i of function f_i.

4.2 Experimental Setup

We access the 24 noiseless BBOB functions in $2D$ through IOHexperimenter [20].

Affine Combinations. The original BBOB functions involved in the affine pairs are all $2D$, to facilitate visualisation. We consider the region of interest $[-4, 4]$ per dimension only since the optimum of these BBOB functions can only be located in this region. For each function pair, a sequence of 51 values $\alpha \in [0, 1]$ is defined equally spaced with a step of 0.02. We define α as the *proportion of the* Sphere *function*, which is BBOB *f1*. For each combination of two functions with a given alpha, we generate four affine combinations which differ only in the location of the global optimum—we use the same instances of the BBOB constituent parts for each of them. We also

keep the instance number and optimum location consistent across increasing α within each combination of base function pair and optimum placement strategy. Function *instances*—which are the same BBOB function but rotated or translated in different ways—are randomly selected between the numbers 1 and 100. The four placement strategies for the optimum are:

1. near to the boundary (within 0.01) in both coordinates,
2. near the centre (between $[-0.01, 0.01]$ in both coordinates),
3. near to the boundary in one coordinate and central in the other,
4. located randomly for both coordinates between $[-2, 2]$.

In total, we generate 816 affine recombination functions (4 original function pairs $\times$ 51 affine weights $\times$ 4 locations for the optimum). The process of affine combination and placement of the optimum is conducted using functions from the IOHExperimenter package in Python.

Algorithm Performance. For assessing algorithm performance on the 816 functions we consider the top-scoring modCMA configurations for each bias type after careful visual inspection of the SB distributions (See Sect. 3.3) (**bounds** (20 configurations), **centre** (19), **mixed** (7), and **none** (10)). In total, this amounts to 56 CMAES variants and 45 696 algorithm and function-pairs. The algorithm configurations for these are available in Tables 1–4 of the supplemental material [17]. Each CMA-ES configuration is instantiated in modCMA, provided a budget of 5000 evaluations, and is executed 30 times on each of the 816 affine functions. As the performance metric, we use a normalised area under the curve (AUC) with respect to the empirical cumulative distribution function, implemented by Vermetten *et al.*[2]. The distribution function considers the default COCO settings with 51 targets spaced logarithmically beginning at 10^{-8} and terminating at 10^{2}.

5 Results

Figure 4 presents, for different base BBOB function pairings, states of affine combination for increasing α (left to right), which is the proportion of the Sphere function *f1*. Colour represents fitness and the global optimum is placed here near the centre.

Notice from the left-most plots that with $\alpha = 0$, the function contains no aspect of *f1*. For Fig. 4 rows 1 and 2, their intermediate stages ($0.25 \leq \alpha \leq 0.75$) show the influence of the 'roundness' from *f1* being added into the function. In the case of Fig. 4 rows 3 and 4, the effect of increasing α manifests differently, with the bowl shape and concentric structure of *f1* beginning to appear. We can see that when $\alpha = 1$, the function is entirely *f1*.

Figure 5 presents algorithm performance (AUC median and variance) over 30 runs for the top-scoring CMA-ES variants in each bias class across increasing α (shown on the horizontal axis), split by location of the optimum (left to right: bounds, centre, centre of bounds, and random) and for base function pairs (top to bottom): *f3-f1*, *f15-f1*, *f16-f1*, and *f21-f1*.

The first three rows (that is, the first three function pairs) show similar patterns and trends. Generally, increasing α (proportion of *f1*) is associated with increasing

[2] https://zenodo.org/records/10376912.

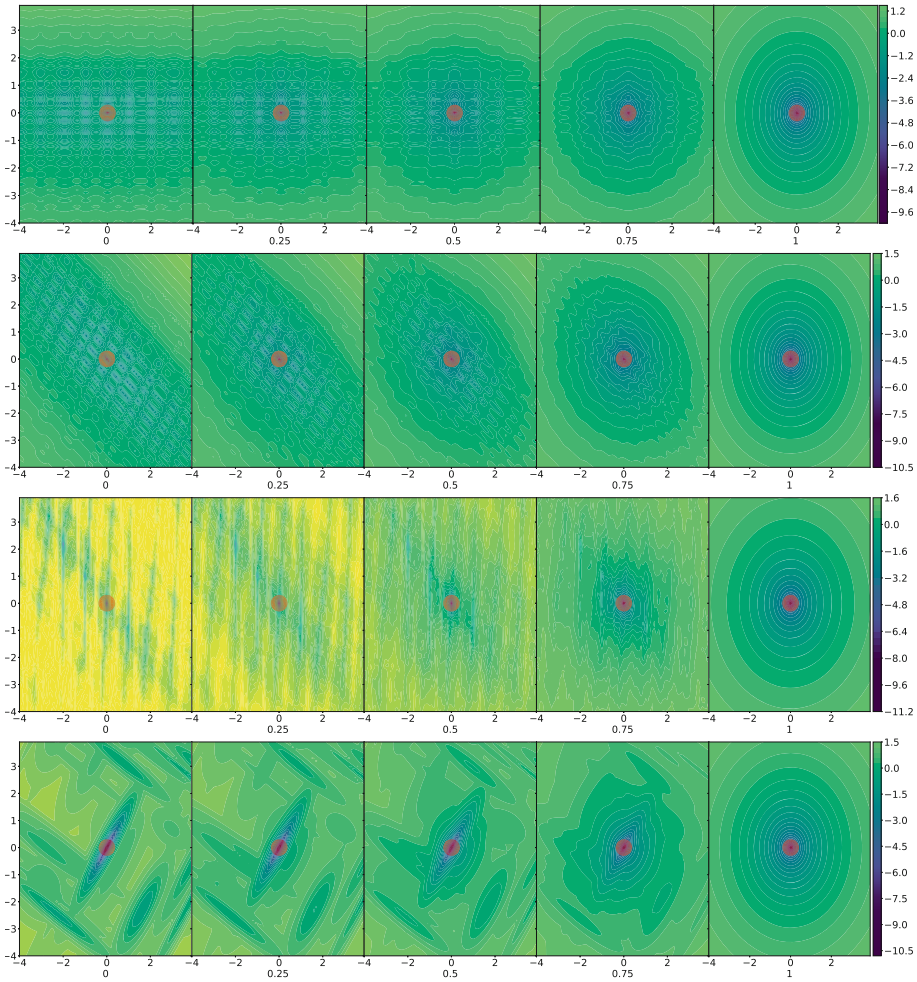


Fig. 4. Example $2D$ landscapes for affine combinations of functions *f3*, *f15*, *f16*, *f21* (top to bottom) with *f1* for 5 affine weights α shown as labels below individual plots, where increasing α corresponds to increasing the proportion of *f1*. On the instances shown here, the location of the global optimum is fixed near the centre of the domain and marked in red. (Color figure online)

performance. Notice by comparing the median lines of, for example, the second plot of the first row with the other three plots in the same row that placing the optimum at the centre leads to the best performances by the algorithms, regardless of bias class. Algorithms perform at their worst when the optimum is placed at the bounds in at least one of the two co-ordinates (see the first and third columns of plots). We also notice from comparing the plots in the final column that when α is 1.00 (that is, when the function being optimised is entirely *f1*) there is a difference in performance—despite

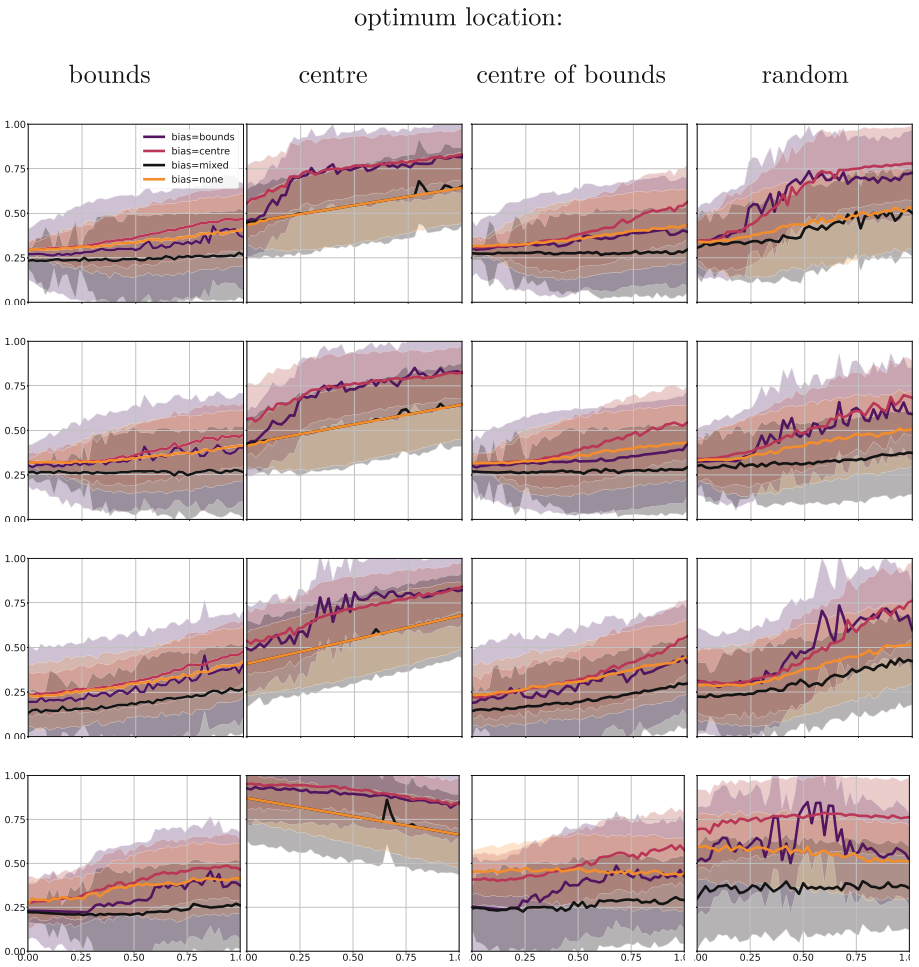


Fig. 5. Performance median and variance over 30 runs for CMA-ES variants; split into bias classes, with one line per bias class, across increasing α on the x-axis. Results are shown for optimum placements (left to right): bounds, centre, centre of bounds, and random—and for base BBOB function pairs (rows, top to bottom): *f3-f1*; *f15-f1*; *f16-f1*; and *f21-f1*.

the fact that the algorithms are being applied to the same function. This may be because this particular column of plots relates to *random* optimum placement.

We now consider how algorithms from the different bias classes compare. Observe from the first and third plot in the first three rows that the *centre* and *none* classes of algorithms perform best (in that order) when the optimum is located near the bounds of the function. This is a curious result: intuitively, the *bounds* algorithms would do the best. We checked some of the algorithm runs and noticed that *centre* algorithms appear to have more freedom of movement—making several small improvements in fitness. On the other hand, *bounds* algorithms seem to sometimes get stuck in these cases when

the optimum is on the bounds—struggling to find a fitness improvement and advance towards the optimum location. We leave a statistical analysis of this phenomenon for future study. In the cases where the optimum is placed either centrally or randomly, the best-performing classes of algorithms are *bounds* and *centre* (notice the second and fourth plots in the first three rows). While it makes sense that *centre* algorithms would perform best on these, the high performance of *bounds* is less intuitive. From examination of a sample of performance runs, it seems that although *bounds* algorithms may begin the search as biased towards the bounds, in the specific case where the optimum is centrally located they seem to be able to navigate towards the centre over the course of the search. It seems that the performance of *bounds*-biased algorithms depends on the location of the optimum, but not in the way which might be expected: if the optimum is at the bounds, they may struggle; if it is at the centre, they do better. Note from the Figures that overall, the *mixed* class of algorithms is the worst performing.

The last function pairing, shown in the last row, differs from the other pairings substantially when the optimum is placed centrally (second column). We notice that performance is excellent (with AUC near to 1 in some cases) and that the trend with respect to α has reversed: increasing α is here associated with a decrease in performance. The explanation for this finding can probably be found in the nature of the original base BBOB function *f21*, where the global optimum can be found near the bounds at the bottom of a half-funnel shape. Observe from the lowest-left plot in Fig. 4 that when the optimum is placed in the centre, this appears to stretch and mirror the half-funnel structure which leads to the optimum. We therefore speculate that this stretched funnel surrounding the optimum (in the case when it is centrally placed) is the reason for high algorithmic performance.

6 Conclusions

This study has systematically explored the interplay between the performance of configurations of the modular Covariance Matrix Adaptation Evolution Strategy (modCMA) and structural bias within different optimisation landscapes. Through the extensive configuration testing of modCMA, encompassing 435 456 configurations, we have shown that specific modules notably influence the algorithm's structural bias. Key insights include the significant impact of modules like covariance adaptation and elitism in modulating structural bias towards the centre of the search space and bound correction method Saturate towards the boundaries of the search space.

To investigate the effects of different forms of SB on algorithm performance, we generated pairwise affine recombinations of BBOB functions with varying proportions of each composite function. For each function we considered four strategies for placing the optimum. The configurations with highest confidence per SB class (predicted by the Deep-BIAS tool) of modCMA, were run 30 times on the affine-combined functions. The results showed that when the optimum is placed at the centre, bounds-biased and centre-biased algorithms perform best. The reason for this is likely that when the optimum is near the centre, bounds-biased algorithms can navigate in the right direction even if they have an inherent bias towards the bounds—the bounds structural bias is mainly caused by the bounds correction method *saturate*, which does not often become active when the search leads away from the bounds. When the optimum is near the bounds, centre-biased and unbiased algorithms are performing better. We believe that

centre-biased algorithms have more freedom of movement, and that bounds-biased algorithms with *saturate* bound correction method become early stuck when the optimum is at the bounds.

Future research should focus on extending the analysis of structural bias effects into higher dimensional spaces. As dimensionality increases, the complex interplay between geometry of high-dimensional spaces, structural bias and landscape features may exhibit different characteristics that could bring additional insights. This future work will not only deepen our understanding of structural bias in iterative algorithms but also guide the development of more robust strategies for tackling complex optimisation problems.

References

1. Bäck, T.H.W., et al.: Evolutionary algorithms for parameter optimization-thirty years later. Evol. Comput. **31**(2), 81–122 (2023)
2. Vermetten, D., Caraffini, F., Kononova, A.V., Bäck, T.: Modular differential evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 864–872. GECCO '23, Association for Computing Machinery, New York, NY, USA (2023)
3. de Nobel, J., Vermetten, D., Wang, H., Doerr, C., Bäck, T.: Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1375–1384 (2021)
4. Kononova, A.V., Corne, D.W., Wilde, P.D., Shneer, V., Caraffini, F.: Structural bias in population-based algorithms. Inf. Sci. **298**, 468–490 (2015)
5. van Stein, N., Vermetten, D., Kononova, A.V., Bäck, T.: Explainable benchmarking for iterative optimization heuristics (2024)
6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
7. Davarynejad, M., van den Berg, J., Rezaei, J.: Evaluating center-seeking and initialization bias: the case of particle swarm and gravitational search algorithms. Inf. Sci. **278**, 802–821 (2014)
8. Vermetten, D., van Stein, B., Caraffini, F., Minku, L.L., Kononova, A.V.: BIAS: A toolbox for benchmarking structural bias in the continuous domain. IEEE Trans. Evol. Comput. **26**(6), 1380–1393 (2022)
9. Rajwar, K., Deep, K.: Uncovering structural bias in population-based optimization algorithms: a theoretical and simulation-based analysis of the generalized signature test. Expert Syst. Appl. **240**, 122332 (2024)
10. Kononova, A.V., Caraffini, F., Wang, H., Bäck, T.: Can compact optimisation algorithms be structurally biased? In: Bäck, T., et al. (eds.) PPSN 2020. LNCS, vol. 12269, pp. 229–242. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58112-1_16
11. Vermetten, D., van Stein, B., Kononova, A.V., Caraffini, F.: Analysis of structural bias in differential evolution configurations. In: Kumar, B.V., Oliva, D., Suganthan, P.N. (eds.) Differential Evolution: From Theory to Practice. SCI, vol. 1009, pp. 1–22. Springer, Singapore (2022). https://doi.org/10.1007/978-981-16-8082-3_1

12. Vermetten, D., Caraffini, F., van Stein, B., Kononova, A.V.: Using structural bias to analyse the behaviour of modular CMA-ES. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1674–1682. GECCO '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3520304.3534035
13. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
14. Lundberg, S.M., et al.: From local explanations to global understanding with explainable AI for trees. Nat. Mach. Intell. **2**(1), 2522–5839 (2020)
15. van Stein, B., Vermetten, D., Caraffini, F., Kononova V, A.: Deep-bias v1.0.0 (2023). https://doi.org/10.5281/zenodo.7614586
16. van Stein, B., Vermetten, D., Caraffini, F., Kononova, A.V.: Deep BIAS: detecting structural bias using explainable AI. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, pp. 455–458 (2023)
17. van Stein, N., Thomson, S., Kononova, A.V.: Supplemental Material for A Deep Dive into Effects of Structural Bias on CMA-ES Performance along Affine Trajectories (2024). https://doi.org/10.5281/zenodo.10994149
18. Dietrich, K., Mersmann, O.: Increasing the diversity of benchmark function sets through affine recombination. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tusar, T. (eds.) Parallel Problem Solving from Nature - PPSN XVII. PPSN 2022. LNCS, vol. 13398. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_41
19. Vermetten, D., Ye, F., Bäck, T., Doerr, C.: MA-BBOB: A problem generator for black-box optimization using affine combinations and shifts (2023). arXiv preprint arXiv:2312.11083
20. de Nobel, J., Ye, F., Vermetten, D., Wang, H., Doerr, C., Bäck, T.: IOHexperimenter: Benchmarking platform for iterative optimization heuristics. Evolutionary Computation, pp. 1–6 (2024)

Automated Algorithm Selection and Configuration

Emergence of Specialised Collective Behaviors in Evolving Heterogeneous Swarms

Fuda van Diggelen(✉), Matteo de Carlo, Nicolas Cambier, Eliseo Ferrante, and Guszti Eiben

Vrije Universiteit Amsterdam, De Boelelaan 1111, Amsterdam, The Netherlands
fuda.van.diggelen@vu.nl

Abstract. Natural groups of animals, such as swarms of social insects, exhibit astonishing degrees of task specialization, useful for solving complex tasks and for survival. This is supported by phenotypic plasticity: individuals sharing the same genotype that is expressed differently for different classes of individuals, each specializing in one task. In this work, we evolve a swarm of simulated robots with phenotypic plasticity to study the emergence of specialized collective behavior during an emergent perception task. Phenotypic plasticity is realized in the form of heterogeneity of behavior by dividing the genotype into two components, with a different neural network controller associated to each component. The whole genotype, which expresses the behavior of the whole group through the two components, is subject to evolution with a single fitness function. We analyze the obtained behaviors and use the insights provided by these results to design an online regulatory mechanism. Our experiments show four main findings: 1) Heterogeneity improves both robustness and scalability; 2) The sub-groups evolve distinct emergent behaviors. 3) The effectiveness of the whole swarm depends on the interaction between the two sub-groups, leading to a more robust performance than with singular sub-group behavior. 4) The online regulatory mechanism improves overall performance and scalability.

Keywords: Swarm robotics · Evolutionary robotics · Heterogeneous swarm

1 Introduction

Collective motion is widely documented in groups of animals in nature and has been shown to enhance the group with capabilities that are not apparent in an individual group member. Those emergent capabilities include increased environmental awareness [6,16], protection against predators [21], and gradient sensing [23]. Swarm robotics aims at implementing such collective behaviors in robots to leverage those advantageous emergent properties for engineering purposes. Collective motion can enable robotic swarms to achieve sensing beyond the capabilities of individual agents [15], that is, emergent perception.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 53–69, 2024.
https://doi.org/10.1007/978-3-031-70068-2_4

The main challenge in designing collective motion behaviors, and swarm robotics systems in general, comes from the fact that designers can only implement robot controllers on individual robots, but that the desired behavior is defined at the group level [11]. Therefore, successful group behaviors depend on emergent properties, which are difficult to predict. An approach to this problem is to define a success metric at the group level and to use it as a reward to automatically optimize robot controllers [9,11,27,32]. However, the solutions that emerge through this process tend to overfit their training environment and, consequently, lack flexibility. Therefore, automated design requires a framework that maintains good performance under a variety of environmental conditions. For example, modularization of individual-level control has been proposed as a solution [9].

We believe that, by leveraging on heterogeneity, we can achieve a modular framework at the swarm level, as opposed to the prevailing 'homogeneous designs' typically found in the swarm robotics literature [9,11,27]. Distinct subgroup behaviors can emerge when task specialization is beneficial for the group as a whole [14]. For example, social insects divide their tasks into sub-tasks, assigned to specific group members, in order to improve their efficiency [24]. Maintaining a 'group identity' while splitting in sub-groups can be achieved through genetic homogeneity with phenotypic heterogeneity. For example, within an insect colony, members share similar-to-identical genotypes [25]. The various behaviors used by insects are encoded in the same genotype and are activated by external cues, e.g., queen pheromones [12]. This phenomenon, in which the same genotype expresses a different phenotype through gene regulatory mechanisms, is known as *phenotypic plasticity*. Although it is more commonly useful for task partitioning [8,24], it has also been found in species exhibiting collective motion [1].

In this paper, we propose to improve the flexibility of automated designs by promoting modularity at the swarm level through heterogeneity in self-organized collective behaviors (as apposed to individual-level). Our framework is based on phenotypic plasticity, where we evolve separate controllers for sub-groups inside the swarm and, through a regulatory mechanism, adjust the phenotypic ratio of each group. More specifically, we consider polyphenism, a specific case of phenotypic plasticity, whereby phenotypic plasticity is expressed at birth and remains constant throughout life [1]. Our work is novel as it evolves a heterogeneous swarm on a swarm level without a priori knowledge of how heterogeneity should be leveraged. This makes our method task-agnostic and highly adaptable.

The paper is organized as follows. In Sect. 2, we present the state of the art on automated design for heterogeneous swarms with a unique approach, that we test in an emergent perception task. Our implementation is detailed in Sect. 3, considering robots that have limited sensing capabilities (which prevents them from achieving the group task individually) and lack awareness of the specific roles or specializations within the swarm. In Sect. 4, we present our optimization results and re-test our best controller with different sub-group ratios. This analysis enables us to design an online regulatory mechanism dependent

on local conditions, where robots automatically switch between the controllers with a probabilistic finite-state machine. We discuss, in Sect. 5, specialization and cooperation of the different sub-groups and conclude our final remarks and future work in Sect. 6.

2 Related Work

Behavioral heterogeneity induces several challenges, the first of which is task allocation, i.e. dynamically adjusting the number of agents assigned to each available task [2]. The problem of achieving this goal in a decentralized manner has been widely addressed by swarm robotics. Approaches to this problem usually focus on the mechanism of task switching and, therefore, use relatively simplistic, manually implemented behaviors. With threshold-based responses, robots initially show some preference for a given task, while simultaneously recognizing deficiencies in the accomplishment of other tasks (e.g. objects accumulating). Beyond some threshold, the robot switches to the corresponding task [17]. In some cases, this switch is probabilistic in order to avoid large-scale population switches, which might leave another task unaddressed [4]. Mathematical modeling of task allocation has also been proposed [31], which allows custom task allocation parameters according to their needs.

In addition to task allocation, task specialization focuses on the emergence of several complementary functions within the swarm. It often refers to physically heterogeneous swarms, i.e. multi-robot systems composed of multiple robot platforms [26]. It should be noted that this impairs a vital advantage of swarms, namely robustness, as in such cases the individual members are not interchangeable with one another. A more robust design is when the agents are physically identical but differ in behavioral function [3]. Functional heterogeneity through behavioral specialization does not suffer from this problem [13]. Here, it is possible for robots with the same body to switch behavior while employed, also called task partition. In such a context, maintaining collective behavior at the group level can be tricky, as online regulatory mechanisms tend to switch individual behaviors only [7].

Tuci et al. investigated the evolution of task partition (i.e. both task allocation and specialization), with a *clonal* and *aclonal* evolutionary process, in a physically homogeneous swarm of five e-pucks. Here, *clonal* refers to a single genotype being shared by all the agent, whereas, with *aclonal*, their genotypes are all different from each other. Unsurprisingly, they found that robots performed better in their *aclonal* approach [28,29], especially if their controller was optimized with a multi-objective fitness [30], as they could address the required sub-tasks in parallel. Notably, this approach results in an efficient optimization process, since each agent in the swarm samples a different genome. Moreover, plasticity was observed, in the sense that individual robots were able to switch tasks according to environmental requirements (including their peers' behaviors).

Closely related, [8] also addressed task partition, with a task that evokes the environmental context of leafcutter ants who divide their foraging task into

two sub-tasks: cutting and dropping leaf fragments into a storage area, on the one hand, and collecting and bringing the fragments back to the nest, on the other. Their experimental environment was composed of a slope separating a nest (below) and a source (above) area so that the robots could individually retrieve the food objects back to the nest, or deposit them on the slope and relying on other robots to fetch them. Task allocation was evolved on a probabilistic finite-state machine, composed of simplistic pre-programmed behaviors, without specifying a preference for collective, rather than individual behavior. Their success demonstrates that even homogeneous controllers (i.e. a single phenotype) can handle task partition through individual experience, stigmergy, and stochastic switching alone, given enough knowledge on the task to design viable behaviors.

The presented work on heterogeneous swarm optimization requires in-depth knowledge of the specific learning task. Whether it be the design of subtasks/goals, pre-defined (modular) behaviors, or finite states; a priori knowledge is required for the design of these controllers. If such insight is available, we could utilize more specialized optimization methods (e.g. [20,22]). Unfortunately, in our case, such prior knowledge is undisclosed with emergent capabilities.

We aim to make collective specialized behavior emerge, without any explicit reward on the specific sub-tasks. Differently from the aforementioned works, our approach requires minimal insight about the solution as we do not pre-define various subtasks/behaviors, such as bucket brigading, finite-state machines, or task-allocations. Instead, we define a reward on the overall group-level performance of the whole task, and let specialization evolve as an optimal solution to simplify the design of a good performance metric. Our method is more flexible than a modular design with pre-defined specialized behaviors that presuppose sub-group interactions affecting the overall swarm performance. In addition, we automatically obtain task allocation through our online regulatory mechanism. This straightforward method minimizes the time and effort required, while demonstrably improving overall task performance. Altogether, this work shows that optimal task partition can be designed automatically using evolutionary computing, without any specific sub-task knowledge. To the best of our knowledge, no previous work exists that addresses task specialization and allocation in such a context.

3 Methodology

Optimising a heterogeneous swarm controller without specific knowledge on any sub-task requires a flexible approach that can be applied on any type of task. We tested our method in an emergent perception task for gradient sensing where robots have to find the brightest spot in the center (inspired by fish behavior described in [23]). For this, we utilize black box optimization in the form of an evolutionary algorithm on Reservoir Neural Networks (RNN), a method that has been used on homogeneous swarms in a similar capacity [32]. Full code base can be found at https://github.com/fudavd/EC_swarm/tree/PPSN_2024

Robot Design
Each robot in the swarm has an identical differential drive hardware design based on the Thymio II without any communication capabilities (Bluetooth, radio, WiFi). Our robot consists of a cart with two actuated wheels (max speed is ±14cm/s) in the back and a passive omni-directional wheel in the front. We equip our robots with range and bearing sensing in 4 directions (specifics are detailed in Sect. 3) and a local value sensor to measure local light intensity. These sensors are sampled at 10 Hz to obtain control inputs only based on current information, i.e. no memory of previous state.

It is important to be explicit on the capabilities of our robot design: 1) Robots do not communicate information to each other (e.g. local values at their position, future motor inputs, or any form of message passing); 2) The controller is memoryless, only current local sensor readings are known; 3) there is no notion of specialization inside the controller, meaning robots are 'unaware' of specialization inside the swarm. All in all, this grounds the idea of 'limited sensing', as a single robot is incapable of estimating the gradient of the light.

Controller Design
For general applicability, we require our controller to be as flexible as possible while capable of learning quickly. For this we opt to use neural networks (which are expressive function approximators) with random functionalities in the form of a reservoir to speed up learning (i.e. RNN, [18]). This reservoir is created by freezing the network weights up to the last layer after random initialization, resulting in a fixed set of functions from which we learn an optimal combination in the final network layer.

To allow specialization, we divide our swarm into two sub-groups, with each sub-group containing a different RNN controller (all sub-group members have the same RNN). The two RNNs are randomly initialized with different reservoirs that we save at the start. We describe the swarm genotype as a single vector of weights from which the first half refers to the last layer of the first RNN, and the second half to the last layer of the second RNN. The phenotypic plasticity of our single genome is expressed through our sub-group division.

The phenotype of the controller is illustrated in Fig. 1. The RNN has an input layer of 9 neurons that are rescaled to $[-1, 1]$, namely 4 directional sensors (each providing two values: distance and heading) and 1 local value sensor. The 4 directional sensors cover a combined 360° view of the robot's surroundings (front, back, left, and right quadrants of 90° each). Within each quadrant (i) the sensor obtains the distance (d_i) and relative heading (θ_i) of the nearest neighbor up to a maximum range of 2 m (outside of this range the sensor defaults d_i=2.01 and θ_i=0). The RNN outputs target speed ($v \in [-1, 1]$) and angular velocity ($w \in [-1, 1]$), which are transformed into direct velocity commands for the two wheels.

The RNN architecture is a fully connected neural network with an input layer with 9 neurons ($\mathbf{s}_{in} \in [-1, 1]^9$), 2 hidden ReLU layers of the same size ($h_1, h_2 \in \mathbb{R}^9$), and a final output layer with two $\tanh^{-1}$ neurons, $RNN \in [-1, 1]^2$. All hidden reservoir weights are initialized randomly with a uniform distribution

$(U[-1,1])$. We set all biases to 0 and only optimize the weights of the output layer during evolution (18 weights per RNN). The final RNN controller can be formalized as follows:

$$RNN = \tanh^{-1}\left(\mathbf{W}_{out}\text{ReLU}\left(\mathbf{W}_{h2}\text{ReLU}\left(\mathbf{W}_{h1}\mathbf{s}_{in}\right)\right)\right)$$
$$\text{with,} \quad \mathbf{W}_{h1,h2} \in \mathbb{R}^{9\times 9} \quad \text{and} \quad \mathbf{W}_{out} \in \mathbb{R}^{2\times 9}$$

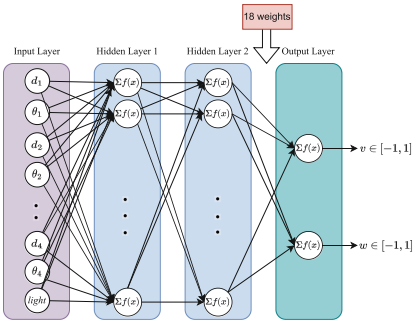
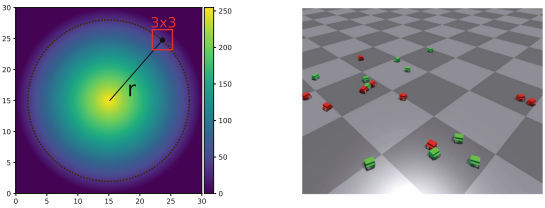


Fig. 1. Reservoir Neuron Network controller design.

Emergent Perception Task

We aim to enhance the sensing capability of our robots so that, when operating collectively as a swarm, they can perceive the gradient. Our set-up consist of 20 robots that are rewarded for navigating to the brightest spot in the center of a 30×30m arena. Since an individual robot lacks the capacity to sense the direction of the gradient field, we anticipate that collective behavior will emerge. This 'task' is not too dissimilar to a behavior found in schools of fish that tend to aggregate in the shadow as it decreases their visibility from predators [23].



(a) Scalar field map (b) Isaac gym

Fig. 2. Experimental setup. (a) The scalar map indicating a random instance of our swarm task. (b) Our swarm with different sub-groups (colored red and green). (Color figure online)

We use Isaac Gym from [19] to simulate our swarm(s) (dt = 0.05s). The task environment is a scalar field map with its maximum value (255) in the center (see Fig. 2a). We randomly placed the swarm in a circle at a fixed distance (r = 12 m) from the center. At this position, we randomly place each swarm member within a 3×3m bounding box (shown in red). The swarm is divided into two sub-groups (red and green) of 10 members each, as shown in Fig. 2b.

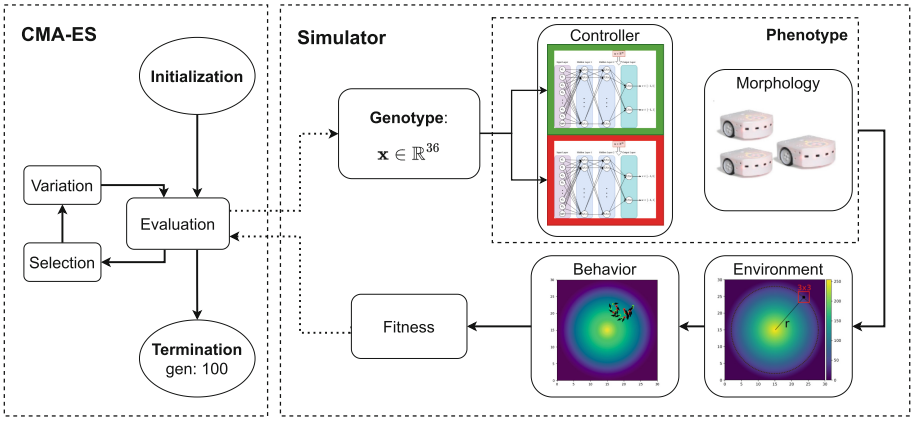


Fig. 3. Experimental setup for optimizing swarm controllers, where an evolutionary algorithm evaluates different genotypes in our swarm simulator (big dashed box). Please note that our genotype encodes two different controllers colored green and white boxes. (Color figure online)

Evolving Swarm Experiment

We optimize our swarm using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES, [10]). We run 10 repetitions of our experiments to obtain the overall best controller. Baseline comparison is made with a homogeneous swarm ofthe same size that is optimized with only one RNN. The choice of CMA-ES is interchangeable with other derivative free methods, as shown in Fig. 3.

Let us draw a clear distinction between the components within *CMA-ES*, where individuals in a population are being optimized, and our *swarm* which refers to an instance of an individual, consisting of robots/members that are assigned to different a sub-group (see Fig. 3). Within our evolutionary algorithm, we thus have individuals that we want to evaluate. All individuals within this evolving population have two RNNs with the same two reservoirs. Differences between individuals are defined by their genotype (a vector of 36 weights, $\mathbf{x} = [\mathbf{W}_{out_{1:}}, \mathbf{W}_{out_{2:}}]$ with, $\mathbf{x} \in \mathbb{R}^{36}$) that encode the last layer weights of the two RNNs. We evaluate our individual by assigning the RNNs to two sub-groups in a single swarm of robots, where each sub-group member (i.e. a robot belonging to a specific sub-group) has the same RNN as the other constituents. After a trial, we calculate a fitness value based on task performance of the swarm and assign it to the corresponding individual.

CMA-ES is a sampling-based evolutionary strategy that aims to find a distribution in the search space to sample high-performing individuals with high probability. Here, CMA-ES samples new candidates $\mathbf{x}$ according to a multivariate normal distribution. The covariance matrix of the sampling distribution is updated at each generation ($\mathbf{C}_{gen}$) to increase the probability of sampling an individual with higher fitness.

We set our population size to 30 individuals (i.e., 30 different swarms) and evolve for 100 generations. Every individual is tested three times, with the median running to represent the final fitness. This reduces the sensitivity to lucky runs that are nonrepeatable and therefore nonviable (Table 1).

Table 1. Evolving swarm experiment parameters

	Value	Description
Runs	10	Number of Runs of our experiment
Learning task: *collective gradient sensing*		
Swarm size	20	Number of robots in a swarm
Ratio	1:1	sub-group division
r	12	Spawn distance from center (meters)
Arena type	center	Environment: center
Eval. time	10	Test duration in minutes
Optimizer: *CMA-ES*		
genotype	$U[-5, 5]$	Initial sampling $\mathbf{x}_{init} \in \mathbb{R}^{36}$
λ	30	Population size
N_{gen}	100	Termination condition
σ_0	1.0	Initial step-size
$N_{repeats}$	3	Number of repetitions per individual

Fitness Function. We define the fitness of a swarm as generally as possible, solely based on the ability to follow the increasing gradient of the scalar field defined by the environment (shown in Fig. 2-a). This is done by aggregating the average light intensity values of all members over time (see Eq. 1).

$$f = \frac{\sum_{t=0}^{T} l_t}{G_{max} \cdot T} \quad \text{and} \quad l_t = \frac{\sum_{n=1}^{N} G_n}{N} \tag{1}$$

where G_n is the scalar value of the grid cell in which agent n (of all agents, N) is located at a time t. Therefore, fitness at a specific time (l_t) is calculated as the mean scalar light value of all swarm members. The trial fitness (f) is calculated by averaging all l_t over total simulation time T. Finally, we normalize using the maximum scalar value G_{max}, always equal to 255 for all experiments. A theoretical maximum fitness of 1 can only be achieved if all members of the swarm stack up in the center for the entire run. Fitness is only evaluated on swarm level with no distinction between sub-groups or any

other task-specific principles that promote specialization. Additionally, we would like to emphasize that our fitness function does not distinguish between robots that are sensing and following the increasing gradient collectively or as solitaries, or sub-groups that cooperate or not.

Validation Experiments

After our evolutionary experiment, we obtain the overall best controller over 10 runs. We re-test this controller to test the emergent perception capability of the swarm to sense the gradient. Our validation is split in two: 1) we are particularly interested in the possible collective (sub-group) behavior(s) and the importance of sub-group interactions, which we elucidate by re-testing the best evolved controller and analysing the swarms behavior; 2) Using our two RNNs, we implement a straightforward online regulatory mechanism to mimic phenotypic plasticity induced by pheromones [12]. We investigate the viability of our (adaptive) heterogeneous swarm through scalability and robustness experiments.

Collective Behavior and Sub-group Interactions: First, we assess whether collective motion has emerged in a single re-test. We examine two different aspects of collective behavior over time: (1) *performance* as mean scalar light value of the swarm (Eq. 1, l_t); (2) *alignment* in terms of order (Eq. 2, Φ), which is defined as follows:

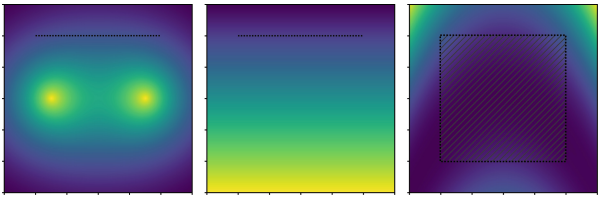
$$\Phi = \frac{\sum_{n=1}^{N} \varphi_n}{N} \quad \text{and} \quad \varphi_n = \frac{\left|\left| \left(\sum_{p=1}^{P} \angle e^{j\theta_p} \right) + \angle e^{j\theta_n} \right|\right|}{P+1} \tag{2}$$

Here, φ_n defines the order value calculated for agent n. Which is the average current heading direction of agent n (noted $\angle e^{j\theta_n}$) and all its perceived neighbors P (noted as $\angle e^{j\theta_p}$). The total swarm order Φ is then defined as the average φ_n for all agents in the swarm. The order measure gives a powerful insight into the alignment of agent's direction of motion. If all agents move towards the same direction, then the order measure approaches 1; and if they move in different directions, the order approaches 0.

Additionally, we investigate the benefits of sub-group interactions on two factors, namely performance and robustness. We retest our best controller in the same environment where we change the sub-group ratios (ratio $\in$ {4:0, 3:1, 2:2, 1:3, 0:4}). Different sub-group ratios can tell us if one sub-group is mainly responsible for the swarm performance or if sub-group interactions are important. These different ratios are tested by initializing the swarm at different distances from the center (r_{ratio} $\in \{0, 0.25, 0.5, 0.75, 1\}$ as a ratio of the original training distance, 12 m).

Online Regulatory Mechanism: Based on the results of the sub-group interactions experiment we can heuristically identify the best performing sub-group ratios at certain light intensities. Subsequently, we create a probabilistic finite state machine where the choice of sub-group controller within a robot is defined such that, on a holistic group level, the optimal sub-group ratios should emerge. This probability is only dependent on the local light value, and thus no communication is implemented to adapt to the best ratio. For example, if we heuristically find a 1:3 ratio to be the best sub-group division at the current local light intensity, the probability to sample an action from the second reservoir is 75%. We update the probabilistic reservoir state every 5 s for stable behavior (this update frequency is found to be optimal by parameter sweep $\{1, 5, 10, 30, 60, 100\}$ seconds).

Scalability and Robustness: The swarm should be able to operate with a wide range of group sizes (i.e. Scalability) and across different types of environment (i.e. Robustness), using the same best controller. In our Scalability experiment, we initialize the swarm in the same environment but with the following swarm sizes 10, 20, 50 to see the impact on performance. In our robustness experiments, we initiate the swarm (of size 20) in different gradient maps 'Bi-modal', 'Linear', 'Banana' as shown below. Each new arena poses different challenges: *Bi-modal*, requires collective decision on where to go (Fig. 4a); *Linear* poses a less salient gradient stretched over the full arena (Fig. 4b); *Banana*, the banana function is a classic nonlinear minimization problem [5] with a curved shallow



(a) Bi-modal (b) Linear (c) Banana

Fig. 4. Validation environments. The black striped line indicates the random initialization location of the swarm (similar to Fig. 2). The striped box in (c) indicates the area of random initialization

Table 2. Validation experiment parameters

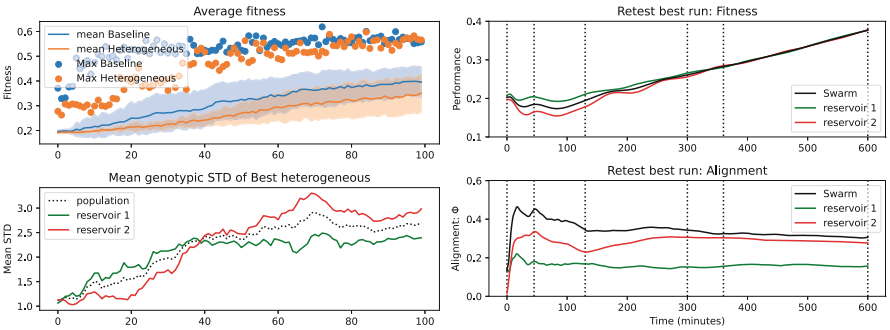
	Value	Description
Repetitions	60	Number of runs per experiment
Statistics	*t*-test	Statistical test
Ratio: *sub-group division environment*		
Swarm size	20	Number of robots in a swarm
Ratio	0:4/1:3/2:2	sub-group division (+inverse)
r_{ratio}	$0/\frac{1}{4}/\frac{1}{2}/\frac{3}{4}/1$	Spawn distance from center (r)
Arena type	center	Environment
Eval. time	10	Test duration in minutes
Scalability: *Different swarm size*		
Swarm size	10/20/50	Number of robots in a swarm
Ratio	2:2/adaptive	sub-group division
Arena type	center	Environment
Eval. time	10	Test duration in minutes
Robustness: *Different environments*		
Swarm size	20	Number of robots in a swarm
Ratio	2:2/adaptive	sub-group division
Arena type	bi-mod./lin./ban.	Environment
Eval. time	10	Test duration in minutes

bottom. For our collective gradient ascent task, this function is interesting, as it has both shallow gradient and local maxima (Fig. 4c). The experimental parameters we used in all the validation experiments are described in Table 2.

4 Results

Evolutionary experiment, data published here: https://doi.org/10.34894/0VSN8Z. We measure the efficacy by the mean and maximum fitness averaged over the 10 independent evolutionary runs for each generation. The results of our evolutionary experiment show similar max performance between baseline (homogeneous) and heterogeneous control Fig. 5a. We see a early plateau in max Baseline while heterogeneous control is still increasing which indicates possible improvement with more generations.

At the bottom of Fig. 5a we show the genotypical variation of our population in the best evolutionary run (calculated as mean Standard Deviation, STD). The rapid increase after generation 20 shows as an increase in variation, while the plateauing of fitness after generation 50 coincides with the steepest decline in genotype variation. It is interesting to see that both reservoirs seem to interchangeably adapt their genotype variation. Where the first reservoir (green) increases first but stabilizes relatively soon, while the second reservoir (red) remains relatively fixed at first and increases when the first reservoir starts to flatten out. This may indicate a concurrent adaptation of reservoirs due to a learned task distribution.



a Evolutionary dynamics **b** Retest best run

Fig. 5. **a**: *top*, mean$\pm STD$ and max fitness over 100 generations (averaged over 10 runs); *bottom*, genotype spread in population (STD) of the best heterogeneous run. **b**: *top*, fitness of a single run with best best controller; *bottom*, the order (alignment) in the same run. Vertical lines correspond with interesting time frames. (Color figure online)

Validation experiments *Collective behavior & sub-group interactions*
Fig. 6 and 5b show the results of our best controller re-tested in the same environment. A video of this run is provided in the supplementary material. During this trial we can measure fitness over time, with its final value around 0.38. More interestingly, we measure the overall alignment of the swarm and the alignment for each sub-group. We

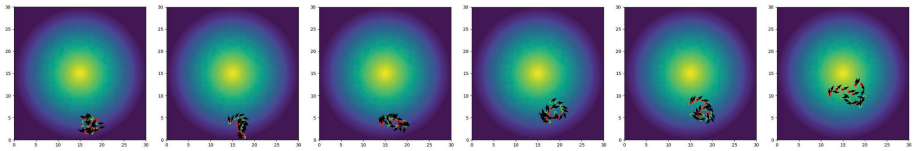


Fig. 6. Snapshots of the interesting timeframes during the re-test experiment. The numbers correspond to the order of the vertical bars in the figure, corresponding to each line plot in Fig. 5b. (1) At first, the swarm spreads out to search for the gradient. (2) The green sub-group senses a gradient and 'aligns' the swarm to the left. (3) The gradient is lost, resulting in the swarm dispersing in different directions (i.e. decreasing alignment). (4) Red sub-group senses the gradient and directs the swarm towards the light source. (5) Red slowly pulls in more green swarm members. (6) the swarm starts to spread around the light source.

provide snapshots of the swarm in the arena in Fig. 6, with the corresponding time frames represented as dotted vertical lines in Fig. 5b (the snapshot from left to right reflect the progression in time).

In Fig. 5b, at the start, we see a low initial swarm alignment (black line) that quickly increases (corresponding to Fig. 6: 1–2). This rapid increase is mainly caused by sub-group 2 (in red) who tends to align more amongst themselves during the whole run. Sub-group 1 (in green) does not align that much, but finds the gradient faster at the start (see snapshot: 3), indicated by a higher fitness. Red follows shortly as a group given their alignment increase. In a later stage, red dominates the swarm's alignment and behavior, thereby concurrently pulling and pushing green towards the center (corresponding to Fig. 6: 4–5). This is also visible by the oscillating sub-group performance.

In Table 3 we evaluated the same best swarm in different setups; we repeat the evaluation at different ratios of sub-groups and at different starting distances from the gradient center. Every configuration was repeated 60 times. We observe how starting closer to the target results in higher fitness. We find that evenly mixed sub-groups at distances far away from the center ($r_{dist} \geq 0.5$) perform statistically significantly higher than any of the extremes (i.e. fully green/red, $p \leq 0.05, df = 118$).

When visually inspecting the behavior of the swarms at $r_{dist} = 0$, we observe how both extreme ratios of sub-groups are capable of staying at the center of the gradient, but with different strategies (video in supplementary material). While the swarm made of only red robots seem to circle around the center with some distance, the swarm made of only green robots seem to fully occupy the space around the center, coinciding with a higher fitness. When evaluating the sub-group ratios at their extremes, we see that both independent sub-groups consistently outperform the best controllers of the first generation in our evolutionary experiments (~ 0.3 at $r_{dist} = 1.0$). However, a mixed ratio of subgroups performs better for all $r_{dist} \geq 0$, indicating an advantage of sub-group interaction. Furthermore, we see a tendency of the second red sub-group to correlate with a higher performance at $r_{dist} \geq 1.0$.

Online Regulatory Mechanism. Based on the results of Table 3 we assign the light intensity threshold values to the best performing sup-group ratio heuristically. I.e., we design a probabilistic state machine such that members in the swarm adapt their behavior

Table 3. Average fitness values ($N = 60$) of retesting the best swarm with different sub-group ratios (green:red) from different starting distances (r_{dist} = distance to the optimum). Sub-group ratios vary from solely sub-group 1 (green) to solely sub-group 2 (red). Solid red boxes indicate best ratio at a given r_{dist}, while the dashed boxed indicate no statistically significant differences with respect to the maximum.

ratio →	4 : 0	3 : 1	2 : 2	1 : 3	0 : 4
$r_{dist} = 0.00$	0.79	0.78	0.77	0.73	0.69
$r_{dist} = 0.25$	0.76	0.76	0.75	0.72	0.68
$r_{dist} = 0.50$	0.70	0.71	0.70	0.66	0.64
$r_{dist} = 0.75$	0.56	0.60	0.60	0.57	0.52
$r_{dist} = 1.00$	0.33	0.41	0.43	0.43	0.38
$r_{dist} = 1.25$	0.06	0.08	0.13	0.09	0.12

automatically to reflect the optimal ratio on a swarm level using only local information. Thresholds are based on the light intensities at $r_{dist} = \{0.125, 0.375, 0.625, 0.875\}$. This results in the following function:

$$P_{green}(light) = \begin{cases} 1.0 & \text{if } light > 229 \\ 0.75 & \text{if } light \in (76, 229] \\ 0.50 & \text{if } light \leq 76 \end{cases}$$

Scalability & Robustness Retesting the best Baseline and best Heterogeneous swarm controller shows significant improvement in Scalability and Robustness ($p \leq 0.05$), with and without our online regulatory mechanism in 6 different environments (3 scalability experiments and 3 robustness). The results of these experiments are presented below (see Table 4). For *Scalability*, the performance of the controllers seems to be positively correlated by the size of the swarm which indicates sensitivity to swarm size. This

Table 4. Validation experiments

Scalability	*Swarm size*		
N=60	10	20	50
Baseline	0.23 ± 0.09	0.33 ± 0.12	0.45 ± 0.05
Best	0.35 ± 0.10	0.39 ± 0.12	0.47 ± 0.03
Adaptive	$\mathbf{0.40 \pm 0.12}^*$	$\mathbf{0.43 \pm 0.077}^*$	$\mathbf{0.49 \pm 0.03}^*$
Robustness	*Arena type*		
N=60	Bi-modal	Linear	Banana
Baseline	0.40 ± 0.19	0.43 ± 0.24	0.19 ± 0.32
Best	0.43 ± 0.17	0.51 ± 0.19	0.25 ± 0.29
Adaptive	$\mathbf{0.47 \pm 0.19}$	$\mathbf{0.59 \pm 0.25}$	$\mathbf{0.31 \pm 0.35}$

positive correlation is the least apparent in the adaptive controller whose performance is the highest for each swarm size (denoted with star; 10 $p \leq 0.05$, 20: $p \leq 0.01$, 50: $p \leq 0.001$). *Robustness* results show the same tendency for the adaptive controller to outperform the others, although these differences were only statistically significant between Adaptive and Baseline. On an aggregate ($N = 360$), Adaptive outperforms the other controllers with Bonferroni correction: Baseline 0.37 ± 0.18 *vs.* Best 0.40 ± 0.18 *vs.* Adaptive 0.45 ± 0.22 with $p \leq 0.01/\alpha$ for all comparisons ($\alpha = 6$, $df = 718$).

5 Discussion

We successfully evolved self-organized sub-group specialization in a swarm of robots, a difficult task due to the complexity of swarm dynamics. This becomes even more evident when factoring in the added complexity of optimizing between-group interactions in conjunction with the specific specialization itself. The strength of our framework lies in its simple approach to learning these behaviors, which lends itself to broad applicability. We demonstrated our methods' effectiveness in the context of an emergent perception task for gradient sensing, which shares similarities to other source localization tasks common in robotics literature [33]. Transferring our method to more complex tasks is also trivial, as 1) the use of task-agnostic nature of RNNs can express a wide array of behaviors, thus requiring no controller design adaptation; and 2) the simplicity of our group-level fitness function can be easily adapted, as its design requires minimal insight into the optimal solution (i.e. no presupposition on certain behaviors or specific task divisions are required).

The results of our validation experiments show the emergence of specialization in our sub-groups that is more robust and scalable than homogeneous control (Table 4). Closer inspection reveals that Sub-group 2 (red) seems to follow the gradient better at low intensities than sub-group 1 (see start of Fig. 5b) and tends to move more in coordination (i.e. higher alignment). In contrast, sub-group 1 performs better when initialized near the center, and shows consistently lower alignment. This indicates that sub-group 1 shows less sensitivity to the swarm's overall behavior and tend to be more greedy as a sub-group. In contrast, sub-group 2 shows more exploratory behavior which provides more coordinated movement of the swarm when the gradient is found.

Exploration and exploitation are fundamental principles in optimization, in general. You could interpret our task as such, where our swarm learns to act as an optimizer that maximizes its local light value. The behavioral findings mentioned above show that the solution converged on a similar exploitation-exploration task division within their sub-group specializations. This task division is interesting as we did not encourage such collective behavior or specialization in our fitness. Arriving at these fundamental optimization principles in the context of sub-group swarm interactions without prior knowledge shows the power of automated design for finding collective behaviors suitable for any user-defined task.

Specialisation provides more robustness and scalability than baseline. Furthermore, leveraging our specialised controllers reveals that only employing one of the two sub-group at lower light intensities leads to lower performances than using a mixed ratio Table 3. This shows that the sub-group performance is enhanced by interactions with the other specialization. Collaboration becomes unnecessary when robots are placed near the center of the gradient ($r_{dist} \leq 0.25$). The online regulatory mechanism furthermore shows the successful division of specialized tasks within our swarm, as it significantly improves performance of the best controller. The idea of biasing evolved

specializations of the swarm adaptively toward a certain phenotype can be found in nature. Phenotypic plasticity emphasizes the expression of specific parts of the genome (i.e. our reservoirs) to obtain higher task competency on a swarm level. In our experiments, we successfully showed that this straightforward implementation of phenotypic plasticity results in higher scalability and significant overall performance.

6 Conclusion

Incorporating specialized behaviors within robot swarms holds the potential to significantly enhance overall swarm efficacy and robustness. However, this endeavor presents formidable challenges. Designing controllers for homogeneous swarms is inherently complex, and extending this to sub-groups within the swarm compounds the difficulty due to the added intricacy of sub-group interactions. In this paper, we show a viable approach to solve this challenge in a (sub)task-agnostic way, by co-evolving controllers heterogeneous swarm controllers while only specifying group-level task performance. We demonstrate that our evolved controllers show clear specialized sub-group behavior with sub-group interactions that improve the collective behavior. These learned behaviors are effectively used in an online regulatory mechanism, to enhance performance and scalability.

In the future, we propose to extend our work to encompass a broader spectrum of tasks, which could reveal other emergent specializations (communication, line-following, and mapping). Additionally, we foresee further improvements in the creation of more sophisticated controller designs where we evolve the number of subgroups and possibly the online adaptation rules to regulate phenotypic plasticity. Finally, we would like to test our controllers in a real world application for which we require the development of sensors to match our work. Such a milestone would be a first step for realizing hardware experiments.

Acknowledgments. This work is supported by Technology Innovation Institute (TII).

References

1. Ariel, G., Ayali, A.: Locust collective motion and its modeling. PLoS Comput. Biol. **11**(12), e1004522 (2015)
2. Bayındır, L.: A review of swarm robotics tasks. Neurocomputing **172**, 292–321 (2016)
3. Bettini, M., Shankar, A., Prorok, A.: Heterogeneous multi-robot reinforcement learning. In: Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, pp. 1485–1494 (2023)
4. Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., Dorigo, M.: Self-organized task allocation to sequentially interdependent tasks in swarm robotics. Auton. Agent. Multi-Agent Syst. **28**, 101–125 (2014)
5. Clerc, M.: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 3, pp. 1951–1957. IEEE (1999)
6. Couzin, I.D., Krause, J., Franks, N.R., Levin, S.A.: Effective leadership and decision-making in animal groups on the move. Nature **433**(7025), 513–516 (2005). https://doi.org/10.1038/nature03236

7. Feola, L., Sion, A., Trianni, V., Reina, A., Tuci, E.: Aggregation through adaptive random walks in a minimalist robot swarm. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 21–29 (2023)
8. Ferrante, E., Turgut, A.E., Duéñez-Guzmán, E., Dorigo, M., Wenseleers, T.: Evolution of self-organized task specialization in robot swarms. PLoS Comput. Biol. **11**(8), e1004273 (2015)
9. Francesca, G., Brambilla, M., Brutschy, A., Trianni, V., Birattari, M.: AutoMoDe: a novel approach to the automatic design of control software for robot swarms. Swarm Intell. **8**(2), 89–112 (2014)
10. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001). https://doi.org/10.1162/106365601750190398
11. Hasselmann, K., Ligot, A., Ruddick, J., Birattari, M.: Empirical assessment and comparison of neuro-evolutionary methods for the automatic off-line design of robot swarms. Nat. Commun. **12**(1), 1–11 (2021)
12. Holman, L.: Queen pheromones and reproductive division of labor: a meta-analysis. Behav. Ecol. **29**(6), 1199–1209 (2018)
13. Hussein, A., Petraki, E., Elsawah, S., Abbass, H.A.: Autonomous swarm shepherding using curriculum-based reinforcement learning. In: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, pp. 633–641 (2022)
14. Ioannou, C.C.: Swarm intelligence in fish? The difficulty in demonstrating distributed and self-organised collective intelligence in (some) animal groups. Behav. Proc. **141**, 141–151 (2017)
15. Karagüzel, T.A., Turgut, A.E., Ferrante, E.: Collective gradient perception in a flocking robot swarm. In: Dorigo, M., et al. (eds.) ANTS 2020. LNCS, vol. 12421, pp. 290–297. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60376-2_23
16. Kearns, D.B.: A field guide to bacterial swarming motility. Nat. Rev. Microbiol. **8**(9), 634–644 (2010). https://doi.org/10.1038/nrmicro2405
17. Krieger, M.J., Billeter, J.B.: The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. Robot. Auton. Syst. **30**(1–2), 65–84 (2000)
18. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**, 127–149 (2009)
19. Makoviychuk, V., et al.: Isaac gym: High performance GPU-based physics simulation for robot learning (2021). arXiv preprint arXiv:2108.10470
20. Montague, K., Hart, E., Paechter, B.: A hierarchical approach to evolving behaviour-trees for swarm control. In: Smith, S., Correia, J., Cintrano, C. (eds.) Applications of Evolutionary Computation. EvoApplications 2024. LNCS, vol. 14634. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-56852-7_12
21. Olson, R.S., Hintze, A., Dyer, F.C., Knoester, D.B., Adami, C.: Predator confusion is sufficient to evolve swarming behaviour. J. R. Soc. Interface **10**(85), 20130305 (2013). https://doi.org/10.1098/rsif.2013.0305
22. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_269
23. Puckett, J.G., Pokhrel, A.R., Giannini, J.A.: Collective gradient sensing in fish schools. Sci. Rep. **8**(1), 1–11 (2018)

24. Ratnieks, F.L., Anderson, C.: Task partitioning in insect societies. Insectes Soc. **46**, 95–108 (1999)
25. Ravary, F., Lecoutey, E., Kaminski, G., Châline, N., Jaisson, P.: Individual experience alone can generate lasting division of labor in ants. Curr. Biol. **17**(15), 1308–1312 (2007)
26. Rizk, Y., Awad, M., Tunstel, E.W.: Cooperative heterogeneous multi-robot systems: a survey. ACM Comput. Surv. (CSUR) **52**(2), 1–31 (2019)
27. Trianni, V., Tuci, E., Ampatzis, C., Dorigo, M.: Evolutionary swarm robotics: a theoretical and methodological itinerary from individual neuro-controllers to collective behaviours. Horiz. Evol. Robot. **153**, 153–178 (2014)
28. Tuci, E.: Evolutionary swarm robotics: genetic diversity, task-allocation and task-switching. In: Dorigo, M., et al. (eds.) ANTS 2014. LNCS, vol. 8667, pp. 98–109. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09952-1_9
29. Tuci, E., Mitavskiy, B., Francesca, G.: On the evolution of self-organised role-allocation and role-switching behaviour in swarm robotics: a case study. In: Artificial Life Conference Proceedings, pp. 379–386. MIT Press (2013)
30. Tuci, E., Rabérin, A.: On the design of generalist strategies for swarms of simulated robots engaged in a task-allocation scenario. Swarm Intell. **9**, 267–290 (2015)
31. Valentini, G., Hamann, H., Dorigo, M.: Global-to-local design for self-organized task allocation in swarms. Intell. Comput. **2022**(4), 1–12 (2022)
32. Van Diggelen, F., Luo, J., Karagüzel, T.A., Cambier, N., Ferrante, E.: Environment induced emergence of collective behavior in evolving swarms with limited sensing. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 31–39. ACM New York, NY, USA, (2022)
33. Wang, S., Wang, Y., Li, D., Zhao, Q.: Distributed relative localization algorithms for multi-robot networks: A survey. Sensors **23**(5), 2399 (2023)

Identifying Easy Instances to Improve Efficiency of ML Pipelines for Algorithm-Selection

Quentin Renau(✉) and Emma Hart

Edinburgh Napier University, Edinburgh, Scotland
{q.renau,e.hart}@napier.ac.uk

Abstract. Algorithm-selection (AS) methods are essential in order to obtain the best performance from a portfolio of solvers over large sets of instances. However, many AS methods rely on an analysis phase, e.g. where features are computed by sampling solutions and used as input in a machine-learning model. For AS to be efficient, it is therefore important that this analysis phase is not computationally expensive. We propose a method for identifying *easy* instances which can be solved quickly using a generalist solver without any need for algorithm-selection. This saves computational budget associated with feature-computation which can then be used elsewhere in an AS pipeline, e.g., enabling additional function evaluations on *hard* problems. Experiments on the BBOB dataset in two settings (batch and streaming) show that identifying easy instances results in substantial savings in function evaluations. Re-allocating the saved budget to hard problems provides gains in performance compared to both the virtual best solver (VBS) computed with the original budget, the single best solver (SBS) and a trained algorithm-selector.

Keywords: Algorithm Selection · Budget Re-Allocation · Black-Box Optimisation

1 Introduction

For any large set of instances, it is well known that different algorithms will elicit different performances, resulting in the need to perform algorithm-selection in order to maximise performance. Typically this is achieved through the use of machine-learning (ML) methods which either predict the label of the best solver or predict the performance of an algorithm [15]. In order to apply an algorithm-selector, some computation is usually required to obtain a feature-vector used by the ML model. In continuous optimisation, the most common approach is to compute Exploratory Landscape Analysis (ELA) features [21], while recent alternatives use short *probing trajectories* as the model input, obtained by running one or more solvers on an instance [19,27]. However, regardless of the type of input chosen, it is crucial to ensure that the analysis required to compute the input data required by a selector does not itself become too expensive [20].

One obvious way to reduce the effort spent on computing the information required to train an algorithm-selector would be to identify *easy* problems that

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 70–86, 2024.
https://doi.org/10.1007/978-3-031-70068-2_5

can be solved to a desired level of precision by a good general solver without any need for a selection process. This is particularly relevant given that Kerschke *et al.* [17] note that algorithm selection systems generally perform poorly on easy function instances. Furthermore, an open-issue in continuous optimisation is that "some problems are embarrassingly easy (e.g., sphere, linear slope)" [16], leading to the authors asking "is there a (cheap) way to distinguish easy from complex problems?". The same sentiment is echoed in [6] in the context of multi-objective optimisation where the authors note that it is "easy to say they [instances] are hard, but hard to say they are easy". While there is a wealth of literature directed at identifying hard problems, particularly in combinatorial optimisation [28,29], there is very little literature directed towards identifying easy ones.

In response to the issues just outlined, we propose a revision to the algorithm-selection pipeline which includes an additional step to predict whether an instance is easy or hard prior to potentially applying a selector. This additional model acts as a filter for recognising 'easy' instances which can be directly solved by a general solver with no need to apply algorithm-selection. The information needed to detect whether an instance is easy is a subset of the information needed by the algorithm-selector: it can be cheaply computed and reused in the selection step if needed. In addition, running the general solver on an easy instance can often produce a solution of desired quality with less budget than the maximum allocated. The unused budget can then be *re-allocated* to enable additional function evaluations on 'hard' instances.

We use the BBOB benchmark test-suite as a testbed [7] to test the new pipeline. Its performance is evaluated in two settings. In the typical 'batch' setting, a set of instances need to be solved and are all available when the solving process starts. We also consider a 'streaming' setting which is typical in many real-word applications: here an infinite stream of instances arrive either periodically or sporadically [4], and each must be solved on arrival, with no knowledge of what might arrive downstream in the future. Specific examples appear in many domains, ranging from the allocation of machines in a factory [23], allocation of processor time slots in a real-time system [2], allocating communication channels in a network [8] or allocation of vehicles for transportation tasks [24].

We demonstrate that:

- Without budget re-allocation, including a classifier to identify easy instances in a pipeline results in a gain in performance compared to using a single trained algorithm-selector. Furthermore, there is only a small loss in performance when compared to the virtual best solver (VBS).
- When re-allocating budget saved by identifying easy instances, we enable extra function evaluations to be performed on hard instances. This results in a considerable performance gain for both the batch and streaming setting, even in comparison to the VBS (calculated using a fixed evaluation budget per instance).

Reproducibility: Code and data related to this study can be found at [25].

2 Related Work

Kothoff *et al.* [20] note that although algorithm-selectors can be beneficial, it is important to ensure that the computational cost of any analysis needed to inform the selector is minimised, stating 'if selecting an algorithm for solving a problem is more expensive than solving the problem, there is no point in doing so'. Note that the cost of using a selector refers to both the cost of computing the input and of executing the selector. In the domain of Boolean satisfiability (SAT) solving, [34] propose the use of a *pre-solver*—an algorithm with good general performance—and start solving an instance with this while simultaneously analysing the instance. The idea is that easy instances will be solved before the analysis phase finishes therefore mitigating the need for an additional algorithm-selection step. However, Tanabe [31] notes that no previous study has used a pre-solver for black-box optimisation so it is unclear to what extent the same process could be used. However, this literature underpins the argument that recognising easy instances in order to save computational effort is an important line of research for the field. Unfortunately, it is equally clear that recognising easy problems is not easy: [6] discuss this in the context of multi-objective combinatorial optimisation problems while [5] conclude that it is 'hard to say it's easy' in general (in relation to combinatorial optimisation).

In the continuous optimisation domain, the vast majority of previous work in AS relies on ELA features which are not cheap to compute: the recommended budget is usually $50d$ samples, with d the dimension of the problem. Some recent work tries to reduce the cost of feature computation by extracting ELA features from a trajectory obtained by running a solver of interest, rather than computing ELA features on a set of separate samples that are then discarded [12,13]. Similarly, in [3], statistics derived from a fitness trajectory are used as features for a selector, however, the budget used to calculate the trajectory is 6 times more than the recommended ELA features budget. In [27], each solver in a portfolio is run for a short time to create a 'probing-trajectory', i.e., short fitness trajectory and this data is used to directly obtain a prediction from a selector: as the selected solver can then be run starting from the point the probing-trajectory terminated, thus saving some budget although this is never re-used. [26] pushes further in this direction in using a selector that uses very short trajectories derived from running a simulated annealing algorithm that has been tuned to generate discriminating trajectories. Despite these recent efforts to reduce the cost of feature-computation, to the best of our knowledge, we are unaware of any cheap method to identify easy instances which can be solved by a generalist solver instead of passing them through an algorithm-selector and can reuse any saved budget to better solve future instances.

3 Motivation

Figure 1 provides an overview of the proposed pipeline in which a new step is introduced to detect whether an instance is easy to solve. First, a small budget

φ_h is used to obtain a feature vector which is passed to the 'hardness classifier': if the instance is classified as easy then the single best solver (SBS) from the portfolio is used to solve the instance with budget b_h. If an instance is labelled as hard, then algorithm-selection proceeds as normal: further input-data for the selector is calculated with budget $\varphi_{AS} > \varphi_h$, and the selector outputs the label of the best solver from the portfolio which is solved using budget $b_{AS} \geq b_h$.

One of the proposed advantages of the pipeline is that it offers an opportunity to dynamically re-allocate budget saved from recognising easy instances to extend the solving budget for hard instances. As additional budget is required to generate features for the algorithm-selector, then this extra budget is saved every time an instance is classified as easy. In addition, a lower budget b_h can be allocated for *solving* instances identified as easy, freeing up further budget. Alternatively, the SBS used to solve easy instances can be terminated once a desired level of precision is reached, i.e. at budget b_t, hence again freeing budget—the amount of budget saved each time will vary depending on how quickly the desired level of precision is reached. Therefore, the total budget saved per easy instance is either $\varphi_{AS} + (b_{AS} - b_h)$ or $\varphi_{AS} + (b_{AS} - b_t)$, depending on whether the run is terminated before b_h evaluations.

The saved budget is re-allocated among hard instances in a manner that depends on the setting:

- In the batch setting in which all instances are known at the start, the *total* budget saved B from solving easy instances is divided equally amongst the remaining n_h hard instances, thus extending the solving budget for each instance by B/n_h
- In the streaming setting, the budget b_i saved on an easy instance is immediately allocated in its entirety to the *next* hard instance that arrives in the stream to extend the run of the predicted solver.

The details of how each step is instantiated in the pipeline are provided in the next section.

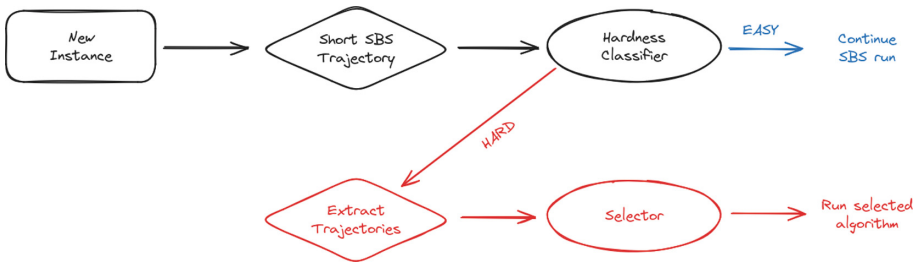


Fig. 1. Algorithm-selection pipeline including the 'easy' instance filter using short trajectories.

4 Methods

We use the Black-Box Optimisation Benchmark (BBOB) functions from the COCO platform [9] as a test-bed. In particular, we consider the first 5 instances of the 24 noiseless functions in dimension $d = 10$. Instances are variations of the same base function such as rotations, translations or scaling. We consider a portfolio of three algorithms—CMA-ES [10], Particle Swarm Optimisation (PSO) [14], and Differential Evolution (DE) [30]. In order to label each instance with the best solver, each algorithm is run 5 times per instance for 4,000 evaluations, and the median fitness is recorded. CMA-ES is best on 11 functions, PSO on 7, and DE on 6. CMA-ES is therefore designated as the single best solver (SBS), i.e., the algorithm having the best median performance. This data is obtained directly from [32] which records search trajectories per run and is described in detail in [33]. Note that as these authors performed some tuning, each solver has a different population size: these are (10,30,40) respectively for CMA-ES, DE and PSO.

4.1 Hardness Classifier

In previous work [26,27], we demonstrated that training an algorithm-selector using *probing-trajectories*—a short time-series of fitness information obtained from running one or more solvers—outperformed classifiers trained using ELA features on the BBOB suite. Therefore we also utilise this approach here to provide input to a 'hardness' classifier. This has the advantage that if an instance is detected as easy, then an instance is simply continued from where the probing trajectory terminated. We label a function as *easy* if *all* 100 runs of the SBS have a performance below 10^{-7}, and *hard* otherwise. This threshold has been chosen arbitrarily and represents a solution that may not be optimal but good enough in practice. For most problems, practitioners are usually able to set a threshold representing a good enough solution. Note that the data is imbalanced: only 3 of the 24 functions are labelled as easy[1].

The hardness classifier is trained using a single short trajectory obtained by running the single best solver (CMA-ES) for a short time (7 generations with a population size 10)[2]. We use an LSTM network [11] as classifier. An LSTM is a type of recurrent network that is able to keep track of arbitrary long-term dependencies in input sequences. Given that our input is a time-series, this is a natural choice. LSTMs have previously been demonstrated to be useful in classifying time-varying data in the online bin-packing domain [1]. The LSTM is trained for 200 epochs, with a learning rate of 0.0001 and optimised with Adam [18]. These parameters were found through empirical tuning. We provide an implementation of the network that can be found in [25]. The model outputs a binary classification that labels an instance as 'easy' or 'hard'.

[1] F1, F5, and F6.

[2] 7 is chosen as this results in a total number of function evaluations that is close to the recommended ELA budget of 500.

To train and evaluate the hardness classifier, we use a 5-fold validation procedure in which a randomly selected 80% of the instance-data is used for training and the remainder for testing, repeated 5 times. The total number of samples is $24 \times 5 \times 100 = 12{,}000$, i.e. 24 functions, 5 instances and with 100 trajectories per instance. Each test-set contains 20% of this data, i.e. 2,400 instances which are presented one by one to the hardness classifier. The model used in the final pipeline is the model with median accuracy from each of the 5 splits.

4.2 Algorithm-Selection

When the output is 'hard', the hardness classifier redirects the pipeline to a classical algorithm-selection model. The input to this classifier consists of concatenated probing-trajectories from each solver in the portfolio. This concatenated vector was demonstrated in [27] to provide improved performance compared to training a classifier with a single probing-trajectory from one solver. Note that the probing-trajectory for the SBS has already been calculated in order to run the first part of the pipeline, so two additional trajectories (from PSO and DE) need to be computed to obtain the input required for this classifier.

To train the model we use trajectories obtained over 7 generations of each of the three algorithms (which corresponds to $7 \times (10 + 30 + 40) = 560$ function evaluations in total for the portfolio)[3]. The algorithm-selection procedure is a classification task, i.e., given a concatenated trajectory representing the instance at-hand, the selector outputs the label of the algorithm that should be run to solve the instance. We compile a dataset using data directly obtained from [32]. This contains 5 runs of each solver per instance. We create a concatenated trajectory per instance per run, resulting in $(24 \times 5 \times 5) = 600$ trajectories. Each trajectory is labelled with the solver that produced the best median performance over the 5 runs of the 5 instances. The algorithm-selector is also an LSTM as described in Sect. 4.1 and is trained in exactly the same manner using a 5-fold validation procedure. We test each of the 5 trained models in turn in the pipeline. For the batch scenario, the instances in each test set can be presented in any order; for the streaming scenario, each instance is presented to the pipeline in exactly the order it occurs in the test set.

4.3 Saving and Re-allocating Budget

We investigate different methods of re-allocating saved budget, depending on the scenario setting (batch or streaming) as described in Sect. 3. We propose two ways to save budget, which can be used in both scenarios:

– *Save Selector Budget (SSB)*: if an instance is classified as easy using the SBS trajectory, then there is no need to obtain trajectories for the other solvers in the portfolio that are needed to use the algorithm-selector. In this case, the

[3] Recall from Sect. 4 that each of three solvers have different population sizes in the data provided by [32].

saved budget b_{SSB} is equal to $7 \times (30 + 40) = 490$ (7 generation trajectories not extracted from DE and PSO).
- *Save Selector Budget and Curtail Easy Runs (SSB-CE)*: if an instance is classified as easy, then the budget for running the SBS is cut by a fixed amount, from 4,000 function evaluations to 2,700 (minimum budget needed for the SBS to reach the 10^{-7} performance for all runs on the three 'easy' functions), therefore saving 1,300 function evaluations on top of the 490 from not evaluating the selector.

In the *batch* setting, the budget saved *per instance* is accumulated over the entire batch of instances. If there are n_h hard instances, this budget is then divided equally between each instance to extend the run of each solver predicted by the algorithm-selector. In the *streaming* setting, budget saved on an easy instance is re-used according to a naïve strategy that extends the length of the run of the *next* instance to be identified as hard with the full saved budget from the previous step. This strategy is simple but provides a reasonable baseline. We discuss potential alternatives in Sect. 6.

5 Results

We first establish the accuracy of the hardness classifier independently of the pipeline and then propose some baselines in which there is no re-allocation of budget. Under this setting (i.e., no re-allocation), both the batch and streaming scenarios are equivalent. Following this, we evaluate the full pipeline on both batch and streaming scenarios using the re-allocation strategies outlined above.

5.1 Baselines

Figure 2 shows the median confusion matrix of the 5 random sub-sampling splits for the hardness classifier. The median accuracy score is 94.5% and the median balanced accuracy score (accounting for imbalanced data) is 82.1%. The imbalance in the data mentioned previously is reflected in the confusion matrix. The 'hard' class is over-represented in the data and is easily classified with 99% accuracy. On the other hand, the under-represented 'easy' class has an accuracy of 65%. This is not unexpected as we did not implement any mechanisms to rectify the imbalance such as weighting classes or oversampling. Nevertheless, the hardness predictor is still useful as discussed below.

Firstly, even if 'easy' instances are only recognised 2/3 of the time, budget is saved on each of these instances. This could amount to a significant saving: for *each* instance identified as easy we save $((30 \times 7) + (40 \times 7)) + 1{,}300 = 1{,}790$ evaluations in the SSB-CE setting: i.e., the number of evaluation needed to compute the PSO and DE trajectories for input to the AS which are run for population sizes (30,40) respectively for 7 generations, and the 1,300 evaluations saved by curtailing the budget. For easy instances that are incorrectly classified as 'hard', no additional cost is incurred—without the additional check in

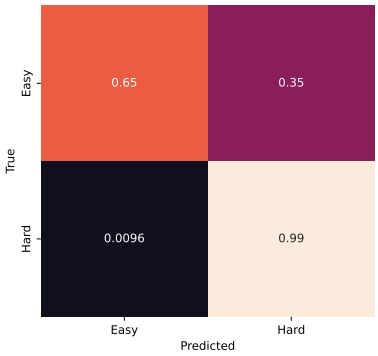


Fig. 2. Median confusion matrix of the hardness prediction.

the pipeline, these instances would have been passed to the algorithm-selector regardless. Secondly, note from Fig. 2 that almost no 'hard' instance is ever mistaken for an 'easy' instance—the accuracy on these instances is 99%. Mistaking a 'hard' instance for an 'easy' one could lead to a significant loss in solution quality as the algorithm-selection step would be bypassed. On the contrary, mistaking easy instances as 'hard' ones does not have any impact on the performance of the pipeline, although they represent a missed opportunity to save budget.

Table 1. Sum of the loss to the VBS for each split of the batch setting. Results are shown for the SBS and the pipeline with and without re-allocation strategies. Positive values indicate a gain over the VBS. Bold indicates the best performance for each column. The selector in the pipeline is the VBS selector.

	Method	Overall loss to VBS	Loss to VBS split1	Loss to VBS split2	Loss to VBS split3	Loss to VBS split4	Loss to VBS split5
Baselines	SBS	−15623.25	−3122.95	−2992.35	−2698.8	−3165.53	−3283.62
	Pipeline No Savings	−70.27	−18.05	−6.27	−24.32	−16.26	−5.38
Re-allocation	Pipeline SSB	129.68	**19.61**	43.56	16.53	16.62	33.36
	Pipeline SSB-CE	**315.37**	−62.83	**67.07**	**122.89**	**92.17**	**96.07**

To establish a baseline, we measure the accumulated loss with respect to the VBS (calculated using the fixed evaluation budget per instance of 4,000) of (1) SBS; (2) the trained algorithm-selector only; (3) using the full pipeline that includes the hardness classifier but uses a 'perfect' algorithm-selector, i.e. the VBS; (4) using the full pipeline that includes the hardness classifier and a trained model as the selector. The loss is calculated as the sum of the performance $VBS_i - t_i$, where i is an instance and t is one of the four methods just described. The sum is calculated over each instance in each of the 5 test sets for each of the four methods t.

Table 2. Sum of the loss to the VBS for each split of the batch setting. Results are shown for the SBS, the trained selector and the pipeline with and without re-allocation strategies. Bold indicates the lowest loss to the VBS for each column. The selector in the pipeline is a trained selector.

	Methods	Overall loss to VBS	Loss to VBS split1	Loss to VBS split2	Loss to VBS split3	Loss to VBS split4	Loss to VBS split5
Baselines	SBS	−1373.93	−335.62	−251.15	−328.59	−276.23	−182.34
	Trained Selector	−471.44	−53.07	−117.81	−116.11	−125.42	−59.04
	Pipeline No Savings	−466.14	−53.96	−117.81	−116.11	−123.34	−54.91
Re-allocation	Pipeline SSB	−450.78	−51.15	−115.27	−114.35	**−119.4**	−50.62
	Pipeline SSB-CE	**−438.67**	**−46.14**	**−110.09**	**−110.43**	−128.5	**−43.51**

The results are shown in Table 1 when the VBS is used as selector in the pipeline and in Table 2 when the trained selector is used in the pipeline ('Baselines' rows). Table 1 shows that the pipeline with no re-allocation outperforms the SBS but has lower performance than the VBS. Both results are expected: since the selector in the pipeline is the VBS selector, it is expected to outperform the SBS. Moreover, the pipeline adds an imperfect hardness classifier to the VBS selector, we expect to see a degradation compared to purely the VBS.

We observe in Table 2 (Baselines rows) that both the trained selector and the pipeline without re-allocation outperform the SBS. This result was expected given that no one solver outperforms the others on all functions and thus there is a benefit to performing algorithm-selection. We also observe that both the trained selector and the pipeline with no re-allocation obtain similar losses to the VBS. These results could also be anticipated since no budget is saved when the hardness classifier is used. Interestingly, the performance of the pipeline is slightly better than the performance obtained by using the trained selector only, even though it only adds a hardness classifier. The use of the hardness classifier seems to mitigate the selector mistakes, i.e., when the selector gets the algorithm-selection wrong on 'easy' instances, then the use of the SBS as a result of classifying the instance as easy almost always leads to better results.

Overall, we observe that using the pipeline presented in Fig. 1 does not degrade the performances compared to a classical algorithm-selection approach. The next sections present results where the pipeline is fully leveraged, i.e., where unused budget is saved and re-allocated.

5.2 Re-allocating Budget: Batch Setting

Algorithm Selection with VBS: We use a test set composed of instances that were not used to train the algorithm selector model, i.e., a batch is composed of $20\% \times 12{,}000 = 2{,}400$ instances. The process is repeated five times, i.e. for each split. These instances are first fed to the hardness classifier and the number of instances classified as 'easy' counted. This determines how many extra function evaluations can be allocated to extending the runs for instances classified as

hard. In the five repetitions of the batch settings, there are between 6.5% and 10% of instances are identified as 'easy'. This results in a median 49 function evaluations added to each of the remaining instances classified as hard when using the SSB strategy and 179 function evaluations when using the SSB-CE strategy.

Table 1 shows the overall results and results for each split of the data. Interestingly, both re-allocation budget approaches outperform the VBS, i.e., re-allocating budget to harder instances improves the overall performance. The SSB-CE strategy obtains the best performance: its overall gain is more than twice the gain of SSB. Moreover, on split 3, SSB-CE obtains almost the same gain as SSB on the five splits combined. However, SSB-CE can also be the worst-performing approach: on split 1, the loss to the VBS is substantial and nearly matches the overall loss of the pipeline without savings.

Algorithm Selection with Trained Selector: In these experiments, we use exactly the same procedure outlined in the previous paragraph but replace the VBS selector with a trained selector. Over the five repetitions, between 6.7% and 10% of instances are identified as 'easy'. This results in a median 39 function evaluations added to each of the remaining instances using the SSB strategy and 145 function evaluations using the SSB-CE strategy.

Table 2 shows the loss to the VBS for the three pipelines: without savings, SSB, and SSB-CE. Even though the number of identified 'easy' instances represents a small part of the whole batch and relatively few evaluations are added to the remaining functions, we observe a gain compared to the 'no savings' result. The best gain is obtained using the SSB-CE strategy with 5.9% improvement compared to no savings. Although this is the best-performing strategy overall, we should note that SSB-CE is outperformed by the trained selector in the split 4. This is due to the curtailing of easy runs, i.e., running the SBS with a smaller number of function evaluations can lead to decreasing performances for instances mistakenly labelled as 'easy'. The SSB strategy offer a good compromise, i.e., it performs 3.3% better than the pipeline with no savings but consistently performs better when considered over all splits.

Overall, SSB-CE obtains better performances with both the VBS and trained selector but is also the most volatile strategy, i.e., although big gains are achieved overall, losses to classical algorithm-selection can arise in some settings.

5.3 Re-allocating Budget: Streaming Setting

Contrary to the batch setting, in this section we consider a stream where instances to solve arrive one by one and must be solved in the order they arrive. We consider two pipelines: one where the algorithm selector is the VBS selector and one where the selector is a trained classifier.

Algorithm Selection with VBS: Figure 3 displays the cumulative fitness gain obtained for each of the 5 streams of unseen instances when budget is saved on 'easy' instances and then reused on the next predicted instance that is predicted

to be‘hard’. The left figure shows the gain when only the budget from the selector is saved (SSB) while the right figure shows the results when easy runs are also curtailed, i.e., fewer evaluations are performed on ‘easy’ instances (SSB-CE).

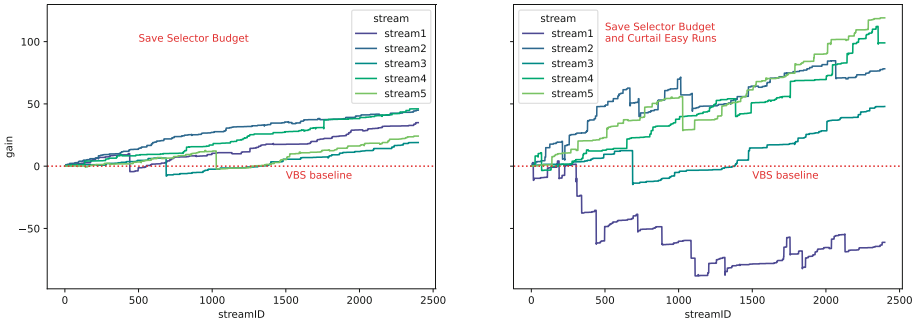


Fig. 3. Cumulative difference between the full pipeline using the hardness classification and VBS selector when budget is saved and re-allocated.

Overall, for every model split, we observe a positive gain by using the extra budget on ‘hard’ instances. The median gain in fitness value on SSB is 34.94 while saving additional evaluations on the ‘easy’ instances (SSB-CE) leads to a 78.22 median gain, more than twice the gain obtained with the selector budget alone. Despite these gains, we also observe losses for some instances. These occur due to a misclassification by the hardness classifier. This has a bigger impact when using the SSB-CE strategy: if an instance is not easy and in particular if CMA-ES (the SBS) performs poorly on that function, then this results in a larger loss due to the curtailed run. This appears to have a particular impact in one of the five models (stream 5).

The results are summarised in Fig. 4. This shows a small loss in comparison to the VBS when not re-allocating saved budget, contrasting to positive gains when using the two forms of budget saving and re-allocation. Here it is clear that by re-allocating unused budget, we realise a gain in performance compared to VBS performance (calculated as previously explained in Sect. 4.2). This is possible because re-allocating the budget enables longer runs on some instances, which then reach better fitness. As expected, this is maximised using the curtailing of easy runs which re-allocates more budget.

Algorithm Selection with Trained Selector: Recall from Sect. 4.2 that the test datasets used to evaluate the 5 trained algorithm-selectors models each only contain 120 instances. In addition to this only being a short stream, it is important to recognise that any gain due to re-allocating saved budget will be influenced by the order in which instances are presented: this results from the simple strategy used which re-allocates saved budget to the next hard instance. Therefore, in order to quantify the effect of this, we create a longer stream in which we randomly select 120 samples (with replacement) from the original 120 test instances,

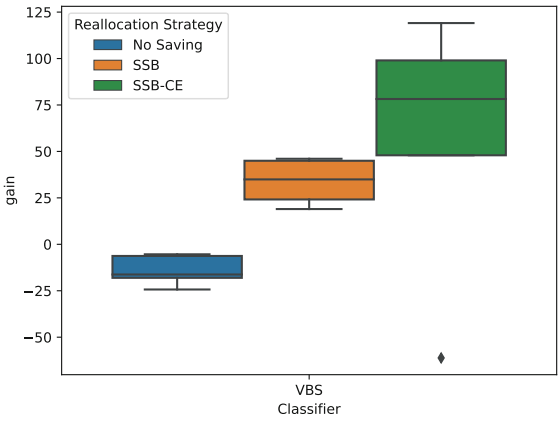


Fig. 4. Boxplots of gains at the end of the stream of instances for each saving budget strategy with VBS selector.

repeating this 20 times, resulting in a stream of length 2,400. Each group of 120 samples therefore may have a different set of instances and a different ordering.

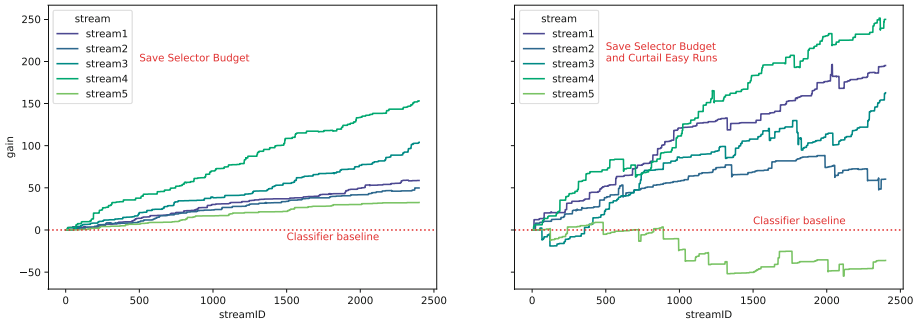


Fig. 5. Cumulative difference between the full pipeline using the hardness classification and trained selector when budget is saved and re-allocated.

Figure 5 displays the cumulative fitness gains obtained on the 5 streams of unseen instances when budget is saved on 'easy' instances and then reused on the next predicted 'hard' instance with a trained selector. Overall, the same behaviour can be seen as observed with a VBS selector, i.e., saving budget improves the performance of the pipeline while saving budget by curtailing runs can lead to bigger gains but also to some losses in performance. For every model but one, the gains obtained with SSB-CE are greater than using SSB. The latter provides a gain which is greater than only using the pipeline without budget re-allocation. For example, for stream 4, the gain of 75.5 obtained by using the pipeline reaches 153 using SSB and 249.7 when SSB-CE is used.

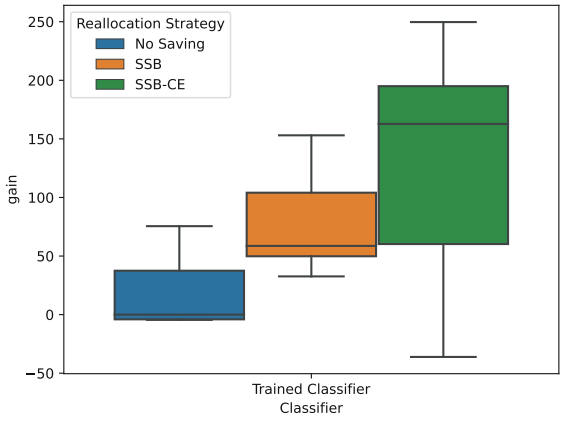


Fig. 6. Boxplots of gains at the end of the stream of instances for each saving budget strategy with the trained selector.

Figure 6 summarises the performance gains using the pipeline that includes the hardness classifier compared to an algorithm-selector only. As described in Sect. 5.2, we see a small gain, even when no budget re-allocation mechanism is used. Much larger gains are seen by re-allocating budget, with SSB-CE again providing the best results.

6 Conclusion

This article is motivated by literature that recognises (1) the difficulty of recognising easy instances [6]; (2) the computational cost of deriving information needed to train algorithm-selectors; (3) the fact that algorithm selectors do not perform well on easy instances [15]; (4) the need to make the most efficient use of a fixed function evaluation budget in an algorithm-selection pipeline. To address these issues, we proposed a pipeline (Fig. 1) that includes a 'hardness' classifier whose role is to filter out easy instances which are simply solved using the SBS. The effect of including this classifier is two-fold: (1) it *saves* budget both by removing the need to calculate some of the input required by an algorithm-selector and by curtailing the run length of easy instances; (2) it *re-allocates* this budget to hard instances, enabling larger solving budgets which results in better performance.

The pipeline was evaluated over two scenarios found in practical applications, i.e., a batch setting where all instances to be solved are known, and a streaming setting in which instances arrive one at time and must be solved immediately.

Over a stream of instances generated from the BBOB benchmark suite, we show that the proposed pipeline including re-allocation results in a gain in performance compared to using a trained algorithm-selector on the hard instances. The magnitude of the gain increases as the amount of budget saved and re-allocated increases. We also show that we can improve on an 'oracle' which uses a fixed

budget to identify the VBS (albeit unrealistic in practice). This is possible due to the dynamic re-allocation of saved budget which increases run length—using the pipeline results in 7% of budget becoming available for re-allocation.

A critical aspect of the design of the pipeline is that there should not be a cost to computing the data required as input to the hardness classifier which would negate the proposed benefits. We addressed this by using a hardness classifier whose input is a *subset* of the data required by the algorithm-selector, therefore needs to be computed regardless of whether or not the hardness classifier is used. Specifically, we use short fitness trajectories as input, which fit the criterion identified above and have recently been shown to outperform selectors based on ELA features [27]. Note that a pipeline that used ELA features as input to both classifiers would in fact also fit this criterion as feature vectors calculated for the hardness classifier could directly be used in the algorithm-selector. However, although a pipeline using ELA features could achieve some performance benefits, budget would only be saved by identifying easy instances and reducing the length of the run of the SBS solver (as opposed to the trajectory approach which also saves budget by reducing feature computation). In addition, in some circumstances, incorrectly identifying instances as easy and solving with the SBS can mitigate an incorrect prediction with the algorithm-selector.

Future work can be separated into two categories: (1) improving the hardness classifier and (2) improving the strategy for re-allocating budget. The accuracy of the hardness classifier could be improved by addressing the imbalance in the training dataset, for example generating new easy instances through instance-generation methods (e.g., [22]) or using imbalance correction techniques such as weighting, down-sampling, or over-sampling. With regard to budget re-allocation, we used a naïve strategy of spending for both the batch and streaming scenarios. Alternative methods could be explored such as spreading the budget over the next h 'hard' instances in the stream or even predicting the level of 'hardness' via regression rather than a binary label of easy/hard.

Acknowledgments. The authors are supported by funding from EPSRC award number: EP/V026534/1

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

References

1. Alissa, M., Sim, K., Hart, E.: Automated algorithm selection: from feature-based to feature-free approaches. J. Heuristics **29**(1), 1–38 (2023). https://doi.org/10.1007/s10732-022-09505-4
2. Buttazzo, G.C.: Hard Real-time Computing Systems: Predictable Scheduling Algorithms and Applications, vol. 24. Springer Science & Business Media, New York (2011). https://doi.org/10.1007/978-1-4614-0676-1

3. Cenikj, G., Petelin, G., Doerr, C., Korosec, P., Eftimov, T.: Dynamorep: trajectory-based population dynamics for classification of black-box optimization problems. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2023, Lisbon, Portugal, 15-19 July 2023, pp. 813–821. ACM (2023). https://doi.org/10.1145/3583131.3590401
4. Degorre, A., Maler, O.: On scheduling policies for streams of structured jobs. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 141–154. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85778-5_11
5. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR-Spektrum **22**, 425–460 (2000)
6. Figueira, J.R., et al.: Easy to say they are hard, but hard to see they are easy-towards a categorization of tractable multiobjective combinatorial optimization problems. J. Multi-Criteria Decis. Anal. **24**(1–2), 82–98 (2017)
7. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2010: presentation of the Noiseless Functions (2010). http://coco.gforge.inria.fr/downloads/download16.00/bbobdocfunctions.pdf
8. Gan, C.H., Lin, P., Perng, N.C., Kuo, T.W., Hsu, C.C.: Scheduling for time-division based shared channel allocation for UMTS. Wireless Netw. **13**, 189–202 (2007)
9. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tusar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. Optim. Methods Softw. **36**(1), 114–144 (2021). https://doi.org/10.1080/10556788.2020.1808977
10. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001). https://doi.org/10.1162/106365601750190398
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
12. Jankovic, A., Eftimov, T., Doerr, C.: Towards feature-based performance regression using trajectory data. In: Castillo, P.A., Jiménez Laredo, J.L. (eds.) EvoApplications 2021. LNCS, vol. 12694, pp. 601–617. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72699-7_38
13. Jankovic, A., Vermetten, D., Kostovska, A., de Nobel, J., Eftimov, T., Doerr, C.: Trajectory-based algorithm selection with warm-starting. In: IEEE Congress on Evolutionary Computation, CEC 2022, Padua, Italy, 18-23 July 2022, pp. 1–8. IEEE (2022). https://doi.org/10.1109/CEC55065.2022.9870222
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995). https://doi.org/10.1109/ICNN.1995.488968
15. Kerschke, P., Hoos, H., Neumann, F., Trautmann, H.: Automated algorithm selection: survey and perspectives. Evol. Comput. **27**(1), 3–45 (2019)
16. Kerschke, P., Preuss, M.: Exploratory landscape analysis. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, p. 990-1007. GECCO 2023 Companion, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3583133.3595058
17. Kerschke, P., Trautmann, H.: Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. Evol. Comput. **27**(1), 99–127 (2019)
18. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

19. Kostovska, A., et al.: Per-run algorithm selection with warm-starting using trajectory-based features. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds) International Conference on Parallel Problem Solving from Nature, vol. 13398, pp. 46–60. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_4
20. Kotthoff, L., Kerschke, P., Hoos, H., Trautmann, H.: Improving the state of the art in inexact TSP solving using per-instance algorithm selection. In: Dhaenens, C., Jourdan, L., Marmion, M.-E. (eds.) LION 2015. LNCS, vol. 8994, pp. 202–217. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19084-6_18
21. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory Landscape Analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 829–836. ACM (2011). https://doi.org/10.1145/2001576.2001690
22. Muñoz, M., Smith-Miles, K.: Generating new space-filling test instances for continuous black-box optimization. Evol. Comput. **28**(3), 379–404 (2020)
23. Pinedo, M.: Planning and Scheduling in Manufacturing and Services. Springer, New York (2005). https://doi.org/10.1007/978-1-4419-0910-7
24. Raff, S.: Routing and scheduling of vehicles and crews: the state of the art. Comput. Oper. Res. **10**(2), 63–211 (1983)
25. Renau, Q., Hart, E.: Identifying easy instances to improve efficiency of ml pipelines for algorithm-selection - code and data (2024). https://doi.org/10.5281/zenodo.10590233
26. Renau, Q., Hart, E.: Improving algorithm-selection and performance-prediction via learning discriminating training samples. In: In Press of the Genetic and Evolutionary Computation Conference, GECCO 2024, Melbourne, VIC, Australia, 14-18 July 2024, pp. 813–821. ACM (2024). https://doi.org/10.1145/3638529.3654025
27. Renau, Q., Hart, E.: On the utility of probing trajectories for algorithm-selection. In: Smith, S., Correia, J., Cintrano, C. (eds.) Applications of Evolutionary Computation - 27th European Conference, EvoApplications 2024, Held as Part of EvoStar 2024, Aberystwyth, UK, 3-5 April 2024, Proceedings, Part I. LNCS, vol. 14634, pp. 98–114. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-56852-7_7
28. Smith-Miles, K., Christiansen, J., Muñoz, M.: Revisiting where are the hard knapsack problems? Via instance space analysis. Comput. Oper. Res. **128**, 105184 (2021)
29. Smith-Miles, K., van Hemert, J., Lim, X.Y.: Understanding TSP difficulty by learning from evolved instances. In: Blum, C., Battiti, R. (eds.) LION 2010. LNCS, vol. 6073, pp. 266–280. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13800-3_29
30. Storn, R., Price, K.: Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997). https://doi.org/10.1023/A:1008202821328
31. Tanabe, R.: Benchmarking feature-based algorithm selection systems for black-box numerical optimization. IEEE Trans. Evol. Comput. **26**(6), 1321–1335 (2022). https://doi.org/10.1109/TEVC.2022.3169770
32. Vermetten, D., Hao, W., Sim, K., Hart, E.: To Switch or not to Switch: Predicting the Benefit of Switching between Algorithms based on Trajectory Features - Dataset (2022). https://doi.org/10.5281/zenodo.7249389

33. Vermetten, D., Wang, H., Sim, K., Hart, E.: To switch or not to switch: predicting the benefit of switching between algorithms based on trajectory features. In: Correia, J., Smith, S., Qaddoura, R. (eds.) Applications of Evolutionary Computation, vol. 13989, pp. 335–350. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30229-9_22
34. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Satzilla: portfolio-based algorithm selection for sat. Journal of artificial intelligence research **32**, 565–606 (2008)

Landscape-Aware Automated Algorithm Configuration Using Multi-output Mixed Regression and Classification

Fu Xing Long[1,2(✉)], Moritz Frenzel[3], Peter Krause[4], Markus Gitterle[5], Thomas Bäck[1], and Niki van Stein[1]

[1] LIACS, Leiden University, Niels Bohrweg 1, 2333 Leiden, Netherlands
{f.x.long,t.h.w.baeck,n.van.stein}@liacs.leidenuniv.nl
[2] BMW Group, Knorrstraße 147, 80788 Munich, Germany
fu-xing.long@bmw.de
[3] Altair Engineering GmbH, Calwer Straße 7, 71034 Böblingen, Germany
mfrenzel@altair.com
[4] Divis Intelligent Solutions GmbH, Joseph-von-Fraunhofer-Straße 20, 44227 Dortmund, Germany
krause@divis-gmbh.de
[5] Munich University of Applied Sciences, Dachauer Straße 98b, 80335 Munich, Germany
markus.gitterle@hm.edu

Abstract. In landscape-aware algorithm selection problem, the effectiveness of feature-based predictive models strongly depends on the representativeness of training data for practical applications. In this work, we investigate the potential of randomly generated functions (RGF) for the model training, which cover a much more diverse set of optimization problem classes compared to the widely-used black-box optimization benchmarking (BBOB) suite. Correspondingly, we focus on automated algorithm configuration (AAC), that is, selecting the best suited algorithm and fine-tuning its hyperparameters based on the landscape features of problem instances. Precisely, we analyze the performance of dense neural network (NN) models in handling the multi-output mixed regression and classification tasks using different training data sets, such as RGF and many-affine BBOB (MA-BBOB) functions. Based on our results on the BBOB functions in $5d$ and $20d$, near optimal configurations can be identified using the proposed approach, which can most of the time outperform the off-the-shelf default configuration considered by practitioners with limited knowledge about AAC. Furthermore, the predicted configurations are competitive against the single best solver in many cases. Overall, configurations with better performance can be best identified by using NN models trained on a combination of RGF and MA-BBOB functions.

Keywords: Black-box optimization · Exploratory landscape analysis · Multi-output mixed regression and classification · Dense neural network · Randomly generated functions

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 87–104, 2024.
https://doi.org/10.1007/978-3-031-70068-2_6

1 Introduction

In landscape-aware algorithm selection problem (ASP) [25,33], the performance of optimization algorithms has been linked to the landscape characteristics of black-box optimization (BBO) problems that are quantified using fitness landscape analysis [20]. By constructing machine learning (ML) models, for instance, the performance of optimization algorithms can be estimated based on the landscape characteristics of problem instances [11]. In other words, the problem landscape characteristics can be exploited to select well-performing optimization algorithms from an algorithm portfolio prior to the actual optimization runs. Using a large set of problem instances, and preferably from diverse optimization problem classes, the corresponding landscape characteristics and algorithm performances are utilized for the training of ML models. Following this, the effectiveness of predictive models is heavily dependent on the representativeness of training data for unseen BBO problems.

Although landscape features are informative in explaining algorithm performances [31], landscape-aware ASP was mainly investigated on benchmarking problems in previous work, such as the widely-used black-box optimization benchmarking (BBOB) suite [7]. The fact that the BBOB suite is not representative enough for engineering applications, such as crashworthiness optimization [14,15] and control system calibration [34] in the automotive industry, raises concerns that predictive models trained using only the BBOB suite might generalize poorly to unseen problem classes that are not being covered. Moreover, for real-world BBO problems that require expensive function evaluations, e.g., time-consuming and/or costly simulation runs, the function evaluation budget can be particularly limited, hindering the generation of a large data set for the model training. To fill in the gap, we explore an alternative in building pre-trained general purpose models that can generalize well for different applications, including expensive BBO problems, while maintaining an affordable computational effort. Our ultimate vision is to assist practitioners with little domain knowledge about ASP, e.g., from engineering fields, to automatically identify the best suited algorithm configuration for their applications.

Our Contribution: In this work, we investigate the potential of tree-based randomly generated functions (RGF) for the training of predictive models, which are much more diverse than the BBOB suite in terms of optimization landscape characteristics. In this context, we implement a selection process to identify RGF that are appropriate as training data. Furthermore, we extend our investigations towards landscape-aware automated algorithm configuration (AAC) by combining both algorithm selection and hyperparameter optimization (HPO), that is, finding the best suited algorithm and fine-tuning its hyperparameters. For the prediction of optimal configurations, we consider dense neural network (NN) models, which can easily handle multi-output mixed regression and classification tasks. Based on our empirical results, near optimal configurations can be identified using the proposed approach, which can outperform the off-the-shelf default configuration and compete against the single best solver (SBS) for many

BBOB functions. In some cases, NN models can perform better than random forest (RF) models, which are typically considered for landscape-aware ASP.

This paper has the following structure: Related works are introduced in Sect. 2, followed by the explanations of our methodology in Sect. 3 and experimental setup in Sect. 4. Next, results are analyzed and discussed in Sect. 5. Lastly, conclusions and future works are presented in Sect. 6.

2 Related Work

The idea of using RGF for the training of feature-based predictive models has been previously investigated, such as in [42]. In summary, it was reported that RGF were ineffective for the training of high-quality models in terms of prediction accuracy. Independently of the previous work, our work differs mainly in the following extensions.

1. Instead of simply using any RGF, we implement an intermediate step to select RGF that are appropriate for the model training purposes. We argue that this step is crucial to improve model accuracy, as discussed in Sect. 3.1. In fact, we suspect that this might partly explain the low model accuracy in [42].
2. We propose to consider NN-based predictive models to handle the multi-output mixed regression and classifications tasks in AAC, which can sometimes perform slightly better than RF models, refer to Sect. 3.2.
3. Rather than just selecting the best algorithm from a portfolio of limited algorithms, as typically done in ASP, we extend our investigations towards combined algorithm selection and hyperparameter optimization (CASH) [35], or we call AAC [33] in this work.

2.1 Automated Algorithm Configuration

To tackle AAC problems, where the search space can be a mix of continuous, integer, categorical, and conditional variables, various optimization algorithms have been implemented, such as tree-structured Parzen estimator (TPE) [1] and sequential model-based algorithm configuration (SMAC) [13]. As a variant of Bayesian optimization [23], TPE utilizes Parzen estimators as surrogate models, which can handle mixed-integer search space and scale well to high dimensionality. For example, TPE has been previously applied to fine-tune the learning rates of covariance matrix adaptation evolutionary strategy (CMA-ES) [41].

In this work, we focus on fine-tuning the configuration of modular CMA-ES [26], developed based on the original CMA-ES algorithm [8,9]. In short, different variants, such as active learning, mirrored sampling, threshold convergence, and recombination weights, are integrated as modules that can be individually activated or deactivated, allowing a custom instantiation of CMA-ES. Subsequently, modular CMA-ES offers a convenient examination of the interactions between different modules as well as between modules and hyperparameters, e.g., population size and learning rates.

2.2 Black-Box Optimization Benchmarking

In previous work, landscape-aware ASP was commonly investigated based on BBO benchmarking suites, such as the well-known BBOB suite [7] available in the comparing continuous optimizers (COCO) platform [6] and iterative optimization heuristics profiler tool (IOHProfiler) [4]. Altogether, the BBOB suite consists of 24 single-objective, continuous, and noiseless functions of different optimization landscape complexity, which we refer to this suite as *the* BBOB.

Principally, the BBOB functions can be scaled up to arbitrary dimensionality and different problem instances can be created through transformations of the search space and objective values, which is controlled by an internal identifier. Typically, investigations based on the BBOB suite are carried out within the box-constrained search space $[-5, 5]^d$, where the global optimum is located within $[-4, 4]^d$, where d represents the dimensionality. Extensive analysis of the BBOB problem instances is available in [17].

To complement the diversity of the BBOB suite, additional functions can be generated via affine combination of two BBOB functions [3], which is based on an interpolation between two selected BBOB functions and uses a weighting factor to control the shifting between functions. This approach was later generalized to affine combinations of many BBOB functions, also known as many-affine BBOB (MA-BBOB) functions, where the affine combination is no longer limited to only two functions [38,39].

2.3 Randomly Generated Functions

Apart from the benchmarking suites, a set of functions can be generated using the function generator proposed in [36], covering a diverse set of optimization problem classes, as shown in [40]. By using a set of selection pressures, mathematical operands and operators are randomly selected from a predefined pool to construct tree-structured function expressions, which we call RGF. In fact, RGF with similar landscape characteristics to automotive crashworthiness optimization problems can be created, which is lacking in the BBOB suite [15].

Nonetheless, the properties of RGF are not known a priori, e.g., the global optimum and optimization complexity, as oppose to the well-studied BBOB suite. To tackle this problem, an extension has been attempted on this function generator to guide the function generation towards specific optimization complexity using genetic programming [16], which is beyond the scope of this work.

2.4 Exploratory Landscape Analysis

In landscape-aware ASP, exploratory landscape analysis (ELA) is one of the popular approaches employed to numerically quantify the high-level landscape characteristics of continuous optimization problems, such as multi-modality and global structure. While initially only six fundamental feature classes were proposed in ELA, namely y-distribution, level sets, meta-models, local searches,

curvature, and convexity [21,22], more feature classes have been progressively proposed to complement them, e.g., dispersion, nearest better clustering (NBC), principal component analysis (PCA), linear models, and information content of fitness sequences (ICoFiS) [10,12,18,24].

In brief, a design of experiments (DoE) is required for the ELA features computation, consisting of some samples $\mathcal{X} = \{x_1, \cdots, x_n\}$ and objective values $\mathcal{Y} = \{y_1, \cdots, y_n\}$, which are computed using an objective function f, i.e., $f\colon \mathcal{X} \to \mathcal{Y}$, where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, and n is the sample size. Consequently, the effectiveness of ELA features can be dependent on the DoE sample size, dimensionality, and sampling strategy [29]. To overcome potential bias of the hand-crafted ELA features in capturing specific landscape characteristics, deep NN-based methods have been explored to characterize BBO problems based on latent space features, e.g., DoE2Vec [32], which we leave for future work.

3 Methodology

The workflow of our landscape-aware AAC approach is visualized in Fig. 1. In the first step, the landscape characteristics of RGF are captured using ELA and the corresponding best configurations are identified using HPO during the training phase. Next, the ELA features and configurations are properly pre-processed for the training of NN models. Eventually, optimal configurations for unseen BBO problems can be predicted based on their ELA features using the trained models. Our approach is described in detail in the following.

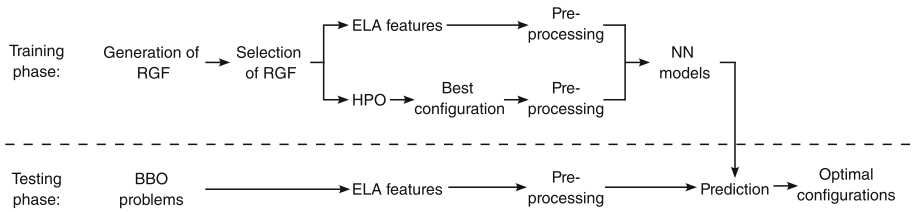


Fig. 1. An overview of our proposed landscape-aware AAC approach that can identify optimal configurations for BBO problems, consisting of a training and testing phase. During the training phase, using a preferably large set of RGF, the respective ELA features and optimal configurations identified through HPO (performed on RGF) are utilized to train NN models. The pre-trained models can then be deployed to predict the best suited configuration for unseen BBO problems based on their ELA features in the testing phase.

Generation and Selection of RGF. Firstly, a large set of RGF is generated for the training of NN models, using the function generator implemented in [15]. Before the model training, a pre-selection step is integrated to identify RGF that are appropriate for AAC purposes, refer to Sect. 3.1 for detailed explanations.

Computation of ELA Features. Secondly, the optimization landscape characteristics of RGF are computed using ELA based on some DoE samples. To combat inherent bias [27], the objective values are normalized using min-max scaling before the ELA features computation. Since many of the ELA features are redundant [30], highly correlated ELA features based on Pearson's correlation coefficient (>0.95) are discarded, using a similar approach as in [15]. To improve the performance of NN models, we ensure that the remaining ELA features are within a comparable scale range via normalization using min-max scaling.

Identifying the best Configuration using HPO. For each individual RGF, the best performing algorithm configuration found using HPO is considered as the best suited configuration identified for that function. Similar to the ELA features, the configuration data are pre-processed for the model training, where categorical hyperparameters are one-hot encoded, while continuous hyperparameters are linearly re-scaled to the scale range of $[0, 1]$ using Eq. 1.

$$z_{new} = \frac{z_{init} - a_{min}}{a_{max} - a_{min}} \cdot (b_{max} - b_{min}) + b_{min}\,, \tag{1}$$

where z_{init} and z_{new} are the initial and re-scaled values, a_{max} and a_{min} are the lower and upper bound before re-scaling, and b_{max} and b_{min} are the lower and upper bound after re-scaling. In this work, we focus on finding the best configuration of modular CMA-ES.

Training of NN Models. For the training of NN models, the pre-processed ELA features are employed as input, while the best configurations identified using HPO as output. Detailed descriptions of the NN models are included in Sect. 3.2.

Optimal Configurations for BBO Problems. During the deployment or testing phase, the trained NN models can be used to predict optimal configurations of modular CMA-ES for unseen BBO problems based on their ELA features. Similar to the training phase, the input ELA features of BBO problems are normalized, while the predicted configurations are inversely transformed back to the original configuration search space. To avoid invalid configurations, e.g., negative population size, predicted continuous hyperparameters that fall outside the search space will be set to either the lower or upper boundary.

3.1 Selection of Appropriate RGF

Unlike the well understood BBOB functions, the landscape characteristics and global optimum of RGF are not known a priori. Due to the fact that some RGF are insufficiently discriminative in distinguishing different configurations based on their optimization performances, not all RGF are appropriate for AAC purposes based on our preliminary testing. Using the HPO results on three chosen RGF in Fig. 2 as an example, we consider functions with a similar pattern to

'RGF1' as ideal for AAC purposes, where a clear configuration ranking with only a few ties is possible. More importantly, the best configuration can be easily identified. On the other hand, functions similar to 'RGF2' are considered as inappropriate for AAC, where many, or in extreme situations, all configurations are equally good, leading to an ambiguous ranking. We suspect that the optimization complexity of such RGF is too low that the choice of configuration does not matter. Surprisingly, two RGF with a small difference in their ELA features can have the opposite patterns, which raises questions for future research. To improve the robustness of trained models, functions similar to 'RGF3' are additionally neglected, where the global optimum seems to be an extreme outlier and can be found occasionally by a few configurations.

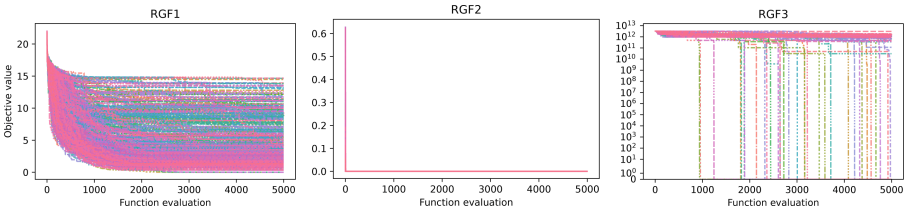


Fig. 2. The optimization convergence of 500 configurations evaluated using HPO on three chosen RGF. The x-axis shows the number of function evaluations, while the y-axis shows the re-scaled objective values, with 0 being the best solution found in all runs. Each curve represents a configuration run using modular CMA-ES (median over 10 repetitions). (*Left*) Ideal for AAC purposes, where a clear ranking of configurations is possible. (*Middle*) Ambiguous ranking of algorithm configurations, where all configurations are equally competitive. (*Right*) The global optimum seems to be an outlier that can only be found by a few configurations.

To overcome these problems, the following measures are implemented to identify RGF that are appropriate for AAC purposes.

1. **Estimation of global optimum:** In a brute-force manner, we perform HPO on each RGF, focusing on finding a better solution, i.e., a smaller objective value, and using a similar setup as described in Sect. 4. Eventually, the global optimum y_{opt} is approximated based on the best solution found in all HPO runs y_{hpo} using Eq. 2.

$$y_{opt} = \begin{cases} \lfloor y_{hpo} \rfloor, & \text{if } 0 \leq |y_{hpo}| < 10 \\ \lfloor y_{hpo}/10 \rfloor \cdot 10, & \text{if } 10 \leq |y_{hpo}| < 100 \\ \lfloor y_{hpo}/10^p \rfloor \cdot 10^p, & \text{otherwise} \end{cases}, \qquad (2)$$
$$p = \lfloor \log_{10} |y_{hpo}| \rfloor - 1,$$

where y_{hpo} is either rounded to the nearest lower integer for a small $|y_{hpo}|$, or rounded based on the nearest lower power of 10. Having an estimated global

optimum for RGF is essential in our approach to facilitate an evaluation of configuration performance (refer to Sect. 4.1) and a comparison between different functions with varying scale ranges.
2. **RGF appropriate for AAC:** Using the same HPO results from previous step, all configurations evaluated are ranked according to their performances, where ties are assigned with the same rank. The ranking ambiguity is evaluated based on the Kendall rank correlation coefficient between the HPO configuration ranking and a strict ranking (without tie). For a correlation lower than 0.9, e.g., due to too many ties, such ranking is considered as ambiguous. Furthermore, we compute the standard score or z-score of the global optimum found to estimate its deviation from the distribution of other solutions. When the global optimum is 3 standard deviations away from the distribution mean, it is considered as an extreme outlier.
3. **Elimination of RGF:** A RGF is excluded from the training data, if any of the aforementioned conditions is fulfilled.

While additional computational effort is required for the above-mentioned measures in identifying RGF appropriate for AAC purposes, we argue that they are critical in improving the performance of NN models. Moreover, this process needs to be done only once, since the RGF identified can be re-used in the future for the same BBO problem classes.

3.2 Multi-output Mixed Regression and Classification

Dense Neural Network: In this work, we investigate the potential of dense NN models with the following architecture for the multi-output mixed regression and classification tasks in landscape-aware AAC, as visualized in Fig. 3.

- **Input layer:** The size of the input layer is equal to the number of ELA features available in the training data.
- **Hidden layers:** To determine an optimal inner architecture, different combinations of number of hidden layer $\{1, 2, 3\}$, hidden layer sizes $\{16, 32, 64, 128\}$, and epochs $\{100, 150, 200\}$ are evaluated using a grid search approach, $80 : 20$ train-test split of the training data, and a repetition of five times. Eventually, the hidden layers are constructed based on the combination with the smallest validation loss and assigned with rectified linear unit (ReLU) as activation function.
- **Output layers:** In short, different layers are assigned for the mixed regression and classification tasks. While a single output layer with linear activation function is dedicated for the multi-output regression task, the multi-output multi-class classification task is split into multiple classifications tasks. Precisely, an output layer with softmax activation function is allocated for each categorical hyperparameter. Consequently, the size of each output layer depends on the number of hyperparameters respectively.

– **Loss functions:** The dense NN models are trained using mean squared error as loss function for regression and categorical cross entropy for classification task.

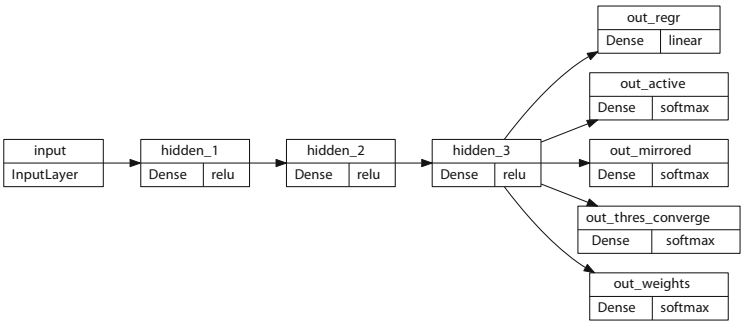


Fig. 3. An example of the architecture of a dense NN model. From left to right, an input layer, three hidden layers, and several output layers, with one output layer for regression and four layers for classification tasks.

Random Forest: For a fair evaluation, the performance of trained NN models is compared against RF models, which are popular in landscape-aware ASP. Precisely, the RF models are optimally constructed with fine-tuned configurations using `auto-sklearn` [5], an automated CASH tool designed for ML, and $80:20$ train-test split of the training data. Since multi-output multi-class classification is currently limited in `auto-sklearn`, the algorithm configuration problem is defined as a multi-target regression task, where the categorical hyperparameters are encoded as numerical labels.

4 Experimental Setup

In brief, the scope of our investigations can be summarized as follows:

– In $5d$, using a set of 1 000 RGF as training data, while the 24 BBOB functions of the first instance as unseen test problems. For a comprehensive analysis, we also investigate models trained using 1 000 MA-BBOB functions and a combination of both RGF and MA-BBOB functions;
– An optimization landscape is characterized based on a total of 68 ELA features that can be computed without requiring additional function evaluations, using a DoE of $50 \cdot d$ samples, `pflacco` [28], and a similar workflow proposed in [15];
– In this work, we consider fine-tuning the configuration of modular CMA-ES within the configuration search space in Table 1, with all optimization runs are allocated with a budget of $1\,000 \cdot d$ evaluations and 10 repetitions; and

Table 1. An overview of the 11 hyperparameters of modular CMA-ES considered for AAC. The default configuration is highlighted in bold, where the default learning rates are automatically determined based on other hyperparameters. The 'number of children' predicted by predictive models is rounded-off to integer. Symbol: $\mathbb{Z}$ for integer, $\mathbb{R}$ for continuous variable, and **C** for categorical variable.

Num.	Hyperparameter	Type	Domain
1	Number of children	$\mathbb{Z}$	{ 5, . . ., 50 } (**4** + $\lfloor$(**3** · **ln**(***d***))$\rfloor$)
2	Number of parent (as ratio of children)	$\mathbb{R}$	[0.3, 0.5] (**0.5**)
3	Initial standard deviation	$\mathbb{R}$	[0.1, 0.5] (**0.2**)
4	Learning rate step size control	$\mathbb{R}$	[0.0, 1.0]
5	Learning rate covariance matrix adaptation	$\mathbb{R}$	[0.0, 1.0]
6	Learning rate rank-μ update	$\mathbb{R}$	[0.0, 0.35]
7	Learning rate rank-one update	$\mathbb{R}$	[0.0, 0.35]
8	Active update	**C**	{ True, **False** }
9	Mirrored sampling	**C**	{ **none**, 'mirrored', 'mirrored pairwise' }
10	Threshold convergence	**C**	{ True, **False** }
11	Recombination weights	**C**	{ '**default**', 'equal', '1/2^lambda' }

– The TPE available in `HyperOpt` [2] is employed to identify optimal configurations of modular CMA-ES and assigned with a budget of 500 evaluations.

To analyze the performance of our approach for BBO problems in higher dimensionality, our investigations are extended to $20d$ using a smaller experimental scope to minimize computational effort, namely a DoE of $20 \cdot d$ samples for ELA features computation, $100 \cdot d$ evaluations for optimization runs, 300 evaluations for TPE, and only the seven real-valued hyperparameters of modular CMA-ES are considered.

4.1 Optimization Performance Metric

For real-world applications, (i) it is often practical to find good solutions within a shorter time, rather than finding the global optimum, and (ii) the global optimum is usually not known, making it difficult to use some popular performance metrics, e.g., expected hitting time [37]. Hence, we propose to measure the performance of a configuration based on its area under the curve (AUC) of optimization convergence (Fig. 2). By minimizing the AUC metric, we are essentially searching for configurations that have an optimal trade-off between the solution found and convergence speed. In this work, all AUC during HPO are computed using the min-max normalized objective values based on the global optimum and worst DoE sample.

4.2 Optimization Baseline

Principally, we consider the following three algorithm configurations as comparison reference to evaluate the potential of our approach.

- **Default configuration:** The readily available configuration in its original implementation that is simply utilized by practitioners with limited experience in fine-tuning configurations. Inline with our motivation, our approach is primarily compared against it.
- **SBS:** The configuration that can perform well on average across all 24 BBOB functions and serves as our secondary target to beat in this work. Precisely, it is identified based on the mean performance of configurations evaluated across all BBOB functions.
- **Virtual best solver (VBS):** The best performing configuration for a particular BBOB function, which can be treated as the lower bound.

Unlike typical ASP approaches, where the SBS and VBS are selected from a portfolio of limited algorithms using grid search, evaluating all possible configurations within the large search space in Table 1 is computationally infeasible. Subsequently, we determine both solvers via HPO using TPE within an allocated budget. Due to the stochastic nature of TPE, there might be configurations that can outperform the VBS identified, but are not discovered during HPO.

5 Results

Due to the limited space, experimental results and figures not included in this paper can be found in our repository at https://doi.org/10.5281/zenodo.10965507.

5.1 Representativeness of Training Data

Before delving into analyzing the configuration performances, we take a closer look at the representativeness of training data. Naturally, predictive models trained using MA-BBOB functions are expected to perform well, since the problem classes available in the training data should sufficiently cover the BBOB suite. While this can be observed most of the time, it is not always the case, notably for F7 (step ellipsoidal) and F12 (Bent Cigar) in Sect. 5.2. The poor performances could be due to the insufficient coverage of ELA feature space by MA-BBOB functions, as shown in Fig. 4, which might be related to the generation of MA-BBOB functions [38]. In comparison, RGF can better cover the ELA feature space, highlighting the benefits of using RGF as training data. In fact, it seems to be advantageous to combine the large distribution of RGF and the more focused distribution of MA-BBOB on some of the BBOB functions.

5.2 Performance of Predicted Configurations

The optimization performances using different configurations for 24 BBOB functions in $5d$ are compared in Fig. 5. In general, the optimal configurations identified using predictive models can clearly outperform the default configuration on most BBOB functions. On the other hand, the predicted configurations seem to

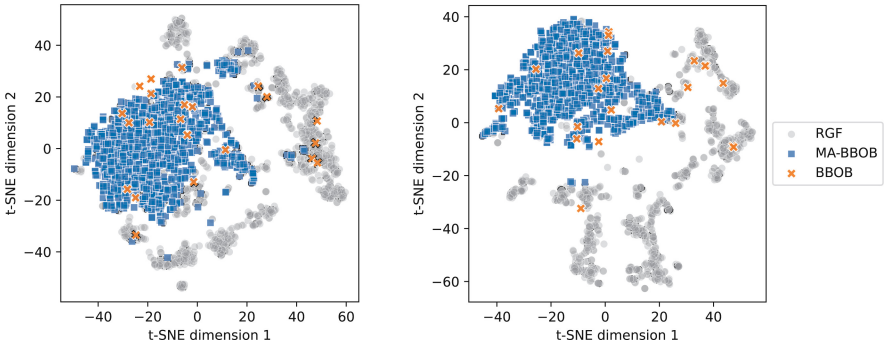


Fig. 4. Projection of the ELA feature space to a 2d visualization using t-distributed stochastic neighbor embedding (t-SNE) [19] for 1 000 RGF, 1 000 MA-BBOB, and 24 BBOB functions in 5d (*left*) and 20d (*right*), using a similar approach as in [15].

be competitive against the SBS, such as for F7 and F17 (Schaffers F7). Not only that, our approach using NN models can perform better than the SBS in some cases, for instance, for F5 (linear slope) and F13 (sharp ridge). Nonetheless, the performance of predicted configurations is lacking for highly multi-modal functions, e.g., F16 (Weierstrass) and F23 (Katsuura), which might be due to the absence of ELA features that can accurately capture the landscape characteristics of such complex functions, revealing the weaknesses in our approach. When compared against the VBS, the predicted configurations sometimes seem to have a comparable performance, e.g., for F21 (Gallagher's Gaussian 101-me peaks).

Using the Wilcoxon signed-rank test with the hypothesis *optimal configurations identified using our approach can perform better*, we statistically evaluate the performance of different configurations. Precisely, we focus on comparing NN models against the default configuration, SBS, and RF models, using RGF as training data. Inline with our previous observations, optimal configurations predicted using our approach can indeed beat the default configuration for most BBOB functions, while outperforming the SBS on many BBOB functions, as depicted in Fig. 6. It is worth reminding that our approach can be competitive against the default configuration and SBS in a few remaining BBOB functions, as previously discussed in Fig. 5. This analysis also indicates that our current approach is more effective on simple functions (first half of the BBOB suite) compared to complex functions (second half), which might be related to the ELA features. Apart from that, the performances of NN models are as good as or even better than RF models for some BBOB functions, particularly in 5d.

As illustrated in Fig. 6, we can in general have similar observations for the BBOB functions in 20d. When compared to the default configuration, our approach are more effective for many BBOB functions. Nevertheless, the performance improvements gained using our approach compared to the SBS in 20d are less than in 5d, showing rooms for improvement in high dimensionality.

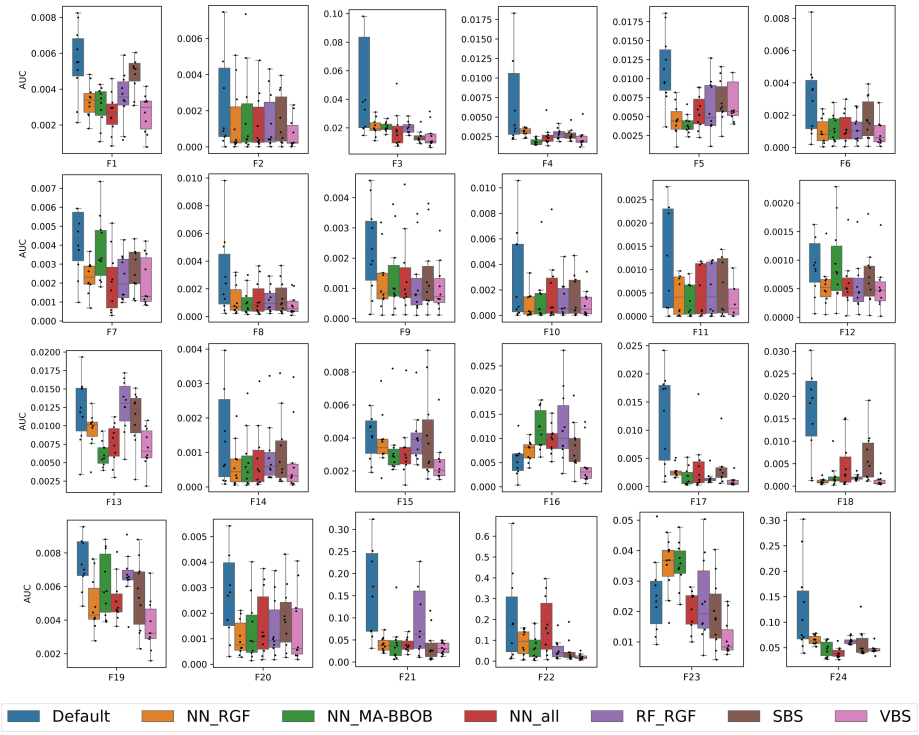


Fig. 5. Performance of modular CMA-ES using different configurations for 24 BBOB functions in 5d, each repeated for 10 times. The AUC is computed based on objective values min-max normalized using the global optimum and worst solution in all configurations, divided by the evaluation budget. A lower AUC is better.

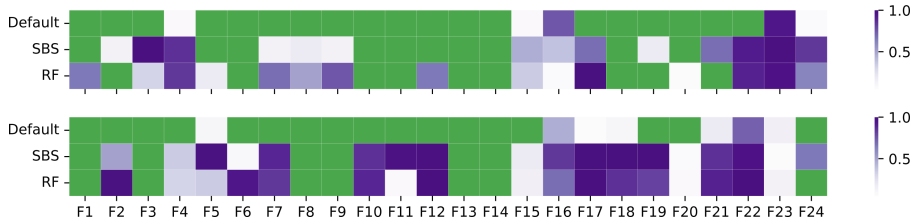


Fig. 6. Pairwise performance comparison between the configuration predicted using NN models (our approach) against the default configuration, SBS, and RF models for 24 BBOB functions in 5d (*top*) and 20d (*bottom*) based on the p-value computed using the Wilcoxon signed-rank test. The green color indicates that there is statistically significant evidence to support the hypothesis *optimal configurations predicted using NN models can perform better*, with a p-value smaller than 0.05. Alternatively, a darker purple color (larger p-value) indicates that the hypothesis is more likely to be rejected, while a lighter purple color (smaller p-value) for a lower chance of rejection.

6 Conclusions and Future Work

Aiming to assist practitioners unfamiliar with fine-tuning of algorithm configurations, we propose to construct general purpose predictive models towards landscape-aware AAC that can identify optimal algorithms as well as hyperparameters for different practical applications. To improve the generalization of our approach, we consider tree-based RGF as training data, which covers a diverse set optimization problem classes. Furthermore, a pre-selection step is implemented to select RGF that are appropriate for AAC purposes, and thus, to improve the prediction accuracy. Moreover, we investigate the potential of dense NN models for the multi-output mixed regression and classification tasks, which can easily handle the mixed-integer search space and large training data sets.

When evaluated on the BBOB suite in $5d$ and $20d$ using modular CMA-ES, our results reveal that we can predict near-optimal configurations that outperform the default configuration and compete against the SBS in most cases. This is particularly encouraging for real-world applications, where such a SBS is usually not available. In fact, properly selected RGF have promising potential as training data for landscape-aware AAC, since they cover a broader spectrum of function complexity compared to BBOB and MA-BBOB functions. Subsequently, we believe that our approach can generalize well beyond the BBOB suite, provided that the unseen problems is well represented by the RGF training set. Overall, configurations with better performance can be best identified using dense NN models trained on a combination of RGF and MA-BBOB functions.

For future work, we plan to improve our investigations as follows:

- The configuration search space can be expanded to include a variety of optimization algorithms and hyperparameters;
- The performance of NN models can be further improved by fine-tuning more hyperparameters using an optimizer, e.g., learning rate and batch size;
- An analysis can be extended to better understand the impact of ELA features pre-processing, e.g., using normalization vs. standardization;
- To further minimize the overall computational costs, alternatives that can efficiently identify RGF appropriate for AAC purposes can be explored;
- Despite the fact that the estimated y_{opt} seems to be robust in our work, i.e., always smaller than all solutions found, further investigations are needed for confirmation and/or improvements; and
- Eventually, we aim to evaluate and quantify the benefits of our approach for real-world expensive BBO problems.

Acknowledgments. The contribution of this paper was written as part of the joint project newAIDE under the consortium leadership of BMW AG with the partners Altair Engineering GmbH, divis intelligent solutions GmbH, MSC Software GmbH, Technical University of Munich, TWT GmbH. The project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag.

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

References

1. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, vol. 24 (2011)
2. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: International Conference on Machine Learning, pp. 115–123. PMLR (2013)
3. Dietrich, K., Mersmann, O.: Increasing the diversity of benchmark function sets through affine recombination. In: Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, 10–14 September 2022, Proceedings, Part I, pp. 590–602. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_41
4. Doerr, C., Wang, H., Ye, F., van Rijn, S., Bäck, T.: IOHprofiler: a benchmarking and profiling tool for iterative optimization heuristics. arXiv e-prints:1810.05281 (2018). https://arxiv.org/abs/1810.05281
5. Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., Hutter, F.: Auto-Sklearn 2.0: Hands-free AutoML via meta-learning. J. Mach. Learn. Res. **23**(261), 1–61 (2022)
6. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. Optim. Methods Softw. **36**(1), 114–144 (2021). https://doi.org/10.1080/10556788.2020.1808977
7. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter black-box optimization benchmarking 2009: noiseless functions definitions. Research Report RR-6829, INRIA (2009). https://hal.inria.fr/inria-00362633
8. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 312–317. IEEE (1996)
9. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
10. Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: Detecting funnel structures by means of exploratory landscape analysis. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 265–272. GECCO 2015, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2739480.2754642
11. Kerschke, P., Trautmann, H.: Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. Evol. Comput. **27**(1), 99–127 (2019). https://doi.org/10.1162/evco_a_00236
12. Kerschke, P., Trautmann, H.: comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. In: Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., Vichi, M. (eds.) Applications in Statistical Computing. SCDAKO, pp. 93–123. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25147-5_7
13. Lindauer, M., et al..: SMAC3: a versatile bayesian optimization package for hyperparameter optimization. J. Mach. Learn. Res. **23**(54), 1–9 (2022). http://jmlr.org/papers/v23/21-0888.html

14. Long, F.X., van Stein, B., Frenzel, M., Krause, P., Gitterle, M., Bäck, T.: Learning the characteristics of engineering optimization problems with applications in automotive crash. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1227-1236. GECCO 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3512290.3528712
15. Long, F.X., van Stein, B., Frenzel, M., Krause, P., Gitterle, M., Bäck, T.: Generating cheap representative functions for expensive automotive crashworthiness optimization. ACM Trans. Evol. Learn. Optim. **4**(2) (2024). https://doi.org/10.1145/3646554
16. Long, F.X., Vermetten, D., Kononova, A., Kalkreuth, R., Yang, K., Bäck, T., van Stein, N.: Challenges of ELA-guided function evolution using genetic programming. In: Proceedings of the 15th International Joint Conference on Computational Intelligence - Volume 1: ECTA, pp. 119–130. INSTICC, SciTePress (2023). https://doi.org/10.5220/0012206200003595
17. Long, F.X., Vermetten, D., van Stein, B., Kononova, A.V.: BBOB Instance Analysis: Landscape Properties and Algorithm Performance Across Problem Instances. In: Correia, J., Smith, S., Qaddoura, R. (eds.) Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, 12-14 April 2023, Proceedings, pp. 380–395. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-30229-9_25
18. Lunacek, M., Whitley, D.: The dispersion metric and the CMA evolution strategy. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 477–484. GECCO 2006, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1143997.1144085
19. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(86), 2579–2605 (2008). http://jmlr.org/papers/v9/vandermaaten08a.html
20. Malan, K.M.: A survey of advances in landscape analysis for optimisation. Algorithms **14**(2), 40 (2021). https://doi.org/10.3390/a14020040
21. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 829–836. GECCO 2011, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2001576.2001690
22. Mersmann, O., Preuss, M., Trautmann, H.: Benchmarking evolutionary algorithms: towards exploratory landscape analysis. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 73–82. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_8
23. Mockus, J.: The Bayesian approach to global optimization. In: Drenick, R.F., Kozin, F. (eds.) System Modeling and Optimization, pp. 473–481. Springer, Heidelberg (1982). https://doi.org/10.1007/BFb0006170
24. Muñoz, M.A., Kirley, M., Halgamuge, S.K.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE Trans. Evol. Comput. **19**(1), 74–87 (2015). https://doi.org/10.1109/TEVC.2014.2302006
25. Muñoz, M.A., Sun, Y., Kirley, M., Halgamuge, S.K.: Algorithm selection for black-box continuous optimization problems: a survey on methods and challenges. Inf. Sci. **317**, 224–245 (2015). https://doi.org/10.1016/j.ins.2015.05.010
26. de Nobel, J., Vermetten, D., Wang, H., Doerr, C., Bäck, T.: Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1375–1384. GECCO 2021, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3449726.3463167

27. Prager, R.P., Trautmann, H.: Nullifying the inherent bias of non-invariant exploratory landscape analysis features. In: Correia, J., Smith, S., Qaddoura, R. (eds.) Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, 12–14 April 2023, Proceedings, vol. 13989, pp. 411–425. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30229-9_27
28. Prager, R.P., Trautmann, H.: Pflacco: Feature-based landscape analysis of continuous and constrained optimization problems in Python. Evol. Comput., 1–25 (2023). https://doi.org/10.1162/evco_a_00341
29. Renau, Q., Doerr, C., Dreo, J., Doerr, B.: Exploratory landscape analysis is strongly sensitive to the sampling strategy. In: Bäck, T., et al. (eds.) PPSN 2020. LNCS, vol. 12270, pp. 139–153. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58115-2_10
30. Renau, Q., Dreo, J., Doerr, C., Doerr, B.: Expressiveness and robustness of landscape features. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 2048–2051. GECCO 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319619.3326913
31. Simoncini, D., Barbe, S., Schiex, T., Verel, S.: Fitness landscape analysis around the optimum in computational protein design. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 355–362. GECCO 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3205455.3205626
32. van Stein, B., Long, F.X., Frenzel, M., Krause, P., Gitterle, M., Bäck, T.: DoE2Vec: deep-learning based features for exploratory landscape analysis. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, pp. 515–518. GECCO 2023 Companion, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3583133.3590609
33. van Stein, N., Vermetten, D., Kononova, A.V., Bäck, T.: Explainable benchmarking for iterative optimization heuristics. arXiv preprint arXiv:2401.17842 (2024). https://arxiv.org/abs/2401.17842
34. Thomaser, A., Kononova, A.V., Vogt, M.E., Bäck, T.: One-shot optimization for vehicle dynamics control systems: towards benchmarking and exploratory landscape analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 2036–2045. GECCO 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3520304.3533979
35. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 847–855. KDD 2013, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2487575.2487629
36. Tian, Y., Peng, S., Zhang, X., Rodemann, T., Tan, K.C., Jin, Y.: A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks. IEEE Trans. Artif. Intell. **1**(1), 5–18 (2020). https://doi.org/10.1109/TAI.2020.3022339
37. Vermetten, D., Wang, H., Doerr, C., Bäck, T.: Integrated vs. sequential approaches for selecting and tuning CMA-ES variants. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, p. 903–912. GECCO 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3377930.3389831

38. Vermetten, D., Ye, F., Bäck, T., Doerr, C.: MA-BBOB: a problem generator for black-box optimization using affine combinations and shifts (2023). https://arxiv.org/abs/2312.11083
39. Vermetten, D., Ye, F., Doerr, C.: Using Affine Combinations of BBOB Problems for Performance Assessment. CoRR **abs/2303.04573** (2023). https://doi.org/10.48550/arXiv.2303.04573
40. Škvorc, U., Eftimov, T., Korošec, P.: A complementarity analysis of the COCO benchmark problems and artificially generated problems. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, p. 215–216. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3449726.3459585
41. Zhao, M., Li, J.: Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators. In: 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI), pp. 613–618 (2018). https://doi.org/10.1109/ICACI.2018.8377530
42. Škvorc, U., Eftimov, T., Korošec, P.: Transfer learning analysis of multi-class classification for landscape-aware algorithm selection. Mathematics **10**(3) (2022). https://doi.org/10.3390/math10030432, https://www.mdpi.com/2227-7390/10/3/432

Feature Encapsulation by Stages in the Regression Domain Using Grammatical Evolution

Darian Reyes Fernández de Bulnes[1(✉)], Allan de Lima[1], Edgar Galván[2], and Conor Ryan[1]

[1] University of Limerick, Limerick, Ireland
{darian.reyesfernandezdebulnes,allan.delima,conor.ryan}@ul.ie
[2] Maynooth University, Maynooth, Ireland
edgar.galvan@mu.ie

Abstract. Feature Encapsulation by Stages (FES) is a recently proposed mechanism that can be implemented in any Evolutionary Computation (EC) metaheuristic. Encapsulation occurs via input space expansion in several stages by adding the best individual so far as an additional input. FES has been shown to perform well in training Boolean problems. This paper extends FES to the regression domain. Grammatical Evolution (GE), a branch of Genetic Programming (GP), supports the implementation of the FES approach by enabling the investigation of performance across various search guides expressed in the grammar. We conduct experiments on both synthetic and real-world symbolic regression problems, including multi-target issues. Additionally, we study several FES-based approaches utilising the best selection process for each problem, choosing between tournament, ϵ-Lexicase, and ϵ-Lexi^2. Statistical tests on unseen subsets' results show that FES outperforms the standard baseline in all problems. Furthermore, we analyse individual complexity across generations, showing that populations utilising FES consist of simpler individuals, thereby reducing computational costs.

Keywords: ϵ-Lexi^2 · Feature Encapsulation by Stages · Grammatical Evolution · Regression

1 Introduction

Feature Encapsulation by Stages (FES) [19] is a novel mechanism for encapsulating and transferring features from a short evolutionary run (referred to as a *stage*) as knowledge into a subsequent stage. It has shown considerable performance improvements by combining the original dataset with encapsulated extra features to obtain a joint solution that enhances the standard baseline, even when considering various selection processes. Encapsulation occurs through the expansion of the input space during the transition from one stage to the next, managed by a grammar. At the beginning of each stage, the entire population

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 105–120, 2024.
https://doi.org/10.1007/978-3-031-70068-2_7

is re-initialised, which allows the emergence of individuals that reuse these new features and avoid premature convergence.

FES has been demonstrated to perform well in training for the Boolean domain [19], and this paper extends its application to generalise on unseen test sets in the regression domain. Here, we hypothesise that the trained and validated encapsulated features, which may exhibit a bias towards the seen dataset, play a crucial role in addressing the unseen test dataset and statistically outperform the standard evolutionary process.

To implement and explore the methods, we use Grammatical Evolution (GE), which evolves programs in any Backus-Naur Form compliant language [21]. We use GE for its demonstrated capabilities in the Regression domain [5]. Furthermore, variants of Lexicase selection [22] for regression problems have shown promising results [10] compared to error-aggregating parent selection methods such as Tournament selection [4]. Recently, several bloat control strategies have been proposed, such as Lexicographic parsimony pressure [12] for Lexicase selection in Genetic Programming (GP) [8]. The framework for this paper combines all of these procedures. Therefore, we first conduct experiments to compare the GE standard evolutionary process (our baseline) using tournament, ϵ-Lexicase, and ϵ-Lexi^2 to tackle the studied problems. Later, we conduct the FES experiments only with the best overall selection process to compare it with the standard baseline. Comparisons are made regarding performance on the test set and measuring the complexity of the individuals.

Complexity is defined by measuring the number of utilised nodes. In GE, the number of nodes represents the number of utilised terminals in the phenotype, influencing both the mapping and evaluation processes. Thus, this metric directly correlates to the execution time in GE.

The main contributions of this paper are:

- Presentation of a new variant of FES to generalise on unseen test sets by encapsulating the output of the best-trained and best-validated individuals;
- Examination of FES methods with the ϵ-Lexi^2 selection process for bloat control to address regression problems;
- Comparison of FES methods using synthetic and real-world symbolic regression problems;
- Analysis of the complexity of individuals across generations reducing computational costs.

In Sect. 2, we discuss related work. Following that, in Sect. 3, we describe the proposed methodology FES in detail. Following that, Sect. 4 shows the experimental setup for GE, while Sect. 5 contains the results and discussion. Finally, conclusions and future directions are outlined in Sect. 6.

2 Background

Various strategies have been studied in the fields of Machine Learning (ML) and optimisation to expand the input space or reuse substructures. This research is influenced by strategies within and outside of Evolutionary Computation (EC).

One of the pioneering works to leverage the reuse of substructures was Koza et al. [9]. Although this research is specifically related to the circuit problem domain, its foundation lies in the concept that solving a problem necessitates multiple applications of the same sophisticated submodules, all of which should be designed for reusability.

2.1 Evolutionary Computation

Much of the prior related work has been conducted within the realm of EC, predominantly in the domain of GP. Keijzer et al. [7] introduced the concept of Run Transferable Libraries (RTL). These were a mechanism for transferring knowledge obtained in one GP run to another, rather than relying solely on generational approaches. The primary aim was to address challenges associated with highly scalable problems.

Some years later, Medernach et al. [16] introduced Wave, an approach that divides the GP run into several periods. This proposal had the peculiarity of re-initialising the population at the beginning of each period. However, the drawback of this proposal is that it can only be applied to problems where the objective function can be decomposed into sub-functions for each period. Much recent work has also focused on evolving features for reuse using Multidimensional Multiclass Genetic Programming (M3GP) by Batista et al. [2]. M3GP was designed to address multiclass classification problems by enriching the representation of the solution space in higher dimensions with a predictive algorithm, such as the Mahalanobis distance classifier. More recently, Murphy et al. [17] proposed a method for GE mapping to identify valuable modules that comprise a solution.

2.2 Stacked Methods

Over the last two decades, alongside developments in EC, stacked ML techniques for regression or classification tasks with multiple outputs have emerged. For example, Godbole and Sarawagi [3] exemplify the importance of feature combination in addressing multi-label classification text tasks, revealing connections between classes. Similarly, Spyromitros et al. [23] proposed a strategy based on input space expansion for multi-target regression problems. They examined the relationship between the outputs and assessed how their estimation contributes to computing the actual outputs in a subsequent stage.

Liu et al. [13] also analysed multi-output regression problems. They introduced a framework based on Gaussian methods, investigated knowledge transfer among correlated outputs to enhance the quality of predictions, and contrasted their proposal with the 2-stage-based Stacked Single-Target (SST). A couple of years later, Mastelini et al. [14] presented Deep Structure for Tracking Asynchronous Regressor Stacking (DSTARS) as an extension of SST by combining multiple stacked regressors into a deep structure using a validation sub-procedure. Additionally, Xia et al. [25] directed their attention to multi-label classification problems, examining a weighted ensemble approach to tackle problems involving up to 75 labels and over a thousand features. However, all these

methods are highly susceptible to premature convergence, which we identify as one of their main drawbacks.

3 Feature Encapsulation by Stages

The transfer of prior knowledge via feature encapsulation and the re-initialisation of the population, thereby avoiding premature convergence and destroying previously (evolved) individuals, is carried out in stages. This paper describes a method to augment a grammar to encapsulate fit individuals in a GE population. The population is re-initialised once the grammar has been augmented to utilise the encapsulated features. This process is referred to as a stage.

Furthermore, we introduce a novel FES variant called Training/Validation/Test (TVT), which includes a validation procedure to address model overfitting in the regression domain. Instead of solely encapsulating the outputs of the best-trained individual (Training/Test (TT) strategy), the outputs of the best-validated individual are also encapsulated. Compared to the previous TT strategy (without validation), this new TVT strategy doubles the number of additional inputs (extra features) facilitated by the GE grammar. The best-validated individual is selected to confront the test set once the maximum number of generations is reached.

A complete workflow for both strategies, TT and TVT, is presented in Fig. 1. It illustrates how, at the end of each stage, the outputs of the best individual in the population are transformed into encapsulated features for the subsequent stage. In the TT strategy, the best individual (depicted in green) refers to the best-trained individual, while the TVT strategy refers to the best-validated individual.

The process consists of a series of stages represented by large loops (depicted as curved arrows). Within each stage, Sensible Initialisation [20] and a specified number of generations occur. Sensible Initialisation is a modification of the ramped-half-and-half initialisation procedure created for GP. The first stage utilises the standard grammar, while subsequent stages employ the extended grammar. Upon reaching the maximum number of generations (stopping criterion), irrespective of the current stage, the test set is evaluated using the best individual in the population.

In contrast to the approach outlined in Medernach et al., [16], where the optimal solution involves combining the best individuals from each period to form a large structure, our methods maintain a hall of fame. FES involves preserving the best individual across generations throughout the stages without necessitating the artificial creation of a phenotype at the end of the evolutionary process. Moreover, unlike the procedure described in Spyromitros et al. [23], which only manages two offline stages, FES offers flexibility in the number of stages, automatically adjusting within the same evolutionary process. Within our approach, the input space expands only once throughout the evolution, with no increase in training instances. Hence, expanding the input space does not result in a significant rise in computational costs.

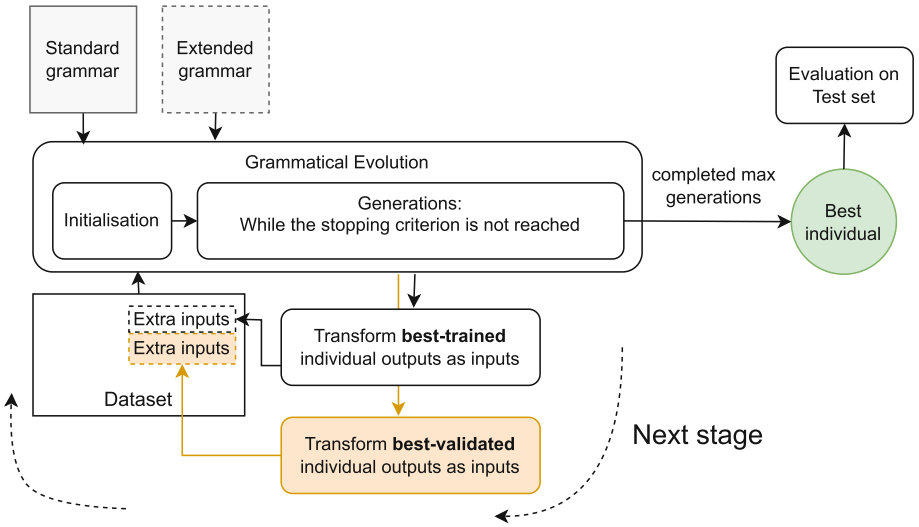


Fig. 1. Feature Encapsulation by Stages presenting two strategies: Training/Testing (TT) and Training/Validation/Testing (TVT). TT strategy does not include the elements in orange, which is the difference between both strategies. (Color figure online)

Furthermore, we present five different management approaches for FES to investigate the two strategies above:

- 2 stages 25%: two stages, with the second stage starting after 25% of the generations have elapsed;
- 2 stages 50%: the same as above, but the second stage starts after half of the generations have elapsed;
- 3 stages: the second stage starts after one-third of the generations have elapsed, while the third stage commences after two-thirds have elapsed;
- 4 stages: the second stage starts after 25% of the generations have elapsed, the third stage after 50% of the generations, and the final one after 75% of the generations;
- Multi-stage: a dynamic number of stages is launched, where a new stage starts every time the best fitness value changes from generation to generation. To avoid excessive triggering of stages in early generations with the Multi-stage approach, we always wait until at least 25% of the total generations have passed after the previous stage ended before starting a new stage; thus, the total number of stages ranges from one to four.

The first stage for all proposed methods is identical to the standard baseline, utilising the standard grammar specific to the problem. Once this initial stage is completed, the output vectors of both the best-trained and best-validated individuals are included as new inputs in the extended grammar, effectively creating new features.

To better understand and visualise the FES method, we present a hypothetical example in Fig. 2. This diagram illustrates how solutions are encapsulated in stages, utilising both original and encapsulated features from previous stages. The example highlights the distinction between nodes and equivalent nodes; a node in GE corresponds to a terminal, representing an original feature in this example. After the last stage (depicted in green), the best individual is selected for evaluation on the test set.

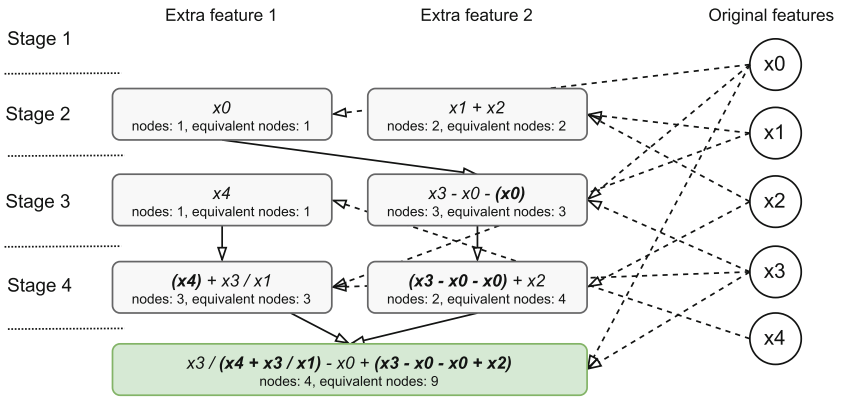


Fig. 2. Hypothetical example of a four-stage approach with five original and two additional features. Encapsulated features from a previous stage are indicated within parentheses and bold text. The best individual phenotype at the end is depicted in green, combining original features and the two additional features. (Color figure online)

It is also important to note that FES does not increase the number of evaluations (a critical subprocess in EC techniques) or the number of generations of the evolutionary process. This is the main reason why the comparison to determine the feasibility of FES is made against the baseline of the standard GE.

4 Experimental Design

This work is focused on the regression domain; for that reason, we select $RMSE$ as the fitness function:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(p_i - t_i\right)^2} \tag{1}$$

where n is the number of training cases, p the vector of predictions, and t the target vector.

Additionally, in this section, we define the six studied problems, GE as the EC framework for FES, and all relevant experimental parameters.

4.1 Problems

Six regression problems are studied, comprising two synthetic and four real-world problems, as listed in Table 1.

The two synthetic problems (Paige 1 and Vladislavleva 4) are recommended in [15]. In contrast, the four real-world problems are as follows:

- Concrete: This problem concerns the prediction of concrete compressive strength, a highly nonlinear function of age and ingredients [1] and is crucial in civil engineering.
- Energy Efficiency: This problem aims to assess buildings' heating and cooling load requirements [1].
- Boston Housing: This problem utilises the Boston house-price data for regression diagnostics [24].
- Tower: This problem involves 15-minute averaged time series data from a chemical distillation tower to predict propylene concentration [10].

Table 1. Regression problems in order by # features.

Problem	Type	# features	# outputs	# training cases	# test cases
Paige 1	Synthetic	2	1	676	10000
Vladislavleva 4	Synthetic	5	1	1024	5000
Concrete	Real-world	8	1	721	309
Energy Efficiency	Real-world	8	2	538	203
Boston Housing	Real-world	13	1	354	152
Tower	Real-world	25	1	3499	1500

The benchmarks are diverse, featuring 2 to 25 input features and varying numbers of training and test cases. Additionally, one of the problems involves multi-output. In the TVT strategy, where a validation set is required, 50% of the training cases are allocated to it. For example, in the case of Concrete problem, there are 361 cases for training, 360 cases for validation, and 309 cases for testing.

4.2 Grammatical Evolution

GE [21] is a grammar-based variant of GP that harnesses the power and flexibility of grammar to define and constrain the syntax of potential solutions. Like other EC techniques, GE evolves a population of solution candidates by adhering to the Darwinian principles of evolution. A modulo rule is used to dictate production rule selection in the grammar. The process involves expanding all non-terminals to terminal symbols. A valid program or phenotype is obtained once all non-terminals have been expanded to terminals. Operations within the grammar are safeguarded against well-known execution time errors to ensure

that all results remain within the realm of real numbers. For instance, Eq. 2 is employed to prevent division by zero.

$$AQ = \frac{a}{\sqrt{1 + b^2}} \tag{2}$$

where a is the dividend and b is the divisor.

As an example, the grammars for the Paige 1 problem in GE are presented in Table 2. The table comprises two columns to illustrate the differences between the TT and TVT strategies. It is worth noting that the probability of selecting that extra feature in the grammar increases in cases where encapsulated features are identical-such as when the best-trained and best-validated individuals are the same. Operations, constants, and input features within the grammars enable exploration of the continuous real-values search space in this domain. The grammars for the remaining problems adhere to these same designs and operations and can be found in our supplementary material.

Table 2. Extended Grammars Version 2 for Paige 1 problem. Input space expansion is highlighted in bold.

TT	TVT
<e> ::= add(<e>,<e>)|sub(<e>,<e>)	<e> ::= add(<e>,<e>)|sub(<e>,<e>)
|mul(<e>,<e>)|AQ(<e>,<e>)	|mul(<e>,<e>)|AQ(<e>,<e>)
|cos(<e>)|sin(<e>)	|cos(<e>)|sin(<e>)
|<x>|<s>0.<c><c>	|<x>|<s>0.<c><c>
|<x'>	|<x'>
<x> ::= x[0]|x[1]	<x> ::= x[0]|x[1]
<x'> ::= **x[2]**	<x'> ::= **x[2]**|**x[3]**
<c> ::= 0|1|2|3|4|5|6|7|8|9	<c> ::= 0|1|2|3|4|5|6|7|8|9
<s> ::= −|+	<s> ::= −|+

The summary of parameters is in Table 3. The GE implementation [11] utilises variable one-point crossover and int flip per codon mutation. The training, validation, and testing sets are created randomly, and each run uses a different random seed to ensure variation.

5 Results and Discussion

We utilise the results from 30 runs for all experiments to compute the mean and conduct statistical tests, such as Friedman or ANOVA ($\alpha = 0.05$), to assess whether to reject the null hypothesis H_0, along with post-hoc tests such as Nemenyi or Tukey. If the samples exhibit different variances, Friedman-Nemenyi tests are performed; otherwise, ANOVA-Tukey tests are conducted, as suggested by Herbold et al. [6]. A higher rank indicates better performance. When methods

Table 3. Summary of parameters.

Parameter type	Parameter value
Number of runs	30
Population size	200
Number of generations	200
Re-/Initialisation method	Sensible
Mutation probability	0.01
Crossover probability	0.80
Elitism size	1
Max. init. tree depth	7

are connected by a horizontal bar, they fall within the Critical Distance (CD), indicating no significant difference in average ranks.

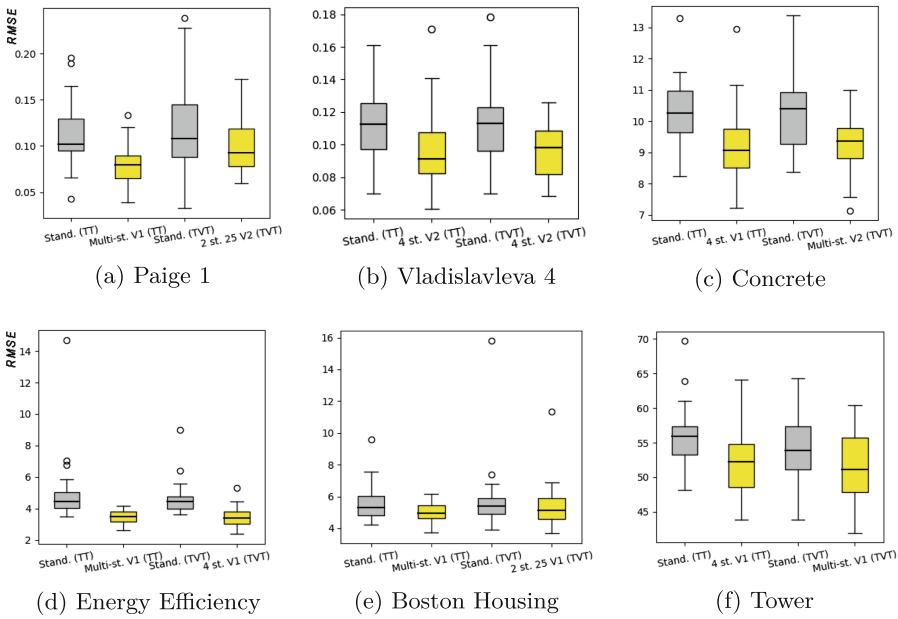
First, to determine the optimal selection process for GE experiments with these problems, we compared the standard baseline using tournament [4], ϵ-Lexicase [10], and ϵ-Lexi^2 [12]. The results demonstrated that the bloat control of ϵ-Lexi^2 led to better results. These findings can be found in the supplementary materials. Although tournament yielded the best result for the Energy Efficiency problem among the three selection processes analysed, it did not statistically outperform ϵ-Lexi^2.

Later, we conducted experiments on the six benchmark regression problems to evaluate the effectiveness of the proposed approaches in reducing the $RMSE$ metric. Table 4 presents the mean fitness on the test set, including TT and TVT strategies and the five FES methods.

We can observe that almost all FES methods outperform the standard baseline. The Multi-stage approach stands out by achieving the best results in all studied problems in at least one of the strategies, with four problems showing superior performance with Extended Grammar Version 1 and two problems with Extended Grammar Version 2. Furthermore, we notice that for the first four problems, the TT strategy (without validation) performs better. In comparison, for the other two problems, the TVT strategy (with validation) generally yields better results. Overfitting occurred twice with the Boston Housing problem: once with the three-stage approach (16.690) and once with the Multi-stage approach (9.108) using the TVT strategy with Extended Grammar Version 2. This suggests that the random validation may not be strong enough to prevent overfitting, and this behaviour arises due to the high convergence of the FES methods. When overfitting occurs in these experiments, the training error continues to be minimised by generations until the last; however, when the best solution in the last generation is evaluated with the test set, the fitness ($RMSE$) jumps considerably to a very high value.

Table 4. The mean fitness on the test set is provided, including TT and TVT strategies specified in the second column. Results for the stages approaches include both grammars (Version 1 and Version 2, see Table 2). The best result for each problem/strategy is highlighted in bold. FES methods consistently outperform the standard baseline. Notably, the Multi-stage approach stands out by achieving the best results in all problems studied in at least one of the strategies.

Problem	Strategy	Baseline	2 stages 25%	2 stages 50%	3 stages	4 stages	Multi-stage
Paige 1	TT	0.111	0.094/0.104	0.092/0.096	0.091/0.091	0.082/0.094	**0.081**/0.087
	TVT	0.119	0.100/0.102	0.091/**0.086**	0.090/0.093	0.095/0.095	0.116/0.114
Vladislavleva 4	TT	0.112	0.111/0.096	0.113/0.105	0.102/0.096	0.103/**0.095**	0.109/0.098
	TVT	0.113	0.114/0.099	0.111/0.104	0.105/0.102	0.106/**0.098**	0.101/**0.098**
Concrete	TT	10.279	9.687/9.874	9.606/9.633	9.419/9.255	**9.202**/9.338	9.228/9.546
	TVT	10.244	9.809/9.507	9.619/9.587	9.282/9.745	9.386/9.648	9.271/**9.233**
Energy Efficiency	TT	4.951	4.343/3.946	4.297/4.052	3.745/3.597	3.489/3.488	**3.462**/3.672
	TVT	4.623	4.089/3.840	4.086/4.016	3.724/3.585	**3.451**/3.483	3.500/3.702
Boston Housing	TT	5.541	5.351/5.323	5.271/5.309	5.236/5.190	5.351/5.168	**5.046**/5.247
	TVT	5.807	**5.425**/5.510	5.713/5.793	5.532/16.690	5.530/5.742	5.795/9.108
Tower	TT	55.907	54.947/53.965	53.652/53.176	52.355/53.119	**52.110**/52.544	52.746/53.863
	TVT	54.048	54.047/54.587	53.548/53.644	51.558/52.786	51.653/52.134	**51.541**/53.060



(a) Paige 1 (b) Vladislavleva 4 (c) Concrete

(d) Energy Efficiency (e) Boston Housing (f) Tower

Fig. 3. Boxplot charts comparing for all problems the standard baseline (grey) with the best FES method (yellow) in terms of the mean of $RMSE$ in 30 runs, including both strategies TT and TVT. V1 and V2 are Extended Grammar Version 1 and Extended Grammar Version 2 respectively. (Color figure online)

We also note that for the two problems where the TVT strategy performs better (Boston Housing and Tower), Extended Grammar Version 1 should be used to avoid overfitting.

After reviewing these results, we selected the best FES approach to generate the Boxplot charts in Fig. 3, comparing it with the standard baseline using the two strategies. Upon examining the results of the Energy Efficiency problem, we observe a more significant improvement compared to the others. This can be attributed to the multi-output problem, resulting in double encapsulation. Furthermore, after the first stage, a correlation between outputs proves beneficial.

5.1 Statistical Tests

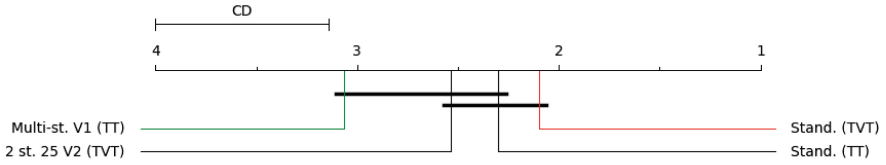
Friedman-Nemenyi statistical tests comparing the standard baseline with the best FES strategy in terms of mean is shown in Fig. 4.

The results of our study are statistically significant, indicating that the Multi-stage approach consistently outperforms the baseline in at least one of the strategies across four out of six problems. This finding underscores the importance of our research in the field of EC, particularly within the context of GE. Notably, the most substantial difference between the FES methods and the baselines is observed in the Energy Efficiency problem, further emphasising the relevance of our findings, particularly in the context of multi-output problems.

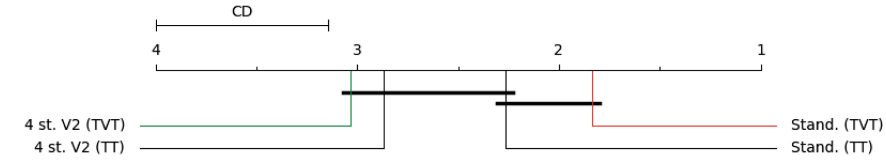
5.2 Complexity of the Individuals

Finally, in Fig. 5, we illustrate the number of nodes utilised by the best-trained individual in the population across generations for the Paige 1, Concrete, and Tower problems using Extended Grammar Version 1.

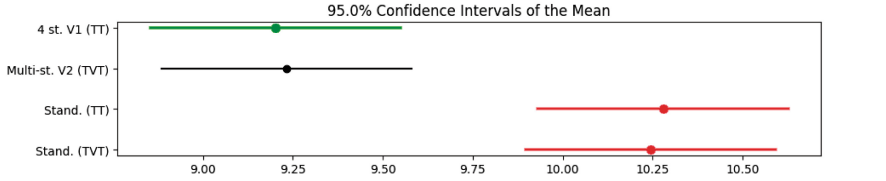
The difference between the stripes reflects the node savings provided by FES. The collapses in the Multi-stage strip correspond to the re-initialisations. Additionally, the TVT strategy tends to achieve greater savings due to the higher probability of selecting encapsulated features. For instance, when comparing these methods with the TT strategy using an Intel i9 with 16 GB of RAM, the improvements in terms of mean elapsed execution time are as follows: Paige 1 - 73.10%, Concrete - 66.78%, and Tower - 72.00%. These results collectively demonstrate that FES is a promising method for EC in the regression domain. All supplementary materials are available in [18].



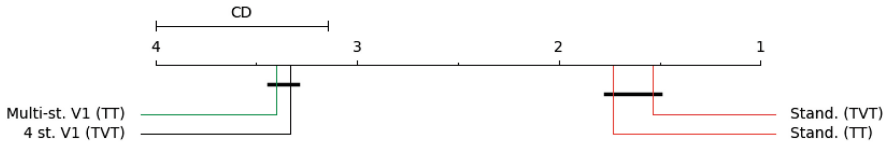
(a) Paige 1



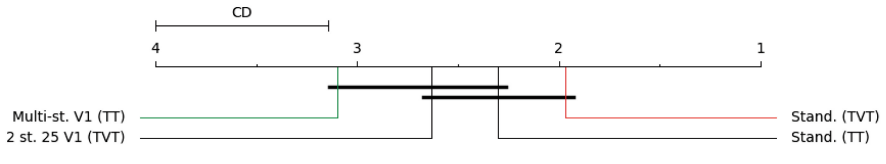
(b) Vladislavleva 4



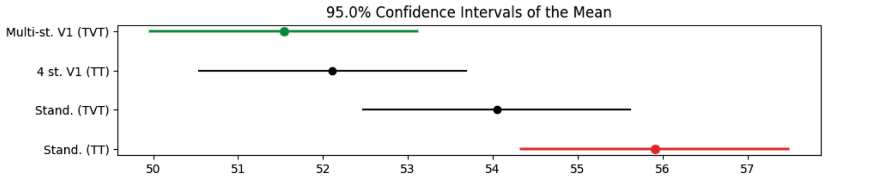
(c) Concrete



(d) Energy Efficiency

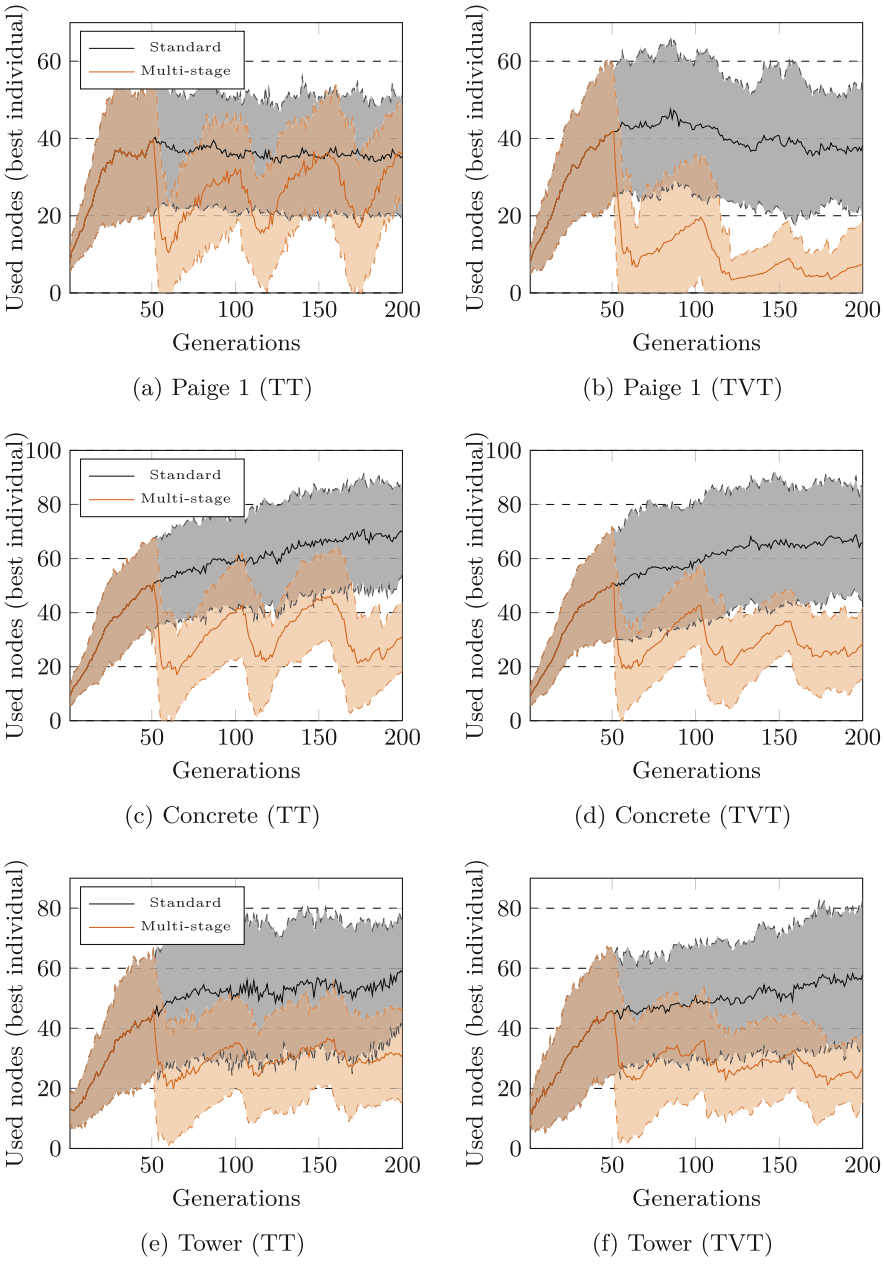


(e) Boston Housing



(f) Tower

Fig. 4. Friedman-Nemenyi or ANOVA-Tukey statistical tests for all problems comparing the standard baseline with the best FES method in terms of mean, including both strategies TT and TVT. V1 and V2 are Extended Grammar Version 1 and Extended Grammar Version 2 respectively. Two treatments being compared are significantly different if their intervals do not overlap. Significant differences from the best FES method (shown in green) for each problem are denoted in red. (Color figure online)



(a) Paige 1 (TT)

(b) Paige 1 (TVT)

(c) Concrete (TT)

(d) Concrete (TVT)

(e) Tower (TT)

(f) Tower (TVT)

Fig. 5. Number of used nodes of the best-trained individual by generations for the problems Paige 1, Concrete, and Tower.

6 Conclusions

We have demonstrated substantial performance improvement and computational cost reduction by dividing the evolutionary process into stages with FES in the regression domain. The experimental results pertain to the evaluation of test sets, and statistical tests were conducted to draw conclusions. The overall best variant is the dynamic Multi-stage approach, as it yielded the best average results across all studied problems, with four problems using Extended Grammar Version 1 and two problems using Extended Grammar Version 2. Additionally, we analysed the complexity of the best-trained individuals in terms of the number of nodes by generations, demonstrating a shorter execution time for GE using FES.

In future work, our first plan is to address multi-target regression problems with more than two outputs, as the encapsulated features might exhibit statistical dependencies among. Furthermore, we aim to study the convergence of training, validation, and test fitness across generations to understand when overfitting occurs and apply a better validation mechanism to avoid it. Lastly, we plan to study FES in other EC algorithms, such as tree-based GP.

Acknowledgments. This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 16/IA/4605.

Disclosure of Interest. The authors have seen and agree with the contents of the manuscript and they declare that they have no conflict of interest.

References

1. Aha, D.: UCI Machine Learning Repository (1987). https://archive.ics.uci.edu/ml/datasets.php
2. Batista, J.E., Silva, S.: Comparative study of classifier performance using automatic feature construction by M3GP. In: 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, pp. 1–8. IEEE (2022). https://doi.org/10.1109/CEC55065.2022.9870343
3. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 22–30. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24775-3_5
4. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In: RAWLINS, G.J. (ed.) Foundations of Genetic Algorithms, Foundations of Genetic Algorithms, San Francisco, vol. 1, pp. 69–93. Elsevier (1991). https://doi.org/10.1016/B978-0-08-050684-5.50008-2. https://www.sciencedirect.com/science/article/pii/B9780080506845500082
5. Gupt, K.K., Kshirsagar, M., Dias, D.M., Sullivan, J.P., Ryan, C.: A novel ml-driven test case selection approach for enhancing the performance of grammatical evolution (2023)
6. Herbold, S.: Autorank: a python package for automated ranking of classifiers. J. Open Source Softw. **5**(48), 2173 (2020). https://doi.org/10.21105/joss.02173

7. Keijzer, M., Ryan, C., Cattolico, M.: Run transferable libraries — learning functional bias in problem domains. In: Deb, K. (ed.) GECCO 2004. LNCS, vol. 3103, pp. 531–542. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24855-2_63
8. Koza, J.R.: Genetic Programming, 1st edn. MIT Press, Cambridge (1992)
9. Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A.: Reuse, parameterized reuse, and hierarchical reuse of substructures in evolving electrical circuits using genetic programming. In: Higuchi, T., Iwata, M., Liu, W. (eds.) ICES 1996. LNCS, vol. 1259, pp. 312–326. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63173-9_56
10. La Cava, W., Spector, L., Danai, K.: Epsilon-lexicase selection for regression. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO 2016, pp. 741–748. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2908812.2908898
11. de Lima, A., Carvalho, S., Dias, D.M., Naredo, E., Sullivan, J.P., Ryan, C.: Grape: grammatical algorithms in python for evolution. Signals **3**(3), 642–663 (2022). https://doi.org/10.3390/signals3030039
12. de Lima, A., Carvalho, S., Dias, D.M., Naredo, E., Sullivan, J.P., Ryan, C.: Lexi^2: lexicase selection with lexicographic parsimony pressure. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 929–937. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3512290.3528803
13. Liu, H., Cai, J., Ong, Y.S.: Remarks on multi-output gaussian process regression. Knowl.-Based Syst. **144**, 102–121 (2018). https://doi.org/10.1016/j.knosys.2017.12.034. https://www.sciencedirect.com/science/article/pii/S0950705117306123
14. Mastelini, S.M., Santana, E.J., Cerri, R., Barbon, S.: Dstars: a multi-target deep structure for tracking asynchronous regressor stacking. Appl. Soft Comput. **91**, 106215 (2020). https://doi.org/10.1016/j.asoc.2020.106215. https://www.sciencedirect.com/science/article/pii/S1568494620301551
15. McDermott, J., et al.: Genetic programming needs better benchmarks. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, GECCO 2012, pp. 791–798. Association for Computing Machinery, New York (2012). https://doi.org/10.1145/2330163.2330273
16. Medernach, D., Fitzgerald, J., Azad, R.M.A., Ryan, C.: Wave: a genetic programming approach to divide and conquer. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion 2015, pp. 1435–1436. Association for Computing Machinery, New York (2015). https://doi.org/10.1145/2739482.2764659
17. Murphy, A., Ryan, C.: Improving module identification and use in grammatical evolution. In: 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, pp. 1–7. IEEE (2020). https://doi.org/10.1109/CEC48606.2020.9185571
18. Reyes, D.: BDS Group repository (2023). https://github.com/bdsul/fes
19. Reyes, D., de Lima, A., Murphy, A., Dias, D.M., Ryan, C.: Feature encapsulation by stages using grammatical evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. GECCO 2024 (2024). https://doi.org/10.1145/3638530.3654097
20. Ryan, C., Azad, R.M.A.: Sensible initialisation in grammatical evolution. In: Barry, A.M. (ed.) GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, Chigaco, pp. 142–145. AAAI (2003)

21. Ryan, C., Collins, J.J., Neill, M.O.: Grammatical evolution: evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) EuroGP 1998. LNCS, vol. 1391, pp. 83–96. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055930
22. Spector, L.: Assessment of problem modality by differential performance of lexicase selection in genetic programming: a preliminary report. In: Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO 2012, pp. 401–408. Association for Computing Machinery, New York (2012). https://doi.org/10.1145/2330784.2330846
23. Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., Vlahavas, I.: Multi-target regression via input space expansion: treating targets as inputs. Mach. Learn. **104**(1), 55–98 (2016). https://doi.org/10.1007/s10994-016-5546-z
24. Vlachos, P.: StatLib-Datasets Archive (1987). https://lib.stat.cmu.edu/datasets/
25. Xia, Y., Chen, K., Yang, Y.: Multi-label classification with weighted classifier selection and stacked ensemble. Inf. Sci. **557**, 421–442 (2021). https://doi.org/10.1016/j.ins.2020.06.017. https://www.sciencedirect.com/science/article/pii/S0020025520306058

Evaluating the Robustness of Deep-Learning Algorithm-Selection Models by Evolving Adversarial Instances

Emma Hart(✉), Quentin Renau, Kevin Sim, and Mohamad Alissa

Edinburgh Napier University, Edinburgh, UK
{e.hart,q.renau,k.sim,m.alissa}@napier.ac.uk

Abstract. Deep neural networks (DNN) are increasingly being used to perform algorithm-selection in combinatorial optimisation domains, particularly as they accommodate input representations which avoid designing and calculating features. Mounting evidence from domains that use images as input shows that deep *convolutional* networks are vulnerable to adversarial samples, in which a small perturbation of an instance can cause the DNN to misclassify. However, it remains unknown as to whether *deep recurrent networks (DRN)* which have recently been shown promise as algorithm-selectors in the bin-packing domain are equally vulnerable. We use an evolutionary algorithm (EA) to find perturbations of instances from two existing benchmarks for online bin packing that cause trained DRNs to misclassify: adversarial samples are successfully generated from up to 56% of the original instances depending on the dataset. Analysis of the new misclassified instances sheds light on the 'fragility' of some training instances, i.e. instances where it is trivial to find a small perturbation that results in a misclassification and the factors that influence this. Finally, the method generates a large number of new instances misclassified with a wide variation in confidence, providing a rich new source of training data to create more robust models.

Keywords: Combinatorial optimisation · algorithm-selection · deep neural networks · adversarial samples

1 Introduction

For most combinatorial optimisation domains, it is well known that for a given set of solvers, each can perform differently on different instances, with no single algorithm dominating the others. This gives rise to the need to perform per-instance algorithm selection, described in detail in a survey by Kerschke *et al.* [11]. Typically a machine-learning algorithm is trained to predict the best solver for an instance. While earlier works generally relied on training models using feature-vectors derived from instances, more recent works have exploited new deep-learning models which circumvent the need to derive features. Deep *convolutional* neural network architectures originally developed for image-classification

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 121–136, 2024.
https://doi.org/10.1007/978-3-031-70068-2_8

have been shown to be capable of learning to predict the best solver from a portfolio in the TSP domain by representing the instance as an image containing the locations of each city [19,23]. In other work, deep *recurrent* neural networks such as LSTM (long short-term memory) [9] and GRU (gated recurrent network) [5] which take an ordered sequence of tokens as input have been trained to act as an algorithm-selectors in the bin-packing [1,2] and vehicle routing [13] domains.

However, mounting evidence from the machine-learning literature shows in particular that deep *convolutional* networks that use images as input are vulnerable to *adversarial attacks*: that is, applying a small perturbation δ to an instance x leads to the perturbed instance $x+\delta$ being misclassified [3,20,25]. As a result, an increasing amount of research is being directed towards generating adversarial attacks, with the goal of understanding how robust a classifier is to variations in input samples. Most of this research considers *black-box* attacks [18]: a scenario in which the attacker only has access to the inputs and outputs of a model, and has no information about the architecture or weights of the model itself. Attacks can be *targeted* in that the adversarial sample is optimised to output a specific (incorrect) class, or *untargeted*, in which case the goal is simply to maximize the loss between the predicted class and the true class. However, to the best of our knowledge, there has been no attempt to understand the extent to which deep *recurrent* networks which have been proposed as algorithm-selectors in combinatorial settings such as bin-packing and VRP (vehicle routing problem) are robust to perturbations in input data. This is crucial to understand: in any real-world setting it is reasonable to assume that a selector may be faced with many instances whose data are very similar to those that the model was trained on (i.e. are *in-distribution*) and therefore will be classified correctly, but a systematic methodology to investigate the extent to which this is true is lacking.

To address this issue we propose a method to determine the extent to which DRNs trained as algorithm-selectors in the bin-packing domain are vulnerable to small changes in input data. We select online bin-packing as a domain for two reasons: (1) the class of problems are NP-hard and are worth studying because they appear as a factor in many other kinds of optimization problem [22]; (2) DRN models have been shown to be highly accurate classifiers in combinatorial settings [2,13]. We propose an evolutionary algorithm (EA) to evolve a mask that when applied to an instance from a dataset used to train a model causes a small modification that results in the perturbed instance being misclassified. The contributions are as follows:

- Defining an EA to evolve adversarial instances of bin-packing instances that are misclassified by a trained DRN while ensuring that the modified instance remains in-distribution
- Providing new evidence that trained DRNs are vulnerable to adversarial attacks using two datasets and associated models.
- Providing new insights into which instances are particularly susceptible to attack and those that are robust, shedding light on which instances lie close to the decision-boundaries of the classifiers

– Generation of a very large set of new instances which can be used for future training: these instances are diverse in terms of the output probability of the correct class, i.e. the confidence with which an instance is classified.

2 Related Work

Deep convolutional neural-networks have rapidly garnered interest within combinatorial optimisation for the purpose of algorithm-selection [11]. Loreggio *et al.* [16] convert a textual description of instances from the SAT and CSP domains[1] to an image and train a convolutional neural network to output solver predictions. Seiler *et al.* [23] also use a convolutional NN trained on images derived from TSP instances to predict the best solver. In continuous optimisation, Prager *et al.* [19] consider both image and point cloud representations of the COCO benchmarks and train a type of CNN called *shuffleNET* [17] to select the best performing algorithm.

Instead of using image-based input, Alissa *et al.* [1,2] directly use the textual description of an instance in the bin-packing domain (i.e. an ordered list of item-sizes) to train two types of deep recurrent neural-networks (DRNs) to predict the best solver. They compare two types of DRN—LSTM [9] and GRU [5]—to feature-based classifiers, finding that the GRU achieves within 5% of the oracle performance on between 80.88 and 97.63% of the instances, depending on the dataset. Diaz [13] propose an attention-based transformer network for selecting a solver for in the VRP domain, finding improved performance compared to a multi-layer perceptron.

However, despite the increasing focus in the use of deep classifiers in algorithm-selection in the combinatorial domain, we are unaware of any work which has investigated the extent to which these classifiers are robust to perturbations in instances via generating adversarial samples. Liu *et al.,.* [15] propose a method to promote data diversity for learning-based branching modules in branch-and-bound (B&B) solvers in which a learning-based solver and instance augmentation policy are adversarially trained, but do not focus per-se on the robustness of the model, rather on instance generation. In contrast, several recent studies in the domain of image-classification have used evolutionary algorithms to generate adversarial samples that represent attacks against models trained on well-known datasets to illustrate the vulnerability of a trained model [3,25] (often described as an 'attack' on the network). For example, *GenAttack* [3] evolves visually imperceptible adversarial examples against state-of-the-art image recognition models trained on three popular datasets (ImageNet, CIFAR-10 and MNIST) with orders of magnitude fewer queries than previous approaches in a black-box setting. Given an input image, it creates a population by applying random modifications to the pixels of the original image. In [20], the authors compare three evolution strategies to generate black-box adversarial attacks on networks trained on the ImageNet dataset, evolving reduced dimensionality samples that are then scaled up to reduce the computational burden. Their results

[1] SAT: Satisfiability, CSP: Constraint Satisfaction Problems [10].

show that CMA-ES [8] is particularly effective in finding adversarial samples with the fewest queries. In contrast to the works just described which modify multiple or even all pixels in an image, in [25], differential evolution (DE) [24] is used to evolve a modification to a *single pixel* showing that current DNNs used for image classification are vulnerable to very low-dimensional attacks. Lin *et al.* [14] propose a technique called Black-box Momentum Iterative Fast Gradient Sign Method (BMI-FGSM) that is also inspired by differential evolution, as well as by iterative gradient-based methods. It leverages DE to approximate gradient direction by searching for the gradient-sign, generating adversarial samples that are hard to detect and which successfully attack DNNs trained on MNIST, CIFAR10, and ImageNet datasets.

Inspired by successful attempts to illustrate the vulnerability of convolutional networks by using an EA to generate attacks, we develop a methodology to evaluate the robustness of deep recurrent network classifiers that use a sequence of tokens as input, using bin-packing as a case-study. Unlike the work in image-classification which tends to treat the task as a continuous optimisation problem, we evolve a mask consisting of discrete values in the range $-n \leq x \leq n$ which indicates how each of the discrete variables describing a combinatorial optimisation instance should be modified.

3 Methods

We assume a target model $\mathcal{M}$ that has been trained to output the probability of selecting a target solver s for a given domain. $\mathcal{M}$ can only be queried as a black-box function, i.e. the inputs and outputs are known but there is no information about the model itself. We search for a perturbation of an *original* instance i_O (from the training data of the model) labelled as won by the kth solver s_k from a portfolio such that the *perturbed* instance i_p is now misclassified by the model $\mathcal{M}$. Specifically, we consider without loss of generality a scenario in which $k = 2$, i.e. there are two available solvers. For each of the *original* instances, there is therefore a winning solver s_w and a losing solver s_l, determined according to a metric that quantifies the quality of the packing produced by the solver. For each instance, applying a perturbation can result in two types of misclassification:

1. The perturbed instance is won by the same solver s_w as the original instance, but the model outputs s_l
2. The perturbed instance is now won by s_l, but the model still outputs s_w

We seek to maximise the probability associated with the output of the *incorrect* class. If p_w is the probability output by the network for the winning solver, and $p_l = (1 - p_w)$ the probability of the losing solver, then we maximize $o = p_l - p_w$. Positive values of o indicate the instance is misclassified while negative values indicate a correct classification with the magnitude of $|o|$ indicating the classifier confidence in the prediction[2].

[2] the approach can be generalised to n classes by taking the difference between $argmax(p_l)$ and p_w.

3.1 Online Bin-Packing

We use the general method described above to evolve adversarial instances in the online bin-packing domain. An instance consists of a sequence of items which must be packed strictly in the order they arrive and no information about the sequence is known in advance of each item arriving. The goal is typically to minimise the number of bins. Simple packing heuristics that determine which bin each item should be placed in are surprisingly effective, particularly best-fit (BF) and first-fit (FF) [7]. BF places each item into the feasible bin that minimises the residual space, while FF places each item into the first feasible bin that will accommodate it. The quality of the resulting packing O_{Falk} is defined by the commonly used Falkenaeur metric [6] which returns a value between 0 and 1 where 1 is optimal, and rewards packings that minimise empty space. For a given portfolio of solvers, the *winning* solver is defined as the solver that maximises O_{Falk}. Note that two instances that have identical *sets* of item-sizes but differ in the order in which items arrive can result in different packings, and elicit different performances from each heuristic. Therefore, the ordering of items is an important characteristic of an instance, in addition to the distribution of item-sizes. In order to minimise the size of the perturbation applied to an instance, we only evolve perturbations that result in a small modification to the size of each item defining an instance. Evolving perturbations that changed the item ordering would result in very different instances and therefore defeat the objective of trying to understand whether small changes to an instance can result in a misclassification.

3.2 Data and Models

We use the data and models previously described in [2]. Two datasets denoted (DS2, DS4) are used: each dataset consists of 2,000 instances, of which 50% are solved best by the best-fit (BF) heuristic and the remaining 50% by the first-fit (FF) heuristic. Item sizes are generated from a normal distribution in the range (20,100) for each dataset; DS2 has 120 items and DS4 has 250. Bins have a maximum capacity of 150. As we previously showed that a Gated Recurrent Network (GRU) outperforms an LSTM on these datasets [2], we restrict the current study to evaluating the vulnerability of GRU models only. We use the GRU architecture and parameterisation as described in [2] except for the final layer which is replaced with a softmax function [4] in order to predict probabilities rather than classes. Models are trained using DS2 and DS4 respectively, according to a 10-fold cross-validation procedure. For DS2, the model has a mean accuracy of 94.44% ($+/-$ 2.92%) on a validation set (over 10 folds) and 93.5% on the test set of 400 instances. For DS4, it achieves 97.00% ($+/-$ 1.11%) mean accuracy on a validation set and 95.5% on a balanced test set of 400 instances.

In a preliminary step, we apply the trained models to the full dataset of 2,000 instances, then remove the small minority of instances that are misclassified, given that we are interested in generating perturbed variations of instances which result in an instance originally classified correctly now being misclassified. This

results in a small reduction in the size of the datasets used from here on in, by 16 instances for DS2 and 46 instances for DS4.

3.3 Algorithm Details

Individual Representation: We evolve a *mask* that is used to perturb an individual instance. The mask contains i integer values, where i is equal to the number of items in the instance. Each integer in the mask can take one of three values $[-1, 0, 1]$: the item at position j in the original instance is modified by adding the integer from the mask at position j, hence decreasing the item size by 1, doing nothing, or increasing the item size by 1. The modification is restricted to this range in order to minimise the extent of the total change that can be made though clearly could be adapted to allow a larger magnitude of perturbation. The maximum *total change* to the item sizes for an instance is thus equal to the number of items. Note that the modification is not cumulative, i.e. a mask is applied exactly once to the *original instance*.

Algorithm: We use a generational EA to evolve masks. An initial population of size P is initialised as follows: first, every element in each mask is set to 0. Then, with probability p_{init}, each element is uniformly randomly changed to $[-1, 0, 1]$. Following the evaluation of the initial population, a generational loop first selects $n = P$ parents; crossover and mutation are applied with probability p_c, p_m respectively to produce an offspring population, after which individuals are evaluated. The offspring population entirely replaces the parent population. Tournament selection is used with tournament size 2 and one-point crossover. Mutation works as follows: each element of the mask is mutated with probability 1/(number of items). A customised mutation operator selects a new value $[-1, 0, 1]$ with equal probability. If a modification results in an item size which falls outside of the fixed range of item sizes defining a dataset, then the value is clipped to the respective minimum/maximum allowed value. The population size is set to 50, and the algorithm runs for 500 generations. Tuning was deliberately kept to a minimum in order to demonstrate that a 'default' algorithm is capable of finding adversarial samples.

Evaluation Function: As noted above, we consider a setting with two solvers. After a mask is applied to an instance i, then a packing is produced from each solver, resulting in objective values o_{BF}, o_{FF} according to the Falkenauer metric. Assume we label the *winning* solver s_w and the *losing* solver s_l, and that the probabilities output by the classifier for s_w, s_l respectively are p_w and p_l (such that $p_w + p_l = 1$), then the fitness function that drives the search for misclassified instances is defined as:

$$f = p_l - p_w \tag{1}$$

The function aims to maximise the confidence of an incorrect classification. A positive fitness value corresponds to a misclassified instance and a negative value to a correctly classified instance. A fitness of exactly 0 implies both probabilities are equal to 0.5 however in practice this situation is never encountered.

Algorithm 1. Pseudo-code for assigning a fitness score to a perturbation

Require: Mask, s_1, s_2 ▷ s1,s2 = solver 1, solver 2
  $i_P \leftarrow perturb\_instance(i, mask)$
  $o(s_1) \leftarrow solve\_instance(i_P, s_1)$ ▷ Falkenauer metric of s_1 on perturbed instance
  $o(s_2) \leftarrow solve\_instance(i_P, s_2)$ ▷ Falkenauer metric of s_2 on perturbed instance
  $winner \leftarrow argmax(o(s_1), o(s_2))$ ▷ winner is id of solver with max fitness
  $p(s_1) \leftarrow query\_classifier(i_P)$
  $p(s_2) \leftarrow (1 - p(s_1))$
  if winner is s_1 **then**
    $fitness \leftarrow p(s_2) - p(s_1)$
  else if winner is s_1 **then**
    $fitness \leftarrow p(s_1) - p(s_2)$
  end if
  $return(fitness)$

3.4 Experimental Protocol

We conduct an initial experiment in which we randomly sample 500 masks with p_{init} set to the relatively high probability of 0.3 following the initialisation procedure outlined in Sect. 3.3 and apply the masks to each instance. The purpose of this is to gain some insight into how easy it is to generate an adversarial sample by simply randomly sampling masks. An instance is labelled as *fragile* if at least one of the randomly sampled masks results in an instance that is misclassified. For DS2, $875 = 43.4\%$ of instances are not labelled fragile, while for DS4, only 51 instances $= 2.55\%$ are not fragile. This result immediately indicates the model trained on the DS4 instances is much less robust than the DS2 model: randomly sampling masks that on average have 30% of the elements set to $+1$ or -1 results in a misclassified instance. In the remaining experiments, all fragile instances are removed from the datasets and we only attempt to evolve masks to perturb the remaining non-fragile instances. All experiments are repeated 10 times per instance.

4 Results

We first present data from experiments that evaluate the effectiveness of the approach in terms of number of instances that produced misclassifications and the computational effort required, followed by a deeper analysis of the results.

4.1 Effectiveness of the EA

We consider p_{init} values of $\{0.05, 0.3\}$ representing populations that are initialised with very few item perturbations (at $p_{init} = 0.05$) and one with approximately $1/3$ of the items being perturbed on average. We measure the *success-rate*, defined as the percentage of the original instances for which the EA evolved at

least one mask that resulted in a misclassification across the 10 runs. To measure the effort required to find an adversarial sample, we define *queries*[3] as the median of the minimum number of evaluations across each of 10 runs needed to find an adversarial sample. Finally, for each instance, we record the type of misclassification as a % of the successful instances: T1 - the true labels of all new misclassified instances are the same as the original instance; T2 - the true labels of all new misclassified instances are different from the original instance; T3 - the new misclassified instances are from both T1 and T2.

Table 1 shows the data just described. For DS2, the success rate decreases as p_{init} decreases, as would be expected: for the same number of evaluations, the evolved masks are likely to contain fewer modifications. Adversarial samples are found for approximately 33% of instances. Although the success-rate is higher for DS4 ($\approx 56\%$), recall that the number of non-fragile instances to which the EA is applied is very small (51 vs 875 for DS2). Interestingly, the value of p_{init} has no effect on DS4—in fact, the set of instances for which a successful perturbation is found is identical for both values of p_{init}. This suggests there is a subset of instances that are relatively easily perturbed (i.e. at $p_{init} = 0.05$) but further perturbations have no effect. We shed more insight into this in Sect. 5.

4.2 Quality of Evolved Adversarial Instances

Table 1 shows the median, the first, and third quartiles of the maximum fitness obtained per instance across the 10 repeated runs. Recall that positive values indicate that a misclassified instance was found. For DS2(0.3), a large interquartile difference of positive values ranging from just greater than 0 (weak confidence in the misclassification) to 1 (very strong confidence in the misclassification) is observed. For DS2(0.05), some masks are evolved that reduce the probability of a correct classification (compared to the original instance) but are not misclassified ($f < 0$). In contrast for DS4, evolved masks generally create instances that are strongly misclassified.

Table 1. Effectiveness of EA: T1,T2,T3 are the types of misclassification as a % of misclassified instances. Median, first quartile (Q1), and third quartile (Q3) of the maximum fitness obtained at the end of each run over all instances.

	Success Rate (%)	Queries	T1	T2	T3	Median fitness	Q1 fitness	Q3 fitness
DS2(0.3)	33.49	1500	27	61	12	0.9912	0.0002	0.9999
DS2(0.05)	28.69	5600	39	54	7	0.9542	-0.993	0.9999
DS4(0.3)	56.87	50	34	45	21	0.9999	0.9999	1.0
DS4(0.05)	56.87	2300	41	41	18	0.9999	0.9999	0.9999

[3] Following the terminology employed in the literature on evolving adversarial samples for image-classification.

5 Analysis

5.1 Path Towards Misclassification

Recall that each instance is perturbed by the application of a single mask and that fitness is defined as $|p_l - p_w|$, with positive values indicating a misclassification. Positive values close to 0 indicate low confidence in the classification, and values close to 1 indicate very high confidence in the classification. Figure 1 shows some examples of types of fitness curves obtained from evolving masks that successfully cause a misclassification on three instances from DS2. It is clear that different types of behaviours are observed. For example, in Fig. 1(a) there is a gradual improvement of fitness over time, terminating in either a new instance misclassified with medium confidence or high confidence respectively. In Fig. 1(b), a mask is found in the first few generations which causes an immediate 'flip' from the original instance being classified correctly with very high confidence to classified incorrectly with very high confidence. In the final example shown (c), successive generations result in an oscillation between high confidence (correct class) and high confidence (incorrect class). This implies that the original instance appears very susceptible to perturbation.

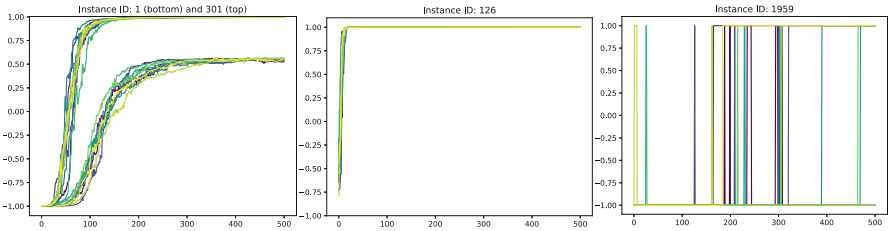


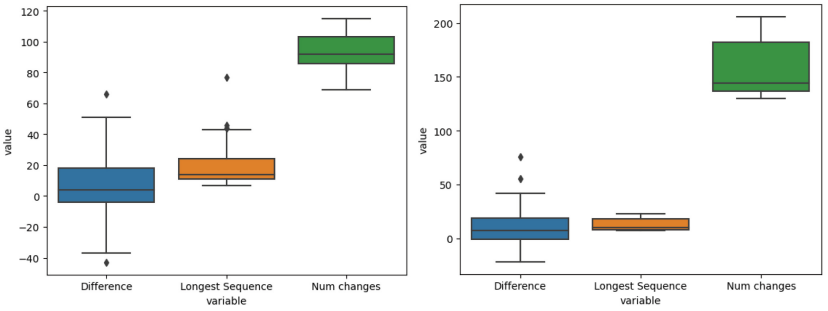
Fig. 1. Behaviours observed during mask evolution: generations (x-axis) vs fitness (y-axis). Each color represents one run. (a) gradual increase in fitness overtime leading to instances classified incorrectly with medium confidence/high confidence; (b) a mask is found in the first few generations which immediately 'flips' the classification to strongly misclassified (c) like (b) but perturbations result in oscillation between high confidence (correct class) and high confidence (incorrect class).

5.2 Insights Into How the Instances Change

The EA is restricted to evolving masks that can only modify the size of each item by $(+/-)1$. Given that item sizes in the original datasets are discrete values drawn from a uniform distribution between 20 and 100 (and clipped to the minimum/maximum values) then even if every item is changed by $+1$ or -1,

the distribution of item-sizes is unlikely to deviate from the original normal distribution[4].

Given that the bin size remains fixed at 150, then the sum of the n item sizes $\Sigma = \sum_{i=1}^{i=n} itemSize_i$ in the instance can influence the number of bins required. In the extreme case for example, if all item sizes are increased by 1, then more bins might be required. For all of the original instances where it was possible to evolve a mask that resulted in a misclassified instance, we calculate the difference D between the sum Σ of the items in an original instance and every new misclassified instance produced for that instance. A box plot of the median value of D over all m misclassified instances produced is shown in Fig. 2 for DS2. The median difference is +4. Figure 2 also plots the median number of changes induced by a mask, i.e. the number of items that will be modified. The median is 92. This indicates that although a large proportion of the items change size by $+/-1$ ($\approx 77\%$ of items), the overall effect of these changes is to more or less cancel each other out in terms of Σ. Therefore it is reasonable to assume that this has little impact on the optimal number of bins required.



(a) DS2 (b) DS4

Fig. 2. Statistics calculated over *all* the new misclassified instances generated from DS2 and DS4, $p_{init} = 0.3$. *Difference*: difference in the sum of item sizes between the original instance and a modified instance; *Longest Sequence*: maximum length of a consecutive perturbation; *Changes*: total number of modifications per instances.

Both heuristics pack items strictly in the order specified by the instance. Therefore if consecutive items are modified, this could potentially alter the packing, thereby causing a misclassification. Figure 2 also shows the median length of the longest consecutive sequence of modifications in a mask that results in a misclassification, where a modification is defined as either an increase or decrease in the item-size. The median length for DS2 is 14.25, and 10 for DS4: the sequences are relatively short compared to the instance length of 120/250 respectively. The

[4] We evaluated this hypothesis empirically by sampling a large number of pairs of (original-instance, modified-instance); a Kolmogorov-Smirnov test showed that the null hypothesis could never be rejected.

median length of sequences with only positive (+1) modifications which might be expected to have more effect on packing is 5 for both datasets.

Table 2. Spearman correlation coefficient between the fitness of misclassified instances and statistics describing the change in instance properties.

Dataset	Statistic	Correlation	p-value
DS2	Longest sequence	−0.898	<< 0.001
	Number of changes	−0.903	<< 0.001
	Difference	−0.451	<< 0.001
DS4	Longest sequence	−0.886	<< 0.001
	Number of changes	− 0.834	<< 0.001
	Difference	−0.623	<< 0.001

To determine if there is a relationship between the statistics depicted in Fig. 2 and the median fitness of the misclassified new instances obtained from each of the original instances, we calculate the Spearman correlation coefficient. The results are shown in Table 2. A very strong negative correlation is obtained both between fitness and the longest sequence, and fitness and number of changes for DS2 and DS4 (<-0.8); in DS4 there is also a strong correlation (<− 0.6 between the difference in the sum of item-sizes D and fitness. This suggests that modifying a contiguous sequence does have an influence on the packing, potentially modifying the winning heuristic and resulting in a misclassification.

(a) Instance 565, median fitness of new instances ≈ 0.007

(b) Instance 694, median fitness of new instances 0.474

(c) Instance 72, median fitness of new instances 0.999

Fig. 3. DS2: Illustrative examples of masks leading to misclassified instances. Yellow indicates a change of +1, teal 0 and purple -1. 5 misclassified instances are shown for each original instance. (Color figure online)

In Fig. 3 we visualise examples of masks leading to misclassifications for three example instances: (a) the median fitness of the new misclassified instances generated from the single original instance is close to 0, i.e. new instances are misclassified with very low confidence; (b) the median fitness is ≈ 0.5, i.e. there is

medium confidence in the (mis)classification; (c) the median fitness ≈ 1, i.e. very high confidence. This figure clearly illustrates different patterns in the evolved masks. For new instances that are misclassified but with very low confidence (p only just greater than 0) there are repeated regions where there are contiguous modifications of +1, as well as repeated but shorter contiguous regions of '-1' modifications. Very few elements are not modified. In contrast, when the probability of the (mis)classification is close to 1.0, we only observe very short sequences of each 'type' of modification, and it is clear there are many more elements that are not modified at all.

Figure 3 show only a few of many thousands of masks generated during runs of the algorithm. The EA tries to maximise the difference in output probabilities $p_l - p_w$. As demonstrated in Fig. 1, in many runs there is a smooth increase in fitness over generations: at each generation, any individual with fitness > 0 is misclassified such that $0 \leq p_l - p_w \leq w$. Therefore, many misclassified instances are discovered during the search process which guides the EA to maximise the fitness function. We count the number of *unique* masks m_U that produced a fitness > 0 for each starting instance i. Figure 4 shows the distribution of m_U over the a instances for which we were able to evolve at least one mask causing a misclassification for each of the two datasets. Statistics are provided in Table 3. The total number of adversarial samples generated is vast, providing a rich source of training data for training new models. Notice that the median number of adversarial samples discovered *per instance* is much higher for DS4 than DS2, indicating that the model is much less robust but that these samples come from very few instances (29). On the other hand, as previously noted, there are more instances in DS2 that can be modified (293) but the number of adversarial samples generated per instance is lower than DS2.

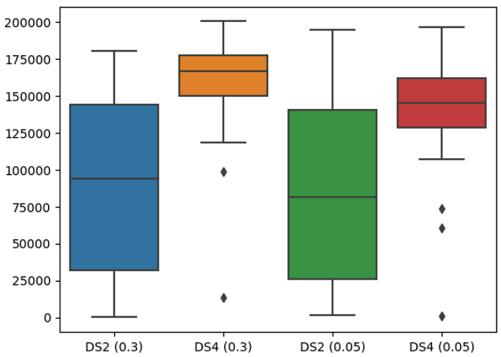


Fig. 4. New misclassified instances generated from each of the original instances that produce at least one misclassification (aggregated over 10 runs).

Table 3. Unique misclassified instances.

	Modifiable Instances	Total Instances	Median per modifiable instance
DS2	293	25,810,846	93,834
DS4	29	4,578,680	166,633
DS2	251	21,194,879	81,607
DS4	29	4,003,832	144,862

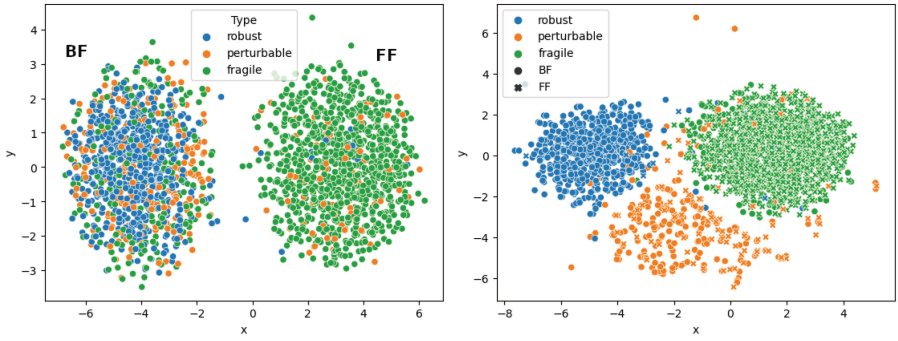
Table 4. DS2: % of instances per solver label and category.

		Robust	Perturbable	Fragile
DS2	BF	29.0	10.5	10.6
	FF	0.5	4.3	45.1
DS4	BF	1.2	0.7	48.3
	FF	0	0.8	49.0

6 Where Are the Fragile Instances?

Finally, we use a dimensionality-reduction technique to visualise the instances in a 2d space to try to uncover any relationships between an instance (described by its data), the winning solver for the instance and the extent to which the original instances are: (a) *fragile*: random search for a mask produces an adversarial sample; (b) *perturbable*: evolution discovers at least one mask that creates an adversarial sample; (c) *robust* : an adversarial sample cannot be evolved.

In Fig. 5a, we use supervised UMAP [12] to learn a projection that accounts for the solver label associated with the instance (BF/FF)—this clearly separates the two classes. We then colour the instances according to the three categories above. It is immediately obvious that most of the instances in the FF cluster are fragile (see Table 4); a small number are perturbable (4.3%) and fewer than 1% are robust. In contrast, most of the robust instances come from the BF class (29%); in this class, the number of fragile and perturbable instances is approximately equal $\approx 10\%$. In Fig. 5b, we again use supervised UMAP but this time train on the three category labels listed above. This clearly separates the three categories, again showing that the fragile instances mainly come from FF. For DS4, the UMAP projection does not separate the three categories and is therefore not shown. Approximately 97% of instances are fragile and are uniformly



(a) Trained using solver labels (b) Trained using 'type' labels

Fig. 5. DS2(0.3), UMAP (supervised) trained with different labels.

distributed across the space, hence we omit this diagram. Statistics showing the percentage of instances per category for both datasets are given in Table 4.

7 Conclusions and Future Work

We investigated the robustness of a deep recurrent network used for algorithm-selection to perturbations in instance data. This was inspired by the wealth of evidence from the image classification domain demonstrating that convolutional networks are particularly vulnerable to adversarial attacks. We proposed a method to evolve a mask that perturbs an instance such that the modified instance is incorrectly classified by a DRN. By restricting the level of perturbation allowed, the method ensures the evolved adversarial samples are similar to the original instances, therefore we expect the trained network should be capable of handling them. However, using two datasets, we showed that instances can be categorised as *fragile, perturbable* or *robust* with respect to the trained models, and that adversarial samples can be evolved efficiently in between 1 and 112 generations, depending on the dataset and the initialisation method. Adversarial samples generated are misclassified with a confidence c, where $0.5 < c \leq 1.0$, i.e. c ranges from very low to very high.

As well as bringing new insight into the robustness of the models, the approach sheds new light on which instances lie close to decision boundaries in the space, i.e. are easily perturbed. We also found a subset of instances in which a perturbation causes the classifier to 'flip' from classifying the instance correctly with very strong confidence to classifying incorrectly with equally strong confidence; every new perturbation can cause the instance to oscillate between these two states. Further work is required to understand what characteristics of the instance data lead to this behaviour, and try to find features that correlate with this. Another promising avenue of work would be to use an multi-objective algorithm to minimise the amount of per perturbation while maximising the probability of misclassification. Finally, a side-effect of the approach is that it generates a very large number of new instances. These instances are associated with a diverse range of classification probabilities and therefore represent a rich source of new training data for training better models in future.

Reproducibility: code and data are available at [21].

Acknowledgements. Emma Hart and Quentin Renau are supported by EPSRC EP/V026534/1

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

References

1. Alissa, M., Sim, K., Hart, E.: Algorithm selection using deep learning without feature extraction. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 198–206 (2019)

2. Alissa, M., Sim, K., Hart, E.: Automated algorithm selection: from feature-based to feature-free approaches. J. Heuristics **29**(1), 1–38 (2023)
3. Alzantot, M., et al.: GenAttack: practical black-box attacks with gradient-free optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1111–1119 (2019)
4. Bridle, J.S.: Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In: Proceedings of the 2nd International Conference on Neural Information Processing Systems, pp. 211–217. NIPS 1889, MIT Press, Cambridge, MA, USA (1989)
5. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
6. Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. J. Heuristics **2**, 5–30 (1996)
7. Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing problems: a survey. In: Ausiello, G., Lucertini, M. (eds.) Analysis and Design of Algorithms in Combinatorial Optimization. ICMS, vol. 266, pp. 147–172. Springer, Vienna (1981). https://doi.org/10.1007/978-3-7091-2748-3_8
8. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001). https://doi.org/10.1162/106365601750190398
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Hoos, H.H., Stützle, T.: Propositional Satisfiability and Constraint Satisfaction. In: Stochastic Local Search: Foundations and Applications. Elsevier (2004)
11. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: survey and perspectives. Evol. Comput. **27**(1), 3–45 (2019)
12. Leland McInnes, John Healy, N.S., Großberger, L.: Uniform manifold approximation and projection. J. Open Source Softw. **3**(29) (2018)
13. Díaz de León-Hicks, E., Conant-Pablos, S.E., Ortiz-Bayliss, J.C., Terashima-Marín, H.: Addressing the algorithm selection problem through an attention-based meta-learner approach. Appl. Sci. **13**(7), 4601 (2023)
14. Lin, J., Xu, L., Liu, Y., Zhang, X.: Black-box adversarial sample generation based on differential evolution. J. Syst. Softw. **170**, 110767 (2020)
15. Liu, H., Kuang, Y., Wang, J., Li, X., Zhang, Y., Wu, F.: Promoting generalization for exact solvers via adversarial instance augmentation. arXiv preprint arXiv:2310.14161 (2023)
16. Loreggia, A., Malitsky, Y., Samulowitz, H., Saraswat, V.: Deep learning for algorithm portfolios. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
17. Ma, N., Zhang, X., Zheng, H.-T., Sun, J.: ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 122–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_8
18. Narodytska, N., Kasiviswanathan, S.P.: Simple black-box adversarial attacks on deep neural networks. In: CVPR Workshops, vol. 2, p. 2 (2017)
19. Prager, R.P., Seiler, M.V., Trautmann, H., Kerschke, P.: Automated algorithm selection in single-objective continuous optimization: a comparative study of deep learning and landscape analysis methods. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) International Conference on Parallel Problem Solving from Nature, vol. 13398, pp. 3–17. Springer (2022). https://doi.org/10.1007/978-3-031-14714-2_1

20. Qiu, H., Custode, L.L., Iacca, G.: Black-box adversarial attacks using evolution strategies. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1827–1833 (2021)
21. Renau, Q., Hart, E.: Evaluating the robustness of deep-learning algorithm-selection models by evolving adversarial instances - code and data (2024). https://doi.org/10.5281/zenodo.10581154
22. Ross, P., Marín-Blázquez, J.G., Schulenburg, S., Hart, E.: Learning a procedure that can solve hard bin-packing problems: a new ga-based approach to hyper-heuristics. 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. 2724, pp. 1295–1306. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45110-2_5
23. Seiler, M., Pohl, J., Bossek, J., Kerschke, P., Trautmann, H.: Deep learning as a competitive feature-free approach for automated algorithm selection on the traveling salesperson problem. In: Bäck, T., et al. (eds.) PPSN 2020. LNCS, vol. 12269, pp. 48–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58112-1_4
24. Storn, R., Price, K.V.: Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997). https://doi.org/10.1023/A:1008202821328
25. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. **23**(5), 828–841 (2019)

Learned Features vs. Classical ELA on Affine BBOB Functions

Moritz Seiler[1(✉)], Urban Škvorc[1], Gjorgjina Cenikj[2], Carola Doerr[3], and Heike Trautmann[1,4]

[1] Machine Learning and Optimisation, University of Paderborn, Paderborn, Germany
{moritz.seiler,urban.skvorc,heike.trautmann}@uni-paderborn.de
[2] Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia
gjorgjina.cenikj@ijs.si
[3] Sorbonne Université, CNRS, LIP6, Paris, France
carola.doerr@lip6.fr
[4] DMB Group, University of Twente, Enschede, Netherlands

Abstract. Automated algorithm selection has proven to be effective to improve optimization performance by using machine learning to select the best-performing algorithm for the particular problem being solved. However, doing so requires the ability to describe the landscape of optimization problems using numerical features, which is a difficult task. In this work, we analyze the synergies and complementarity of recently proposed feature sets TransOpt and Deep ELA, which are based on deep-learning, and compare them to the commonly used classical ELA features. We analyze the correlation between the feature sets as well as how well one set can predict the other. We show that while the feature sets contain some shared information, each also contains important unique information. Further, we compare and benchmark the different feature sets for the task of automated algorithm selection on the recently proposed affine black-box optimization problems. We find that while classical ELA is the best-performing feature set by itself, using selected features from a combination of all three feature sets provides superior performance, and all three sets individually substantially outperform the single best solver.

Keywords: Black-box Optimization · Exploratory Landscape Analysis · Automated Algorithm Selection · Deep Learning

1 Introduction

It is well known that an optimization algorithm's performance depends heavily on the specific problem to be solved. Therefore, choosing the most suitable algorithm for a given problem is crucial to achieve good optimization results. Automating this task, known as *Automated Algorithm Selection* (AAS [28]), has long been of interest to the research community. A prerequisite of the AAS task is the representation of optimization problems in terms of numerical features

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 137–153, 2024.
https://doi.org/10.1007/978-3-031-70068-2_9

which can be used as input to various machine learning (ML) models to predict the algorithms' performances. In recent years, the development of *Exploratory Landscape Analysis* (ELA [22]), a method of transforming samples of an optimization problem into informative *landscape feature*, has shown promising results for these tasks [1,18]. However, ELA has shown certain flaws, such as a large correlation between the individual features, costly computation times, a lack of robustness and expressiveness of some of the features [27], as well as a lack of generalizability to problem sets outside of the widely used *Black Box Optimization Benchmark* (BBOB [14]) [19,26,35].

Recently, novel problem sets were proposed on which these features can be further evaluated, e.g. for the AAS task [11,24,33,36], as well as innovative methods for computing landscape features supposed to solve the inherent drawbacks of classical ELA features [6,29,31]. In this paper, we focus on two recently proposed approaches that utilize deep learning to learn landscape features automatically. These two approaches are *TransOpt* [6] and *Deep Exploratory Landscape Analysis* (Deep ELA [29]) which we compare with each other and also with classical ELA on a recently proposed set of Affine BBOB problems [37], which consists of linear combinations of the 24 widely used single objective optimization problems from the BBOB suite. Thereby, we compare classical ELA features and learned features on novel optimization problems that classical ELA was not specifically designed for. We split this analysis into two parts:

First, we compare the features themselves, by analyzing their correlation and by using ML to determine if one set of features is predicative of the other. The goal is to understand the complementarity of these features, and the amount of new information that is captured by them.

Second, we examine how these features can be used for AAS, and specifically whether combining all of these feature sets and conducting feature selection outperforms only using a single set.

2 Background

In the following, we briefly describe the differences between the three considered feature sets and provide an improvement to Deep-ELA which is the overall fourth considered feature set. In general, we consider two types of feature sets, classical (human-designed) (see Sect. 2.1) and learned feature sets (see Sect. 2.2).

2.1 Classical Exploratory Landscape Analysis

For *Automated Algorithm Selection* (AAS), it is essential to have a method for characterizing the optimization problem's landscape quantitatively, which machine learners can leverage. In current research, the extraction of ELA features is typically used. Pioneered by Mersmann et al. [22], ELA enables the automated extraction of low-level ELA features that directly relate to the high-level characteristics of a problem, like its (multi-)modality or its landscape's

ruggedness. ELA features can be derived from a relatively small set of problem samples. Typically, a sample size of $50d$, or for enhanced stability, $250d$, is used. We use the *flacco* [17] `R`-library to compute classical ELA features as follows:

Basic features provide simple information about the sample set, such as the number of observation and their minimum and maximum boundaries [18].
Dispersion features compare the distribution of the full sample set to the distribution of a subset containing samples with the highest fitness scores [20].
y-Distribution features, such as skewness or kurtosis, indicate descriptive statistics of the fitness values' distribution [22].
Levelset features train discriminant analysis models to predict whether the fitness value of each sample falls above or below a specific quantile of all fitness values. The mean misclassification errors are used as features [18].
Meta-Model features are based on trained linear or quadratic models' performance metrics (e.g. R^2 score) [22].
Information Content features measure certain landscape characteristics, such as smoothness and ruggedness, based on statistic summarization gained from a series of random walks [23].
Nearest Better Clustering feature describes the relation between a set of nearest neighbors (based on their location in the decision space) and a set of nearest 'better' neighbors (based on their objective value) [16].
Principal Component Analysis features are derived from dimensionaltiy reduction by PCA [18].

In total, we obtain 93 features for each problem instance. Further, we excluded all cost-related features as these are meaningless for the characterization of optimization landscapes. In some rare cases, the computation of ELA features results in `NaN` or `Inf` values. Instead of removing all features that contain a single observation with either `NaN` or `Inf` values, we simply mean-imputed these values.

2.2 Learned Features

Of particular interest in the current research are feature-free approaches that utilize deep learning to create alternative problem representations (see e.g. [6,29–31]). In the following, we particularly address the fundamental idea of *TransOpt features* [6] and *Deep Exploratory Landscape Analysis* (Deep ELA [29]).

TransOpt Features (proposed by Cenikj et al. [6].) are obtained by training a transformer-based model on the supervised learning task of predicting the performances of twelve different configurations of *Particle Swarm Optimization* (PSO [15]) algorithms. The inputs to the transformer [34] are a raw set of candidate solutions and their respective objective values. These samples are generated using *Latin Hypercube Sampling* (LHS [21]) and a sample size of $50d$, where d is the decision space's dimensionality. The model follows a transformer encoder architecture, producing representations of the samples which are then fed to a regression head, producing a numerical indicator of the performance of

each of the algorithms. Further, the models are trained on a problem portfolio generated using the random function generator introduced in [33]. A separate model is trained for each problem dimensionality. 2 638 resp. 2 696 functions are used to train the $3d$ resp. $10d$ model. To generate representations of the affine problems, we remove the regression head (last layers of the TransOpt model) and simply make a forward pass of the samples of the affine problems through the trained transformer architecture to obtain their embeddings.

Deep Exploratory Landscape Analysis. (Deep ELA) was proposed by Seiler et al. [29]. The authors designed also a transformer encoder that takes a set of candidate solutions as input and outputs a feature vector that (supposedly) uniquely describes the landscape. The models were trained on 250 000 000 randomly generated optimization problems—containing single- as well as multi-objective problems based on an approach very similar to the one proposed by van Stein [33]. The training routine was designed as a self-supervised learning task as outlined by Chen et al. [8]. The main advantage of this approach is that there is no need for manual or computationally expensive labeling.

In this work, we slightly updated the Deep-ELA approach. Although the predicted feature vector is guaranteed to be invariant to the order of the candidate solutions, the feature vectors may slightly fluctuate depending on permutations of the decision space, i.e. (x_1, x_2, x_3) will not necessarily yield the same output as (x_3, x_2, x_1). To account for this, we perform not only a single forward pass to compute the feature vectors but ten individual forward passes with ten different random permutations of the decision space to compensate for small fluctuations of the feature vectors. Afterwards, we take the arithmetic mean of the ten feature vectors. We will indicate the Deep-ELA variant as proposed by Seiler et al. [29] as *Deep-ELA (v1.0)* and our, updated variant of it as *Deep-ELA (v1.1)*.

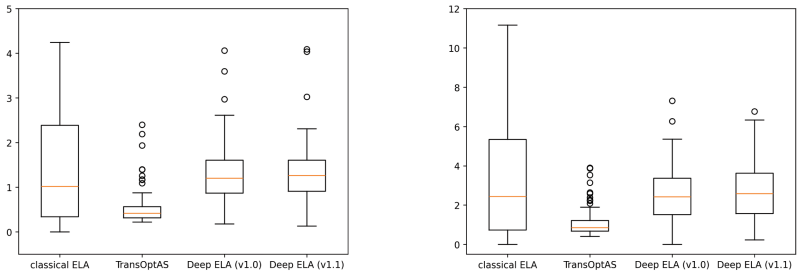
3 Experimental Setup and Methodology

Black-Box Optimization Problems. We reuse most of the data from previous related studies [5] to enable directly comparing results. We generate samples of the affine problem instances using LHS [21] in the range [-5,5] with a sample size of $50d$. To enable a fair feature comparison, we use the same set of samples to calculate the classical ELA, TransOpt, and Deep ELA features.

We create affine BBOB recombinations using the initial five instances from each of the 24 BBOB problem classes [37]. This process involved merging instances from varying classes that share the same instance identifiers. For example, the first instance of the first problem class is combined with the first instances of the other 23 problem classes. We need to highlight here that instances with different instance identifiers from different problem classes are not combined to keep the total number of generated instances manageable. To be more specific, we used the following formula to combine two objectives:

$$F_{f_1,f_2}(X) = \exp\left(\alpha \cdot \log(f_1(X - \mathbf{x}_1^*) - \mathbf{y}_1^*) + (1-\alpha) \cdot \log(f_2(X - \mathbf{x}_2^*) - \mathbf{y}_2^*)\right).$$

Here, f_1, f_2 are the two optimization functions and $\mathbf{y}_1^*, \mathbf{y}_2^*$ their true optimal solutions, and $\mathbf{x}_1^*, \mathbf{x}_2^*$ the locations of the true solutions in the decision space. α is the recombination weight. The recombination is performed with α values of 0.25, 0.50, and 0.75 for all pairs of problem instances. This setup results in 8 280 generated problem instances; $24 \cdot 23$ possible recombinations, with three different α values, and five instances. The 24 different functions and their five instances were taken from the BBOB Suite. Further, we considered the 3-dimensional and the 10-dimensional case for all 8 280 functions, resulting in 16 560 instances in total.



(a) $3d$ (b) $10d$

Fig. 1. Signal to Noise Ratio of the different feature sets. Metrics were created separately for $3d$ (a) and $10d$ (b) data.

Algorithm Performance and Performance Metric. The algorithm portfolio contains *Differential Evolution* (DE [32]), *Genetic Algorithm* (GA [7]), *Particle Swarm Optimization* (PSO [15]), and *Evolutionary Strategy* (ES [2]), executed with their default configuration as specified in `pymoo` [3] (Version 0.6.0). All algorithms use LHS to construct the initial population. The population size is set to $50d$, where d is the decision space's dimensionality. The algorithms are executed on the affine problems in a fixed-budget scenario at budgets of 10, 30, and 50 iterations. For a $10d$ problem, a budget of 50 iterations is equivalent to a total of 5 000 function evaluations. Last, we perform ten executions of each algorithm on all problem instances.

To measure the algorithm performance, we consider a custom performance metric that was originally proposed by Cenikj et al. [5]. Most metrics used to measure algorithm selection performance are either time- or trial-based. While the former utilizes i.e. CPU run time or CPU flops, the latter utilizes i.e. the number of function evaluations. Examples are *Penalized Average Runtime 10* (PAR10) or *Expected Run Time* (ERT). The task of algorithm selection is then to select the algorithm that is expected to solve a given instance the quickest. Yet a major downside of these metrics is their treatment of unsuccessful runs. Not every algorithm is feasible to solve every given instance within a given time budget. Hence, the underlying metric is—de facto—bi-objective. Yet, as

machine learners are most often trained on single-objective loss functions, failed and successful runs have to be factored into a single score in some way or another which is often unintuitive and may cause scores to be not fully commensurable to one another.

In our setup, we prioritize a quality-based measure. Instead of selecting the quickest solver, we select the solver that finds the best solution (in comparison to the other algorithms) within a given budget. Thus, we used a different performance metric which uses the best-found solution: the *Normalized Precision* (NP [5]). It scales the range between the best algorithm's best-found solution and the worst algorithm's best-found solution between zero and one. This makes comparisons between the algorithms straightforward as the VBS is always zero which is also the lower bound. Formally, the NP score can be defined as

$$NP_a = \frac{\hat{y}_a - \min \hat{\mathbf{y}}}{\max \hat{\mathbf{y}} - \min \hat{\mathbf{y}}}$$

where $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_r\}$ is a set of the r algorithms' best-found solutions and $\hat{y}_a$ is the best-found solution of algorithm a. For each of the 8 280 functions, we sampled ten initial candidate solutions with ten different seeds, giving us ten repetitions per instance. The size of this initial set is 150 ($3d$) and 500 ($10d$). These initial sets are evaluated for all 8 280 instances, for $3d$ as well as $10d$, separately. This guarantees equal samples for all feature sets. Afterwards, the four feature sets were created: (1) classical ELA, (2) TransOpt, (3) Deep ELA (v1.0), and (4) Deep ELA (v1.1). These feature sets are the foundation for the following

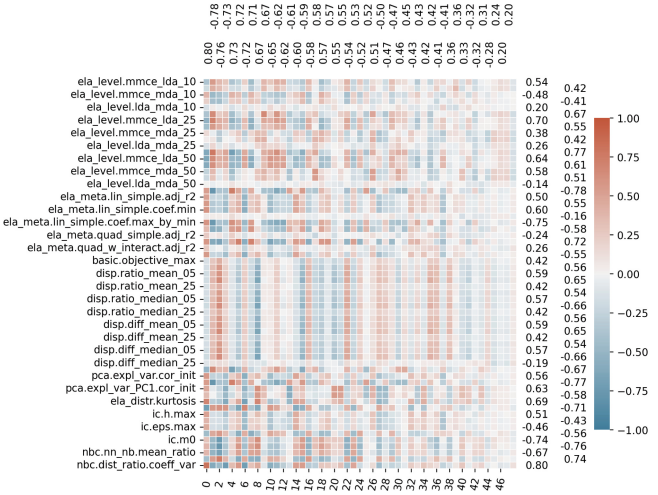


Fig. 2. Spearman Correlation between Deep ELA (v1.1) on the x-axis and classical ELA on the y-axis for the $d = 3$ data. Values on the top (right) show the maximal absolute correlation per column (row).

two studies: (1) we conducted an unsupervised study to compare the four different feature sets to one another, and (2) we performed an automated algorithm selection study using the feature sets separately as well as in conjunction.

Comparison of the Feature Sets. In the first part of our experiments, we perform two studies that assess the similarity of the feature sets used in this paper. In the first study, we examine the Spearman correlation between each pair of feature sets. In the second study, we train a *Support Vector Machine* (SVM [9]) for each feature pair that uses the first feature set to predict the features of the second set. The Spearman correlation is calculated using the library `scipy` [38] using default parameters. The SVM models are trained using the default settings of the `Python` library `scikit-learn` [25], but with a linear kernel, 10 000 maximal iterations, and five-fold cross-validation. Additionally, *Recursive Feature Elimination* [13] implemented by `scikit-learn` is used to determine which features from each feature set contribute the most. As SVMs are affected by different ranges, all features are scaled to a range of $(0, 1)$ before training.

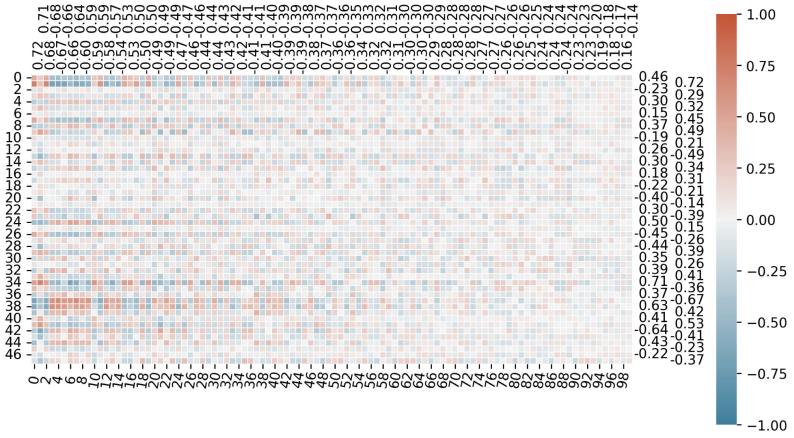


Fig. 3. Spearman Correlation between TransOpt on the x-axis and Deep ELA (v1.1) on the y-axis for the $d = 3$ data. Values on the top (right) show the maximal absolute correlation per column (row).

Algorithm Selection Study. Further, we considered two learners for the algorithm selector: *k Nearest Neighbor* (kNN [10]), and *Random Forest* (RF [4]). After some initial testing, we settled with $k = 15$ for the kNN and 250 trees for the random forest. Other than that, the learned were used with their default configuration as implemented in `scikit-learn`. Input features were min-max normalized for the kNN but left as they are for the random forest.

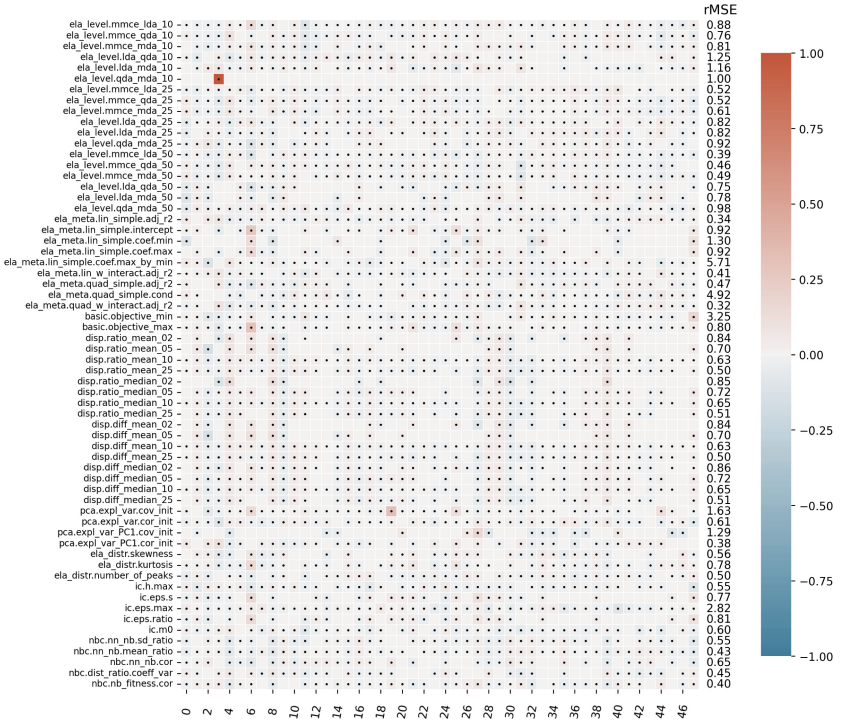


Fig. 4. Detailed results of the feature selection experiments showing feature importance for each predicted feature, with Deep ELA (v1.1) features (x-axis) used to predict classical ELA features (y-axis)

All selectors were trained on a multi-regression task, predicting the performance of each of the four algorithms. Afterward, the algorithm with the lowest predicted regret is selected. To compute the performance scores, we averaged the best-found solution over all ten repetitions per instance to account for stochasticity. We trained the two selectors on all four feature sets with ten-fold cross-validation and *sequential forward feature selection* (SFFS), whereby the cross-validation is the inner loop and the feature selection is the outer one. This way, all learners within an iteration contain the same set of features.

4 Results

Comparison of the Feature Sets. First, we had a look at the Signal-to-Noise (StN) ratio between the different datasets (see Fig. 1):

$$R_{\text{StN}} = \left(\frac{s^2_{\text{between}}}{s^2_{\text{within}}} \right)$$

where s_{between} is the observed standard deviation of a single feature across different functions while s_{within} is the observed standard deviation of a single feature across different instances of a single function. So generally speaking, the StN-ratio captures both the desired stability of a feature within instances of a function and its effectiveness in differentiating between functions, aiming for a balance that highlights features with both low within-function variance and high between-function variance.

Our findings are that classical ELA contains both features with the lowest and the highest StN-ratio. Hence, TransOpt as well as Deep ELA features contain fewer noisy features but also fewer highly descriptive features. Next, we found a small improvement between the default Deep ELA features (as proposed by Seiler et al. [29]) and our improved variant; indicating that the multiple-sampling strategy slightly improves the stability of the Deep-ELA features. On the other hand, it demonstrates that Deep ELA features are already very stable to dimensional augmentations. Last, we found that TransOpt features have the lowest average StN-ratio. This indicates that their stability across instances of the same function is low in comparison to instances across different functions. However, TransOpt features were not explicitly trained to remain stable across different instances of the same function, but are instead trained to have similar representations of problems where algorithms perform similarly.

Next, Figs. 2 and 3 show the results of the correlation analysis. Precisely, Fig. 2 shows the Spearman correlation between the Deep ELA features and the classical ELA features, with the numbers along the rows and columns representing the maximum absolute correlation measured in that specific row and column. To make it easier to examine the maximum correlation, we sort the columns of the table by the maximum column correlation. We only include the results of the experiments that were performed on $3d$ data, as the experiments on $10d$ achieved very similar results.

Figure 2 depicts the correlation between Deep ELA (v1.1) and classical ELA features. We can see that there is some correlation between the two feature sets and that about half of the Deep ELA features have an absolute correlation above 0.5 to at least a single classical ELA feature. However, the other half of the Deep ELA features exhibit a lower correlation. Similar results were observed with the correlation between the TransOpt and the classical ELA features. Further, Fig. 3 shows the correlation between the Deep ELA and the TransOpt features. The results are somewhat similar to Fig. 2 (classical vs. Deep ELA), with some features being heavily correlated. However, the majority of the features show a lower degree of correlation. This demonstrates that while the feature sets examined in this paper are not entirely distinct, all three still contain features that are not heavily correlated with the others, which indicates complementarity.

As correlation can only show us pairwise similarity of features, we perform an additional study to further analyze the similarity of the feature sets. We train SVMs, each using one feature set to predict the features of a different set, with a separate model for each predicted feature. The evaluation metric used for each model is its root Mean Squared Error (rMSE) divided by the predicted feature's

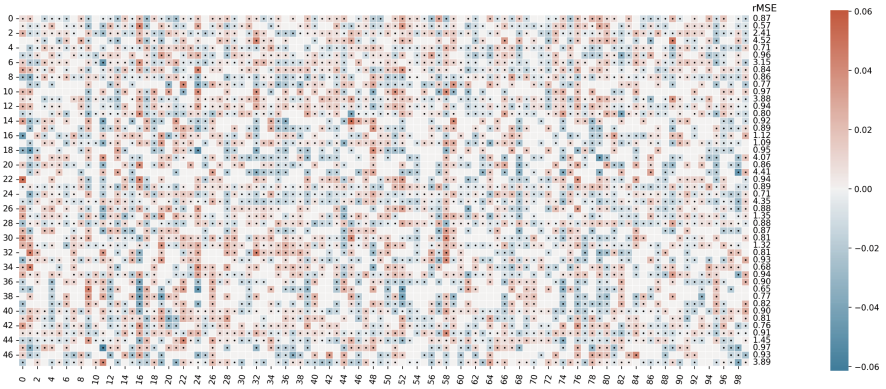


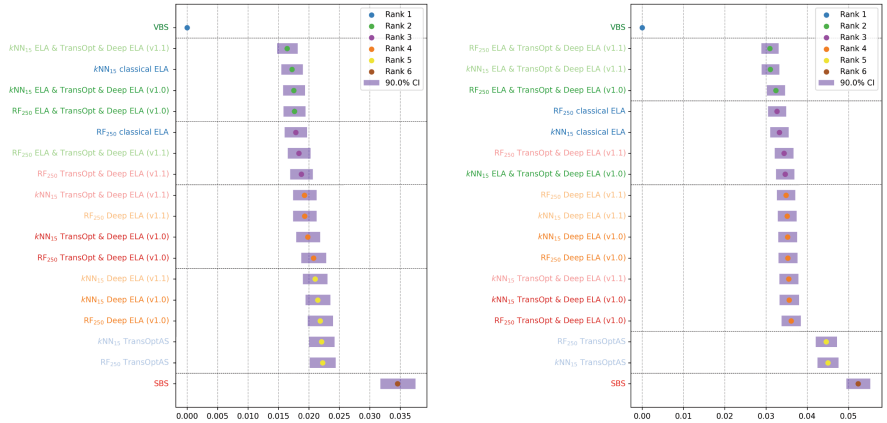
Fig. 5. Detailed results of the feature selection experiments showing feature importance for each predicted feature, with TransOpt features (x-axis) used to predict Deep ELA (v1.1) features (y-axis). Note that to make them more readable, the values are capped to the region of $(-0.06, 0.06)$

standard deviation. Due to this, a value of one indicates performance equivalent to a baseline model that always predicts the mean of the target feature while values below one indicate that the model outperforms the baseline. Hence, we can assume that if the value is below 1, the input feature set is likely to contain similar information. Table 1 shows the aggregated mean results of each feature set, with the rows representing the training set and the columns representing the testing set. We can observe that the best results are achieved when using one Deep ELA version to predict the other, which results in an rMSE of 0.47 or 0.59. This is an expected outcome given the similarity of the two feature sets and shows us what accuracy should be expected when comparing two heavily correlated feature sets. Other models perform worse, achieving normalized rMSE scores of around 0.8 to 0.9, but still below 1, except for TransOpt features. When examining the results in more detail, the worse performance of the TransOpt models can be explained by poor performance on a small number of outlier features. This further shows that all the feature sets contain at least some supplementary features.

Table 1. Mean rMSE normalized by the standard deviation, with rows representing the training feature set and columns representing the testing feature set

	Classical ELA	Deep ELA (v1.0)	Deep ELA (v1.1)	TransOpt
Classical ELA	–	1.0	0.92	0.70
Deep ELA (v1.0)	0.93	–	0.47	0.86
Deep ELA (v1.1)	0.93	0.59	–	0.86
TransOpt	1.3	1.64	1.39	–

Figure 4 and 5 contain more detailed results of the feature prediction experiments. Specifically, they show the feature importance assigned by recursive feature elimination for each predicted feature, as well as the achieved rMSE on each



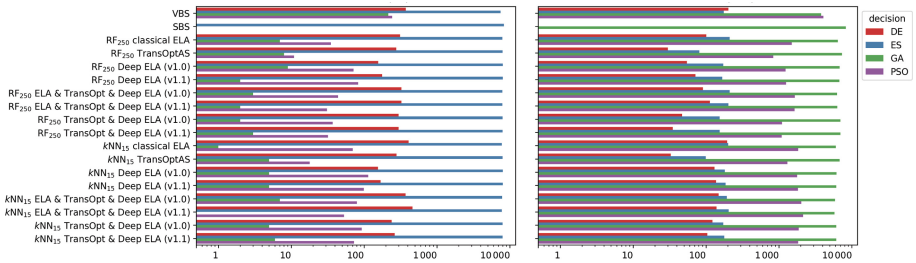
(a) $3d$ (b) $10d$

Fig. 6. Results of the algorithm selection study. Selectors that share the same rank are stochastically tied to one another. We used the robust ranking technique as presented by Fawcett et al. [12] with $\alpha = 0.1$ and 100 000 repetitions. The median performance is shown on the x-axis.

feature. Figure 4 shows these results when the Deep ELA (v1.1) features are used to predict the classical ELA features. We can see that the feature importance is fairly well distributed with a couple of exceptions. Looking at the rMSE for each feature, we can also see that the Deep ELA features are relatively well-suited to predicting most classical ELA features. However, there are a couple of outliers where the model performs extremely poorly.

Figure 5 shows comparable results. The feature importance is evenly distributed, and while the overall rMSE is worse than in the previous figure, there are still noticeable outliers where the model performs worse than the baseline. After examining the data, most (but not all) of the outliers in this figure occur in Deep ELA features that contain very little variance. Since the performance is normalized by the standard deviation, this penalizes such features more harshly, which could explain the relatively poor performance. However, it is also worth noting that, in the algorithm selection study presented in the following chapter, these features are often selected as being highly informative, which substantiates the synergy of these Deep ELA features and the TransOpt features.

Algorithm Selection Study. As explained in the methodology of this paper, we choose to execute our algorithm selection study on two different data sets ($3d$ and $10d$). The results can be found in Fig. 6. In both scenarios, all algorithm selectors provide significantly better performance than the SBS but are still significantly worse than the VBS. Generally speaking, we find that the hybrid selectors, making use of all three feature sets, significantly outperform the other selectors (see Fig. 6 (a,b)). Contrary to the $10d$ case, classical ELA is stochastically tied to the hybrid selectors on the $3d$ dataset. Further, the combination of TransOpt and Deep ELA features significantly outperforms both feature sets on their own in the $3d$ scenario (see Fig. 6 (a)).



(a) 3d (b) 10d

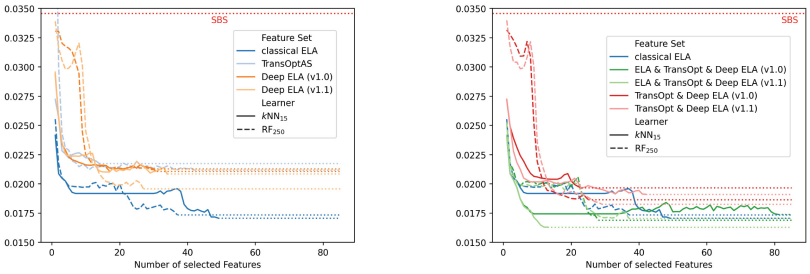
Fig. 7. This figure depicts how often a certain algorithm of four total algorithms was selected by one of the algorithm selectors. Again, these metrics were derived separately for the 3d-case and the 10d-case.*Please note the log-scaling.*

Therefore, our findings are that classical ELA features still provide the most distinct features for algorithm selection, followed by Deep ELA and TransOpt features. Yet, both learned feature sets demonstrate great performance as all selectors outperform the SBS. Further, the best performances were shown by the hybrid models taking all feature sets into account. This indicates that both learned feature sets cover certain characteristics that classical ELA features miss.

Next, we analyzed which algorithm was selected by which selector and how often. The results can be found in Fig. 7. The figure reveals that the VBS, as expected, selects from the algorithm portfolio, the most diversely. The SBS, by definition, only selects a single solver. All algorithm selectors select the SBS the most often—and, in particular, more often than the VBS does. Of the trained algorithm selectors, those trained on classical ELA are most diverse in selecting algorithms. This indicates again that classical ELA features still contain the most relevant information for algorithm selection. Still, both TransOpt and Deep ELA contain sufficient information to distinguish between the different algorithms.

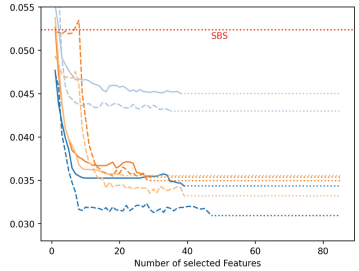
Interestingly, all selectors provide more diverse selections in the 10d scenario in comparison to the 3d. This may be because the algorithms GA and PSO are similar as often selected by the VBS. This is contrary to the 3d case where the ES algorithm is selected by far the most often by the VBS. Hence, it may be easier for the selectors to utilize the algorithm portfolio if all algorithms in the portfolio are about equally as important.

Subsequently, we analyzed the number of selected features that were required to achieve the best performance. In Fig. 8, we show the performance of each algorithm selector on the y-axis and the number of selected features during the forward pass at the x-axis. To better indicate the optimum, we stop at the optimal number of selected features. First of all, when comparing Fig. 8 (a) and (b) as well as (c) and (d) to one another, it becomes apparent that the hybrid models achieve better performance with a lower number of features in comparison to the selectors that are trained solely on separate feature sets. This again demonstrates that every feature set contains highly descriptive features that are very important to the algorithm selection task.

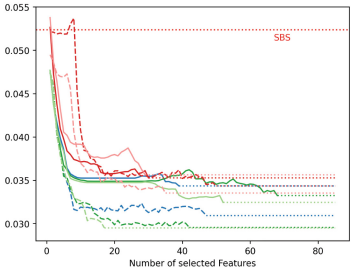


(a) Separate Models 3*d*

(b) Hybrid Models 3*d*



(c) Separate Models 10*d*



(d) Hybrid Models 10*d*

Fig. 8. Performance versus number of selected features of all algorithm selectors during the sequential forward feature selection process. Each depicted line symbolizes a feature selection run and stops at the optimum. *Please note that here the mean performance is depicted.*

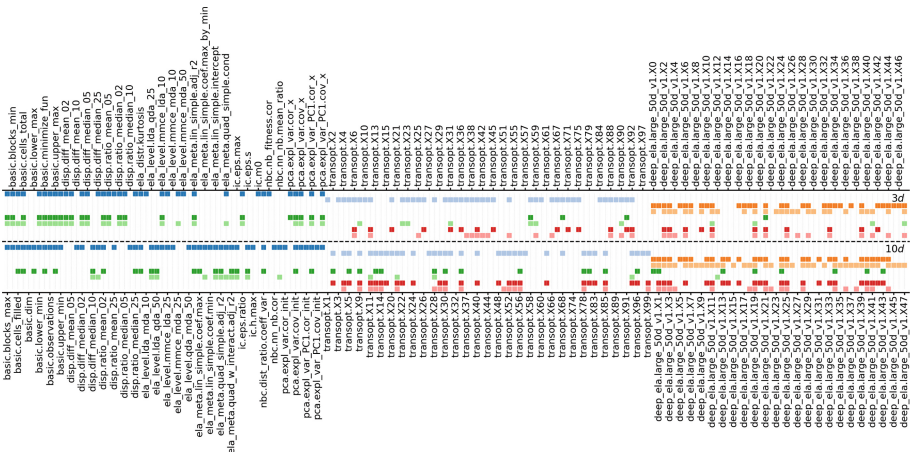


Fig. 9. Selected features of each algorithm selector based on a random forest. Colors represent the same selectors as depicted in Fig. 8.

Thereafter, we took a closer look at which features were selected exactly. The selected features for every random forest-based algorithm selector are depicted in Fig. 9. Many of the selected learned features for the separate selectors are not included in the hybrid models. In fact, the best-performing selector for the $10d$ data, `ELA & TransOpt & Deep ELA (v1.1)`, only considers four TransOpt and a single Deep ELA (v1.1) feature but also twelve classical ELA features. On the other hand, the stochastically tied performing selector, `ELA & TransOpt & Deep ELA (v1.0)`, considers more than twice the number of features. However, this might be just due to stochastic variations in the feature selection process as the latter selector also provides matching performance with a lower number of features (see Fig. 8 (d)). Another reason for the low number of selected Deep ELA features may lie in the findings of our correlation study. Many ELA features are highly correlated to the Deep ELA features and may be removed due to describing similar characteristics. The most important classical ELA features of the hybrid selectors are `basic, dispersion, meta`, and for the $3d$ case also `PCA`-based features. This indicates that neither TransOpt nor Deep ELA covers similar characteristics.

5 Conclusion

We showed that all three feature sets, i.e., classical ELA and the two deep learning-based variants, contain relevant and unique information. This is why the hybrid algorithm selectors, utilizing all three sets, require the fewest number of features while providing the best algorithm selection performance. While the different feature sets contain to some degree shared information, both in terms of correlation and feature predictions, nevertheless, they also contain information that is unique to each feature set. Further, we could demonstrate that those Deep ELA features that were difficult to predict using TransOpt features are important add-ons to the TransOpt features within the AAS Study.

We also observed that classical ELA features are, generally speaking, slightly superior in comparison to learned features. Yet, learned features do provide promising performance. Their main advantage lies in the fact that no manual crafting of new feature sets is required. Instead, a large foundation model learns all the relevant information by itself. This may provide easy scalability as with the adaption and tuning of the training routine, better foundation models can be trained and, thereby, may surpass the superiority of classical ELA features. Learned features may also prove particularly useful for optimization scenarios where human expertise for feature design is scarce. Rather than comparing hand-designed features with learned ones, as we have done in work, we expect to see future studies using learned features to design "human-readable" ones, for domains where hand-designed features are lacking.

Acknowledgments. The third author acknowledges support by the Slovenian Research Agency: research core funding No. P2-0098, young researcher grant No. PR-12393 to GC and project No. J2-4460.

References

1. Belkhir, N., Dréo, J., Savéant, P., Schoenauer, M.: Per instance algorithm configuration of CMA-ES with limited budget. In: Proceedings of the 19th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 681–688. ACM (2017)
2. Beyer, H., Schwefel, H.: Evolution strategies - a comprehensive introduction. Nat. Comput. **1**(1), 3–52 (2002). https://doi.org/10.1023/A:1015059928466
3. Blank, J., Deb, K.: Pymoo: multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)
4. Breiman, L.: Random Forests. Mach. Learn. **45**, 5–32 (2001)
5. Cenikj, G., Petelin, G., Eftimov, T.: A cross-benchmark examination of feature-based algorithm selector generalization in single-objective numerical optimization. Swarm Evol. Comput. **87**, 101534 (2024). https://doi.org/10.1016/j.swevo.2024.101534, https://www.sciencedirect.com/science/article/pii/S2210650224000725
6. Cenikj, G., Petelin, G., Eftimov, T.: Transoptas: transformer-based algorithm selection for single-objective optimization (2024). https://doi.org/10.1145/3638530.3654191, in Press
7. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools Appl. **80**(5), 8091–8126 (2020). https://doi.org/10.1007/s11042-020-10139-6
8. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers (2021)
9. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
10. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
11. Dietrich, K., Mersmann, O.: Increasing the diversity of benchmark function sets through affine recombination. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) Parallel Problem Solving from Nature - PPSN XVII, pp. 590–602. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_41
12. Fawcett, C., Vallati, M., Hoos, H.H., Gerevini, A.E.: Competitions in AI – robustly ranking solvers using statistical resampling (2023)
13. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**, 389–422 (2002)
14. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions. Research Report RR-6829, INRIA (2009). https://hal.inria.fr/inria-00362633
15. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995). https://doi.org/10.1109/ICNN.1995.488968
16. Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: Detecting funnel structures by means of exploratory landscape analysis. In: Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 265–272 (2015)
17. Kerschke, P., Trautmann, H.: The R-package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), pp. 5262–5269. IEEE (2016)

18. Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco. In: Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., Vichi, M. (eds.) Applications in Statistical Computing. SCDAKO, pp. 93–123. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25147-5_7
19. Kostovska, A., et al.: Per-run algorithm selection with warm-starting using trajectory-based features. 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, pp. 46–60. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_4
20. Lunacek, M., Whitley, D.: The dispersion metric and the CMA evolution strategy. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 477–484 (2006)
21. Menčík, J.: Latin hypercube sampling. In: Mencik, J. (ed.) Concise Reliability for Engineers, chap. 16. IntechOpen, Rijeka (2016). https://doi.org/10.5772/62370
22. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 829–836 (2011)
23. Muñoz, M.A., Kirley, M., Halgamuge, S.K.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE Trans. Evol. Comput. **19**(1), 74–87 (2014)
24. Muñoz, M.A., Smith-Miles, K.: Generating new space-filling test instances for continuous black-box optimization. Evol. Comput. (ECJ) **28**(3), 379–404 (2020)
25. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
26. Petelin, G., Cenikj, G.: How far out of distribution can we go with ELA features and still be able to rank algorithms? In: 2023 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 341–346 (2023)
27. Renau, Q., Dréo, J., Doerr, C., Doerr, B.: Expressiveness and robustness of landscape features. In: Proceedings of the 21st Annual Conference on Genetic and Evolutionary Computation (GECCO) Companion, pp. 2048–2051. ACM (2019)
28. Rice, J.R.: The algorithm selection problem. In: Advances in Computers, vol. 15 (1976)
29. Seiler, M.V., Kerschke, P., Trautmann, H.: Deep-ELA: deep exploratory landscape analysis with self-supervised pretrained transformers for single- and multi-objective continuous optimization problems. Evol. Comput. J. arXiv preprint arXiv:2401.01192 (2024). https://arxiv.org/abs/2401.01192
30. Seiler, M.V., Prager, R.P., Kerschke, P., Trautmann, H.: A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes. In: Proceedings of the 24th Annual Conference on Genetic and Evolutionary Computation (GECCO) (2022)
31. van Stein, B., Long, F.X., Frenzel, M., Krause, P., Gitterle, M., Bäck, T.: DoE2Vec: deep-learning based features for exploratory landscape analysis. In: Silva, S., Paquete, L. (eds.) Proceedings of the 25th Annual Conference on Genetic and Evolutionary Computation (GECCO) Companion, pp. 515–518. ACM (2023)
32. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997). https://doi.org/10.1023/A:1008202821328
33. Tian, Y., Peng, S., Zhang, X., Rodemann, T., Tan, K.C., Jin, Y.: A recommender system for metaheuristic algorithms for continuous optimization based on deep

recurrent neural networks. IEEE Trans. Artif. Intell. **1**(1), 5–18 (2020). https://doi.org/10.1109/TAI.2020.3022339
34. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
35. Vermetten, D., Ye, F., Bäck, T., Doerr, C.: MA-BBOB: A problem generator for black-box optimization using affine combinations and shifts. CoRR **abs/2312.11083** (2023)
36. Vermetten, D., Ye, F., Bäck, T., Doerr, C.: MA-BBOB: many-affine combinations of BBOB functions for evaluating AutoMl approaches in noiseless numerical black-box optimization contexts. arXiv preprint arXiv:2306.10627 (2023)
37. Vermetten, D., Ye, F., Doerr, C.: Using affine combinations of BBOB problems for performance assessment. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 873–881. GECCO 2023, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3583131.3590412
38. Virtanen, P., et al.: SciPy 1.0 Contributors: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2

Hybridizing Target- and SHAP-Encoded Features for Algorithm Selection in Mixed-Variable Black-Box Optimization

Konstantin Dietrich[1,2(✉)], Raphael Patrick Prager[3], Carola Doerr[4], and Heike Trautmann[5,6]

[1] Big Data Analytics in Transportation, TU Dresden, Dresden, Germany
konstantin.dietrich@tu-dresden.de
[2] ScaDS.AI, Dresden, Germany
[3] Data Science: Statistics and Optimization, University of Münster, Münster, Germany
raphael.prager@wi.uni-muenster.de
[4] Sorbonne Université, CNRS, LIP6, Paris, France
carola.doerr@lip6.fr
[5] Machine Learning and Optimisation, Paderborn University, Paderborn, Germany
heike.trautmann@uni-paderborn.de
[6] Data Management and Biometrics Group, University of Twente, Enschede, Netherlands

Abstract. Exploratory landscape analysis (ELA) is a well-established tool to characterize optimization problems via numerical features. ELA is used for problem comprehension, algorithm design, and applications such as automated algorithm selection and configuration. Until recently, however, ELA was limited to search spaces with either continuous or discrete variables, neglecting problems with mixed variable types. This gap was addressed in a recent study that uses an approach based on target-encoding to compute exploratory landscape features for mixed-variable problems.

In this work, we investigate an alternative encoding scheme based on SHAP values. While these features do not lead to better results in the algorithm selection setting considered in previous work, the two different encoding mechanisms exhibit complementary performance. Combining both feature sets into a hybrid approach outperforms each encoding mechanism individually. Finally, we experiment with two different ways of meta-selecting between the two feature sets. Both approaches are capable of taking advantage of the performance complementarity of the models trained on target-encoded and SHAP-encoded feature sets, respectively.

Keywords: Mixed-Variable Optimisation · SHAP · Automated Algorithm Selection

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 154–169, 2024.
https://doi.org/10.1007/978-3-031-70068-2_10

1 Introduction

Exploratory landscape analysis (ELA), also referred to as fitness landscape analysis, has shown to be a powerful tool to characterize the landscape of continuous black-box optimization problems [17]. This type of problems lacks a closed-form representation and does not provide any gradient information to exploit.

In a plethora of studies, ELA's usefulness has been demonstrated; either to better understand the behavior of an algorithm during its optimization process [11,15], to determine the degree of multi-modality, global structure or other structural properties of an optimization landscape [17,31], or to use ELA as a means in automated algorithm selection [7,9,21,33] and configuration [3,5,25]. Despite this popularity, ELA has neither been disseminated to other research domains, nor is it really used by practitioners. We hypothesize that one major caveat is the limitation to the purely continuous space. In fact, many black-box optimization problems exhibit a mixed-search space, where the search space can be an arbitrary mixture of continuous, integer, and categorical decision variables. This type of problems is often referred to as *mixed-variable problem* (MVP) [14,23]. While there have been successful attempts to extend ELA to the binary or mixed-integer space [26,27], these works still do not consider categorical decision variables.

This has been addressed in a recent study by [30]. The authors propose a methodology which allows the computation of ELA features for black-box problems with mixed search spaces. This methodology is evaluated in an automated algorithm selection setting where several algorithms are benchmarked on a set of *hyperparameter optimization* (HPO) problems as important representatives of the MVP domain. Thereafter, ELA features are used to automatically select an appropriate algorithm out of the portfolio for any given problem instance. Thereby, they demonstrate that their developed methodology is able to discriminate between problem instances. The results show that this approach is superior compared to relying on any single algorithm out of the portfolio. Their proposed methodology contains a variety of different steps of which one pertains to the transformation of categorical variables into continuous ones. The considered transformations are *one-hot encoding* (as a baseline) and *target-encoding* [18].

The contribution of this study is that we extend the work of [30] by introducing a different encoding method based on SHAP (SHapley Additive exPlanations) values [16]. These values have the advantage that they can represent the features additive contribution to the prediction. In doing so the categorical features are encoded on a 'per observation'-basis while taking into account the interactions to all other contributing features. In this scenario an observation is a single row of data containing various features (input variables) and their corresponding target values (output variable). We use the same experimental settings as [30] to ensure a fair comparison and highlight the merits as well as the caveats of our transformation variant. Our devised method does not outperform the existing results of [30] across all considered problem instances, yet it performs well in certain areas. A combination or what we call hybridization of both encoding methods produces a superior performance than any encoding

method achieves on its own. Leveraging two sophisticated approaches to combine both encoding methods, we manage to improve the automatic algorithm selection substantially.

2 Mixed-Variable Black-Box Optimization

Many optimization problems and HPO problems in particular have a mixed search space. Meaning, the search space of these problems is constituted by a mixture of continuous ($\mathbf{X}_{cont}$), integer ($\mathbf{X}_{int}$), and categorical decision variables ($\mathbf{X}_{cat}$). The latter type poses two challenges for the optimization community especially. First, categorical decision variables do not possess any inherent order relation. Furthermore, $\mathbf{X}_{int}$ and $\mathbf{X}_{cat}$ can impose hierarchical/conditional structure onto other continuous decision variables. In other words, a specific instance of a given categorical decision variable can govern the domain of $\mathbf{X}_{cont}$. In [30], the authors purposefully ignore these constraints in their preprocessing scheme for ELA feature computation. They make the case that the infeasible regions of the search space (reached by ignoring these constraints) are n-dimensional hyperplanes with a constant objective value in the direction of the decision variables with the violated constraint. This information is exploited in their feature generation process. It is important to note that the constraints are only relaxed in the ELA feature generation process and are still in place when algorithms are applied to solve the problem in question.

Hence, we adopt the simplified formal representation of an MVP of [30]:

$$\begin{aligned} \min \quad & f(\mathbf{X}_{cont}, \mathbf{X}_{int}, \mathbf{X}_{cat}) \\ \text{w.r.t.} \quad & \mathbf{X}_{cont} \in \mathbb{R}^{n_{cont}} \\ & \mathbf{X}_{int} \in \mathbb{Z}^{n_{int}} \\ & \mathbf{X}_{cat} \in \mathbb{Z}^{n_{cat}} \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{X}_{cont}, \mathbf{X}_{int}, \mathbf{X}_{cat}) = 0 \\ & \mathbf{h}(\mathbf{X}_{cont}, \mathbf{X}_{int}, \mathbf{X}_{cat}) \leq 0, \end{aligned} \tag{1}$$

where f denotes the objective function with its respective equality constraints $\mathbf{g}$ and inequality constraints $\mathbf{h}$. With that, we define the decision space of our MVP as $\mathbf{X} = (\mathbf{X}_{cont}, \mathbf{X}_{int}, \mathbf{X}_{cat})$ and the objective space is denominated as $\mathbf{Y}$.

3 Problem Representation

3.1 Exploratory Landscape Analysis

Black-box problems in general do not offer any insight into their landscape and what structures they are composed of. To provide the means to distinguish between and characterize different continuous single-objective black-box

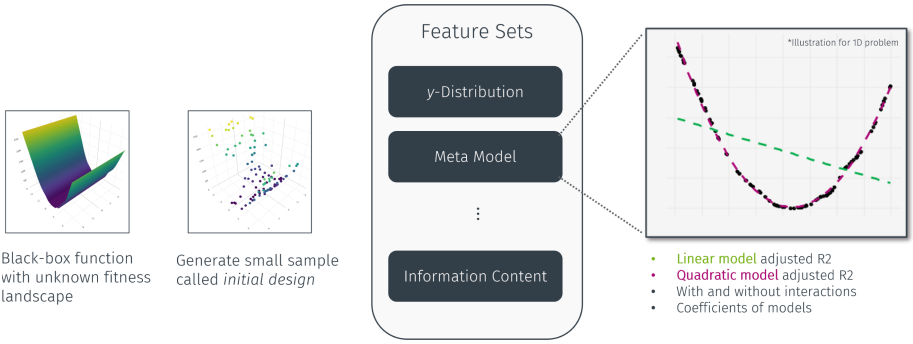


Fig. 1. General procedure of ELA feature generation. Based on the initial design several feature sets can be computed. The feature set 'meta model' fits the several linear and quadratic models and uses model coefficients as well as the adjusted R^2 as features.

optimization problems, *exploratory landscape analysis* (ELA) has been developed [17]. ELA encompasses several different techniques to quantify the landscape of a black-box problem with numerical surrogates. These range from summary statistics as well as model coefficients to distance-based metrics. The aforementioned calculations are based on a so-called 'initial design' $\mathbf{S}$ which essentially is a small sample of the search space $\mathbf{X}$ and the corresponding objective values $\mathbf{Y}$. The size of this initial design usually depends linearly upon the dimensionality, D, of a given problem instance and a common choice is $50D$. An overview of the general procedure of ELA feature generation is depicted in Fig. 1.

Two prominent software packages exist to calculate ELA features, namely the R package `flacco` [10] and the Python package `pflacco` [29]. These software packages also include the advances made since the inception of ELA, i.e., they also include feature sets proposed over the last decade such as the works of [8,20]. In this study, we use the same ELA feature sets as used by [30], i.e. y-distribution, meta model, dispersion, and information content. Note that we adhere to the recommendation of normalizing the objective space prior to feature computation [28] in order to ensure shift and scale invariance.

The original work pertaining to ELA features covered several feature sets of which two are used in this study. These are meta model and y-distribution with nine and three features respectively. The feature set **meta model** fits a linear and quadratic model (with and without interactions) to the initial design where the model coefficients and the model quality serve as individual features. This provides insight about the degree of linearity and convexity of a given problem instance. The feature set **y-distribution**, on the other hand, only focuses on the objective value distribution of the initial design. In particular, the kurtosis and skewness are measured [17]. The **dispersion** feature set consists of 16 features. It segments the objective values of the initial design into different subsets based on varying quantile levels, such as 10% and 25%. Within each subset, measurements are taken for both mean and median distances in the deci-

sion space. This analysis provides valuable insights into the distribution of local optima, indicating whether they are concentrated in specific regions or dispersed throughout the landscape [15]. Five features are part of **information content**. To derive this feature set, a sequence of random walks is performed across the initial design. At each step, comparisons are drawn between the current observation and the subsequent one. These comparisons yield diverse metrics aimed at delineating the landscape's attributes, encompassing aspects of smoothness, ruggedness, and neutrality [20]. Identical to information content, the feature set **nearest better clustering** includes five features that are metrics and ratios computed between two different sets of distances. One of these sets consists of distances of each observation to its nearest neighbor whereas the other set covers distances between observation and its nearest *better* neighbor [8].

3.2 Preprocessing Scheme Based on Target-Encoding

In [30], the authors developed a preprocessing scheme comprising several steps applied to the initial design **S**. One of these concerns the representation of categorical variables with real-valued numbers. In particular, two encoding methods, namely *one-hot encoding* and *target-encoding* (TE) [18], were contrasted with each other. The authors recommend the usage of TE as it performed comparatively better than one-hot encoding and requires less computation time.

TE originates from the *machine learning* (ML) community and is typically used to encode categorical variables for learners incapable of processing these types of variables [18]. TE measures the influence of a given category of a categorical variable on the target in question, which in [30] is modeled as the objective value Y of the initial design **S**. For a given categorical variable X_i and corresponding specific category $j \in X_i$, TE computes the arithmetic mean of the objective values Y and a subset of $Y_j \subseteq Y$ which only contains objective values where $X_i = j$. A weighted mean is calculated based on these two averages and this new value j' replaces the category j. One can think of j' as a value which measures the average deviation from the arithmetic mean of Y for a category j.

3.3 SHAP-Encoding

Even though [30] achieve promising results based on TE the sole purpose of the method was to transform any categorical variable to a numerical value without adding new dimensions. Thus, TE does not capture the variable's effect for individual observations. Rather, it aggregates the mean effect of every feasible category of that variable on the chosen target and replaces the categories with the resulting numerical value. Every distinct category of a categorical variable is, therefore, assigned the same static real-valued number across all its occurrences in the data set. A more nuanced approach can be achieved using *SHapley Additive exPlanations* (SHAP) [16]. SHAP is a unified framework for the interpretation of ML model predictions and is based on Shapley values [32]. Shapley values were first introduced in the context of cooperative game theory as a means of quantifying the average contribution of every team member in a coalition game. This is done by playing the game in all possible coalitions of team members

and averaging over the difference in the outcome before a team member joins and after the member has joined a coalition. This accounts for all interaction effects between the different players and leads to desirable properties like local accuracy, missingness and consistency [16]. The SHAP framework transfers this approach to the ML domain by conceptualizing every observation in a dataset as a game. For instance, in the case of tabular data, this means that every row in the dataset is a distinct game while every feature value in that row can be viewed as a player. The game outcome corresponds to the prediction of the ML model of interest. The Shapley values can than be calculated according to Eq. 2 [16] by

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} \left[f_x(z') - f_x(z' \setminus i) \right]. \tag{2}$$

Here, ϕ_i is the Shapley value for the feature value expressed by feature i within the observation x (e.g. tabular data row) while being subject to the ML model f. M denotes the number of features in the dataset, x' is the *simplified input* which is commonly used by explanation models and can be mapped to the original input by any $h_x(x') = x$. Finally, $z' \in \{0, 1\}^M$ denotes the number of features that are part of the respective coalition. In practice, ML models usually exhibit static architectures. This means the number of input features cannot be arbitrarily varied. To overcome this hurdle every feature that is not part of a coalition instance is sampled randomly from all values exhibited by it within the entire dataset. The idea is to randomize the feature and, thus, make it lose its predictive power. The SHAP framework also provides several solutions for the NP-hardness of the Shapley value calculation exhibited by Eq. 2. The current default relies on antithetic sampling which considers an entire permutation of all features in forward and reverse direction [19]. Thus, SHAP allows to calculate Shapley values even for a large number of features. This provides a local (for a single observation) and additive quantization of the contribution of every feature to the deviation from the expected value. Here, the expected value is the prediction if no feature is part of the coalition. As SHAP values are synonymous to Shapley values applied in the ML domain, we refer to them as SHAP values. In this work, we train a ML model on the initial design $\mathbf{S}$ of ELA where the input of the model is the sample of decision space X and corresponding objective values Y. For each individual observation in X, we now compute the SHAP values of all **categorical** decision variables on the prediction (i.e., the objective value) of the model. We then replace every category in this specific observation with the respective SHAP values. This process is repeated until every observation in X is iterated over and all non-numeric values are replaced.

4 Experimental Setup

4.1 Benchmark Problems

We use HPO problems as representatives of the in general broader class of MVP problems which also e.g. prominently comprises engineering applications such as aircraft design [2,35] and design of induction motors [13]. The HPO problem set

Table 1. Overview of available scenarios in YAHPO Gym, the search space, the number of problem instances per scenario and information whether the search domain is hierarchical or not.

Scenario	Search Space (cont, int, cat)	# Instances	H
rbv2_glmnet	3D: (2, 0, 1)	115	–
rbv2_rpart	5D: (1, 3, 1)	117	–
rbv2_aknn	6D: (0, 4, 2)	118	–
rbv2_svm	6D: (3, 1, 2)	106	✓
iaml_ranger	8D: (2, 3, 3)	4	✓
rbv2_ranger	8D: (2, 3, 3)	119	✓
iaml_xgboost	13D: (10, 2, 1)	4	✓
rbv2_xgboost	14D: (10, 2, 2)	119	✓

is provided by the 'Yet Another Hyperparameter Optimization Gym' (YAHPO Gym) [24] benchmark. All considered HPO problems are classification tasks and associated with a specific learner, its respective hyperparameters that need to be tuned and final metric which represents quality of the learner on this given training task. In the context of YAHPO Gym, a scenario comprises several HPO problems for a specific learner. The different problems within a scenario are instantiated using various datasets sourced from OpenML [34], with the objective metric being the misclassification error, which we aim to minimize.

For the sake of comparison, we use the same subset of scenarios as the authors of [30]. These are presented in Table 1, which delineates the dimensions of the search space and the distribution of continuous, integer, and categorical decision variables. In total, this encompasses 702 individual problem instances. It is essential to emphasize that the instances within a scenario may not necessarily share similar levels of difficulty or overall landscape characteristics. Factors such as the presence of multiple peaks or the unimodality of the fitness landscape are contingent not only upon the scenario but also the specific dataset employed. Thus, the categorization outlined in Table 1 serves solely for illustrative purposes and should not be considered as indicative of problem similarity.

4.2 Algorithm Portfolio

The performance data of our algorithm portfolio was provided by the authors of [30] which benefits comparability. The circumstances under which this data was created are briefly described in the following. `SMAC3` [12], `Optuna` [1], `pymoo` [4], and random search (RS) were compared on the previously described benchmark functions. The allotted budget of each algorithm amounts to $100D$ function evaluations where D denotes the problem dimensionality. Each algorithm is executed on each problem instance of YAHPO Gym 20 times. The runtime of these 20 repetitions of an algorithm until a predefined target is reached is aggregated into a single value called *expected running time* (ERT) [6].

A target can be perceived as a threshold from which ontowards we deem the achieved results satisfactory. In this particular setting, YAHPO Gym models HPO problems with an underlying classification task. The chosen metric to optimize for is the misclassification error. In theory, the best performance is achieved with a misclassification error of zero. However, in practice this might not be reachable as this is constrained by the data set, the chosen ML model and the box-constraints for its hyperparameters. Hence, [30] determine an individual target for each problem instance. This target is calculated by (1) concatenating all optimizer traces of the four algorithms (including their 20 repetitions), (2) ordering the objective values in ascending order, and (3) selecting the 0.01-quantile of the objective values. Thereby, the target is chosen in a manner that it is still challenging for all four algorithms to achieve but simultaneously guarantees that at least one algorithm is able to solve each individual problem instance.

4.3 Exploratory Landscape Feature Generation

While we use the data provided by [30] for the target-encoded ELA features, we generate the ELA feature values based on our proposed SHAP-encoding by using the Python package `pflacco` [29]. The initial design $\mathbf{S}$ is generated uniformly at random in the feasible domain of each problem, with a sample size of $50D$. Again, D represents the dimensionality of any given problem instance. Since the calculation of SHAP values for a single prediction has a complexity of $O(2^N)$, we make use of the `PermutationExplainer` from the Python package provided with the `SHAP` publication [16]. This is the current default `Explainer` of the package and employs antithetic sampling to calculate the SHAP values [19]. The ML model, which serves as basis to calculate the respective SHAP values, is a random forest with a default configuration of the Python package `scikit-learn` [22].

After replacing each non-numeric value in the initial design $\mathbf{S}$ with its corresponding SHAP value, we normalize each decision variable and the objective value of a given initial design independently, constraining them within the interval $[0, 1]$ as recommended in [28,30]. Each set of ELA features is calculated 20 times for a given problem instance to produce more robust results.[1]

4.4 Construction of Algorithm Selectors

Our approach conceptualizes the AAS [7] scenario as a multi-class classification problem. The ELA features serve as input for the AS model while the class label is determined by the best-performing algorithm (lowest ERT value) for each particular instance. We construct two AAS models, one trained on target-encoded and one trained on SHAP-encoded ELA features. It is pertinent to mention that our dataset encompasses a total of $14,040$ observations for each encoding variant, resulting from the generation of ELA features from 20 different samples per

[1] Corresponding source code and results can be found on https://github.com/konsdt/PPSN-SHAP-TE-ELA.

problem instance (of which there are 702). The class label per instance remains consistent across these 20 repetitions.

For the sake of comparability, we use the same ML model as used by [30], i.e., a random forest provided by the Python package `scikit-learn` [22].

We assess the model through a 10-fold cross-validation approach, ensuring that all repetitions of a specific problem instance are grouped within a single fold. This strategy prevents the dispersion of repetitions across multiple folds, which could otherwise result in instances being both trained and tested simultaneously. Such an arrangement helps maintain the integrity of the evaluation process, minimizing potential biases and ensuring robustness in our analysis.

As delineated below, the two models exhibit complementary behavior, prompting us to subsequently propose two strategies that capitalize on this synergy.

5 Results

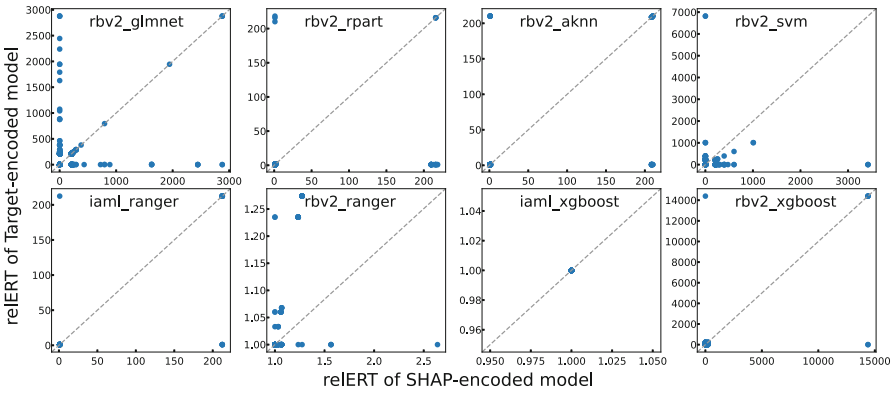


Fig. 2. Performance of both algorithm selectors. The x-axis shows the relERT values of the SH model whereas the y-axis shows the relERT values of the TE model. Points on the grey line exhibit (nearly) identical relERT values for both models. Points above the grey line represent instances where the SH model produces a better performance. Points below the grey line represent the other case. (Color figure online)

The comparison of our SHAP value based algorithm selector (SH model) contrasted against the target-encoded based algorithm selector (TE model) shows a very similar performance between both models as reflected by the TE and SH columns in Table 2. A closer look into the performance distribution compartmentalized into the respective YAHPO Gym scenarios provides a more insightful view. This is shown in Fig. 2. The figure depicts the performances of both models for any given problem instance where the x-coordinate is determined by the relERT value of the SH model and the y-coordinate by the relERT value of the

TE model respectively. Points on the gray line represent instances where the performance of both models is either identical or very similar. The relERT values are determined by dividing all ERT values of a single problem instance by the ERT value of the virtual best solver (VBS). The VBS embodies a theoretical algorithm selector that unfailingly selects the optimal algorithm from our portfolio for every instance.

Again, we can discern that the distribution and consequently summary statistics like the mean performance of the models are similar. Yet, we can also observe that there are a multitude of problem instances where the SH model excels while the TE model exhibits a poor performance and vice versa. For example, this dichotomy is present in the scenario `rbv2_svm`, where points in the bottom left show this behavior. At the same time there are also points in the top left and bottom right corner. Especially, the latter contribute significantly to the complementarity of both models since in these cases one of the models manages to select the VBS while the other selects the worst solver from the portfolio.

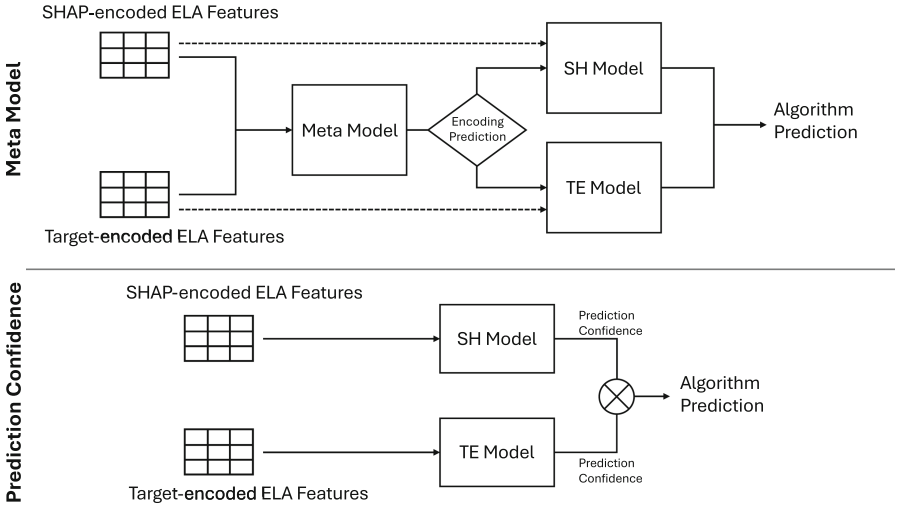


Fig. 3. Illustration of our two suggested approaches to capitalize on the complementary of the two AAS strategies. The meta model classifier utilizes ELA features derived from SHAP values and TE for each problem instance. With a binary target variable representing either the TE or SH model based on better prediction performance, the classifier selects the appropriate algorithm selector. Subsequently, the chosen selector utilizes ELA features specific to its encoding type and selects an algorithm from the portfolio. Alternatively, the prediction confidence approach relies on comparing prediction probabilities from independent AAS models, with the higher probability indicating greater confidence in the prediction.

Even after careful and rigorous analysis we cannot identify the reasons that lead to this dichotomous state. Nevertheless, exploiting this complementary behavior of both encoding methods produces an overall better result. The

full potential of hybridizing both encoding variants can be seen in the column **'Hybrid'** in table Table 2. The values shown there are determined by always selecting the superior algorithm between the two predicted by the TE and the SH model. Looking at the relERT averaged over all problem instances we see that even the hybrid approach still performs 10.82 times worse than the virtual best solver (VBS) of the selected portfolio. To bring this into context we also report the relERT values of the single best solver (SBS) which, as already reported in [30], is `SMAC3`. In agreement with [30] we see that algorithm selection exhibits a large improvement over the general selection of a single solver for all problems. Even though there remains room for improvement the hybridization is a substantial advancement over the individual approaches as it bears 40.42% performance increase over the SH model and 39.49% over the TE model. To leverage this relationship we suggest two approaches which are conceptualized in Fig. 3. In the first approach that can be seen in the top of Fig. 3, we utilize an additional ML model which is responsible for determining which of the two algorithm selectors to use. We call this new model **'meta model'**.

The 'meta model' approach is based on a random forest classifier that receives both, the ELA features based on SHAP values and the ELA features based on TE for every problem instance respectively. The target variable has two classes, and can either be the TE model or the SH model, depending on which was able to predict the better solver for the problem. It then decides which of the two algorithm selectors it should use. Thereafter, the chosen algorithm selector uses only the ELA features according to its respective encoding type and chooses an appropriate algorithm out of the portfolio.

The second approach relies solely on the comparison of prediction probabilities between the two independent AS models. We evaluate both selectors at the same time and trust the prediction which exhibits a higher prediction probability. Doing so, we interpret the prediction probability as prediction confidence of the respective model and thus denote this approach as **'Confidence'**. For a multiclass random forest classifier the predicted probabilities for each class of an input sample are determined by averaging the predicted probabilities across all trees within the forest. In each individual tree, the class probability is calculated as the proportion of samples belonging to the same class within a leaf node. A high-level representation of that concept is illustrated in the bottom of Fig. 3. We asses both approaches using 10-fold cross validation and the results are shown in the columns labeled 'Meta' and 'Confidence' of Table 2. The highlighted values mark the best selection strategy for the respective scenario. We find that the prediction confidence approach performs the best across all problem instances. When examining individual scenarios, the selection method relying on prediction confidence is outperformed by the alternative meta-model approach in only two cases. In the two instances where its performance aligns with the meta-model approach, there is little to no potential for enhancement as the SBS equals or nearly equals the VBS.

To further illustrate how well both approaches perform we lean on the idea of relERT calculation. Since the values in the 'Hybrid' column of Table 2 result

Table 2. Performance of the different approaches which is measured as the arithmetic mean of the relERT value. Highlighted values indicate the approaches that performed best for a given setting. The column 'Hybrid' is not considered since it represents a virtual best encoding choice.

Scenario	# Instances × # samples	SBS	TE	SH	Hybrid	Meta	Confidence
rbv2_glmnet	2 120	558.45	50.90	47.05	29.08	45.35	44.90
rbv2_rpart	2380	28.96	1.66	3.11	1.39	1.76	2.20
rbv2_aknn	2 360	4.54	2.77	3.29	2.15	2.32	2.85
rbv2_svm	80	330.86	19.18	15.80	4.89	14.39	10.47
iaml_ranger	2 300	53.85	40.63	51.20	37.99	40.63	45.92
rbv2_ranger	80	1.02	1.01	1.01	1.01	1.01	1.01
iaml_xgboost	2 340	1.00	1.00	1.00	1.00	1.00	1.00
rbv2_xgboost	2 380	127.87	32.44	38.48	25.57	32.36	26.30
All	14 040	169.19	17.88	18.16	10.82	16.18	14.68

Table 3. Performance of the different approaches which is measured as the arithmetic mean of the relERT normalized with respect to the VBE given by the 'Hybrid' column.

Scenario	# Instances × # samples	TE	SH	Hybrid	Meta	Confidence
rbv2_glmnet	2 120	22.50	17.95	1.00	16.31	16.89
rbv2_rpart	2380	1.28	2.72	1.00	1.37	1.82
rbv2_aknn	2 360	1.62	2.15	1.00	1.18	1.71
rbv2_svm	80	15.28	11.90	1.00	10.50	7.36
iaml_ranger	2 300	3.64	14.21	1.00	3.64	8.93
rbv2_ranger	80	1.00	1.00	1.00	1.00	1.00
iaml_xgboost	2 340	1.00	1.00	1.00	1.00	1.00
rbv2_xgboost	2 380	7.88	13.92	1.00	7.78	1.73
All	14 040	8.01	8.17	1.00	6.20	4.99

from always choosing the best feature encoding strategy we consider it to be the **'virtual best encoding' (VBE)**. Hence, we normalize all relERT values by dividing with the VBE values. The resulting values are shown in Table 3. Continuing this analogy, the **'single best encoding' (SBE)** from the two compared encoding methods is target-encoding. We refrain from showing the SBS column in Table 3 since it reveals no additional relevant information. In total, the prediction confidence approach is able to close the gap between SBE

and VBE by 43.10%. The meta model also achieves an overall improvement over the SBE but this only amounts to a gap closure of 25.84%.

6 Conclusion

Landscape analysis has been used in a variety of studies with different objectives and scopes over the last decade. However, it has been largely limited to either the continuous or combinatorial optimization domain. Recent advances of [26,27,30] have iteratively extended this to the domain of mixed-variable problems. One of the main contributions of [30] is the proposition to transform categorical variables into continuous representations. This is achieved by employing techniques such as one-hot encoding and target-encoding.

In our work, we extend this particular component by introducing SHAP values as an alternative encoding method to transform categorical into continuous values. We demonstrate the merits of our encoding variant in an automated algorithm selection setting and compare our results with the results of [30]. While we cannot determine any substantial difference between the algorithm selector based on the SHAP-encoding and the one based on target-encoding, we show that both encoding methods are performance complementary and achieve a much better performance when switching between them. We propose two strategies to hybridize both encoding methods. One devised approach uses a meta model that decides which subsequent algorithm selector to use. The other compares the prediction probabilities of the AS models trained on both encodings and interprets them as prediction confidence. Both approaches are able to substantially improve the existing results based on target-encoding. In our setting these results are considered the single best encoding while the hybridization of both discussed encodings is considered the virtual best encoding. While the meta model closes the gap between both by 25.84%, the prediction confidence method is able to achieve a substantial improvement with a gap closure of 43.10%.

Despite this advancement, there remains room for improvement which becomes especially clear when considering that the relERT value of the virtual best encoding is still quite high with 10.82. At the forefront of research questions unsolved is, what the underlying mechanisms are which lead to the performance complementary behavior of these two encoding variants. Revealing these will foster our understanding and will help us to incorporate a more sophisticated encoding mechanism directly into the ELA feature calculation. We also plan to further investigate if different ML models during SHAP value calculation can change the expressiveness of the SHAP-encoded features and will also extend our scope to MVP problems beyond the so far considered subclass of HPO problems.

Acknowledgements. This work was realized with the financial support of ANR project ANR-22-ERCS-0003-01 and of CNRS Sciences informatiques project *IOHprofiler*.

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

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
2. Barjhoux, P.J., Diouane, Y., Grihon, S., Bettebghor, D., Morlier, J.: Mixed variable structural optimization: toward an efficient hybrid algorithm. In: Schumacher, A., Vietor, T., Fiebig, S., Bletzinger, K.U., Maute, K. (eds.) Advances in Structural and Multidisciplinary Optimization, pp. 1880–1896. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-67988-4_140
3. Belkhir, N., Dréo, J., Savéant, P., Schoenauer, M.: Per instance algorithm configuration of CMA-ES with limited budget. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO), pp. 681–688. ACM (2017). https://doi.org/10.1145/3071178.3071343
4. Blank, J., Deb, K.: pymoo: multi-objective optimization in Python. IEEE Access **8**, 89497–89509 (2020)
5. Guzowski, H., Smolka, M.: Configuring a hierarchical evolutionary strategy using exploratory landscape analysis. In: Silva, S., Paquete, L. (eds.) Proceedings of Genetic and Evolutionary Computation Conference (GECCO), Companion, pp. 1785–1792. ACM (2023). https://doi.org/10.1145/3583133.3596403
6. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2010: experimental setup. Research Report RR-7215, INRIA (Mar 2010). https://inria.hal.science/inria-00462481
7. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: survey and perspectives. Evol. Comput. **27**(1), 3–45 (2019). https://doi.org/10.1162/evco_a_00242
8. Kerschke, P., Preuss, M., Wessing, S., Trautmann, H.: detecting funnel structures by means of exploratory landscape analysis. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 265–272. GECCO '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2739480.2754642
9. Kerschke, P., Trautmann, H.: Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. Evol. Comput. **27**(1), 99–127 (2019). https://doi.org/10.1162/evco_a_00236
10. Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco. In: Bauer, N., Ickstadt, K., Lübke, K., Szepannek, G., Trautmann, H., Vichi, M. (eds.) Applications in Statistical Computing. SCDAKO, pp. 93–123. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25147-5_7
11. Vermetten, D., et al.: Per-run algorithm selection with warm-starting using trajectory-based features. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) Parallel Problem Solving from Nature - PPSN XVII, pp. 46–60. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_4

12. Lindauer, M., et al.: SMAC3: a versatile bayesian optimization package for hyperparameter optimization. J. Mach. Learn. Res. **23**, 1–9 (2022). https://www.jmlr.org/papers/volume23/21-0888/21-0888.pdf
13. Liuzzi, G., Lucidi, S., Piccialli, V., Villani, M.: Design of induction motors using a mixed-variable approach. Comput. Manage. Sci. **2**(3), 213–228 (2005). https://doi.org/10.1007/s10287-005-0024-2
14. Lucidi, S., Piccialli, V., Sciandrone, M.: An algorithm model for mixed variable programming. SIAM J. Optim. **15**(4), 1057–1084 (2005). https://doi.org/10.1137/S1052623403429573
15. Lunacek, M., Whitley, D.: The dispersion metric and the CMA evolution strategy. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 477–484. GECCO 2006, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1143997.1144085
16. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30, pp. 4765–4774. Curran Associates, Inc. (2017). http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
17. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 829–836. GECCO 2011, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2001576.2001690
18. Micci-Barreca, D.: A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. SIGKDD Explor. Newsl. **3**(1), 27–32 (2001). https://doi.org/10.1145/507533.507538
19. Mitchell, R., Cooper, J., Frank, E., Holmes, G.: Sampling permutations for shapley value estimation. J. Mach. Learn. Res. **23**(1), 1–46 (2022)
20. Muñoz Acosta, M.A., Kirley, M., Halgamuge, S.K.: Exploratory landscape analysis of continuous space optimization problems using information content. IEEE Trans. Evol. Comput. (TEVC) **19**(1), 74–87 (2015). https://doi.org/10.1109/TEVC.2014.2302006
21. Muñoz, M.A., Sun, Y., Kirley, M., Halgamuge, S.K.: Algorithm selection for black-box continuous optimization problems: a survey on methods and challenges. Inf. Sci. **317**, 224–245 (2015). https://doi.org/10.1016/j.ins.2015.05.010, https://www.sciencedirect.com/science/article/pii/S0020025515003680
22. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
23. Pelamatti, J., Brevault, L., Balesdent, M., Talbi, E.G., Guerin, Y.: How to deal with mixed-variable optimization problems: an overview of algorithms and formulations. In: Schumacher, A., Vietor, T., Fiebig, S., Bletzinger, K.U., Maute, K. (eds.) Advances in Structural and Multidisciplinary Optimization, pp. 64–82. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-67988-4_5
24. Pfisterer, F., Schneider, L., Moosbauer, J., Binder, M., Bischl, B.: YAHPO Gym - an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In: Guyon, I., Lindauer, M., van der Schaar, M., Hutter, F., Garnett, R. (eds.) Proceedings of the First International Conference on Automated Machine Learning. Proceedings of Machine Learning Research, vol. 188, pp. 3/1–39. PMLR (2022). https://proceedings.mlr.press/v188/pfisterer22a.html

25. Pikalov, M., Mironovich, V.: Automated parameter choice with exploratory landscape analysis and machine learning. In: Krawiec, K. (ed.) Proceedings of Genetic and Evolutionary Computation Conference (GECCO), Companion, pp. 1982–1985. ACM (2021). https://doi.org/10.1145/3449726.3463213
26. Pikalov, M., Mironovich, V.: Parameter tuning for the $(1 + (\lambda, \lambda))$ genetic algorithm using landscape analysis and machine learning. In: Jiménez Laredo, J.L., Hidalgo, J.I., Babaagba, K.O. (eds.) EvoApplications 2022. LNCS, vol. 13224, pp. 704–720. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-02462-7_44
27. Prager, R.P., Trautmann, H.: Investigating the viability of existing exploratory landscape analysis features for mixed-integer problems. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, pp. 451–454. GECCO 2023 Companion, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3583133.3590757
28. Prager, R.P., Trautmann, H.: Nullifying the inherent bias of non-invariant exploratory landscape analysis features. In: Correia, J., Smith, S., Qaddoura, R. (eds.) Appl. Evol. Comput. Springer International Publishing, Cham (2023)
29. Prager, R.P., Trautmann, H.: Pflacco: Feature-based landscape analysis of continuous and constrained optimization problems in Python. Evol. Comput., 1–25 (2023). https://doi.org/10.1162/evco_a_00341
30. Prager, R.P., Trautmann, H.: Exploratory landscape analysis for mixed-variable problems. CoRR arXiv preprint arXiv:2402.16467 (2024). https://arxiv.org/abs/2402.16467, under revision with IEEE Transactions on Evolutionary Computation
31. Seiler, M.V., Prager, R.P., Kerschke, P., Trautmann, H.: A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 657–665. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3512290.3528834
32. Shapley, L.S.: A value for n-person games. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to the Theory of Games II, pp. 307–317. Princeton University Press, Princeton (1953)
33. Smith-Miles, K.A.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. **41**(1) (2009). https://doi.org/10.1145/1456650.1456656
34. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. SIGKDD Explor. **15**(2), 49–60 (2013). https://doi.org/10.1145/2641190.2641198
35. Venter, G., Sobieszczanski-Sobieski, J.: Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization. Struct. Multidiscip. Optim. **26**(1), 121–131 (2003). https://doi.org/10.1007/s00158-003-0318-3

iMOPSE: a Comprehensive Open Source Library for Single- and Multi-objective Metaheuristic Optimization

Konrad Gmyrek, Paweł B. Myszkowski(✉), Michał Antkiewicz, and Łukasz P. Olech

Faculty of Information and Communication Technology, Wrocław University of Science and Technology, Wrocław, Poland
{konrad.gmyrek,pawel.myszkowski,michal.antkiewicz}@pwr.edu.pl, lukasz.piotr.olech@gmail.com

Abstract. The Intelligent Multi-Objective Problem Solving Environment (iMOPSE) is a robust open-source C++ library designed to tackle NP-hard optimization problems. It hosts a suite of multi-objective optimization algorithms, including state-of-the-art NSGA-II, MOEA/D, SPEA2, or NTGA2, complemented by a set of single-objective optimization metaheuristics such as Genetic Algorithms, Differential Evolution, Ant Colony Optimization, Tabu Search, Simulated Annealing, and Particle Swarm Optimization. One of iMOPSE's notable strengths lies in its ability to handle classical NP-hard problems with constraints, ranging from the Traveling Salesman and Traveling Thief to Capacitated Vehicle Routing and Multi-Skill Resource-Constrained Project Scheduling Problems. Its flexible encoding mechanism adeptly manages different problems and facilitates the utilization of specialized operators. Moreover, iMOPSE offers pre-configured problem instances and method setups, along with a suite of tools for data collection, visualization, and analysis, bolstering its efficacy for rigorous research and optimization result interpretation. iMOPSE also provides extensive customization options, enabling researchers to explore and research various optimization methods and scenarios effectively. Its user-friendly interface streamlines setup procedures through intuitive input parameters and configuration files, ensuring accessibility across Windows and Unix-based operating systems. Together, these features position iMOPSE as a comprehensive solution for addressing real-world optimization challenges.

Keywords: Optimisation · Multi-Objective Optimization · Metaheuristic · NP-hard problems · Software Library

1 Introduction

Over the years, the field of metaheuristic optimization has significantly evolved, propelled by the escalating complexity of real-world challenges. Within this landscape, the introduction of the iMOPSE C++ library contributes to the development of optimization methods and enhances the transparency of research in two

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 170–184, 2024.
https://doi.org/10.1007/978-3-031-70068-2_11

ways. First, it enriches the field of metaheuristic single- and multi-objective optimization by supplying a repository of problem instances and established techniques, facilitating comparative studies and experimentation. Second, it empowers the broader community by providing easy access to a suite of state-of-the-art methods and NP-hard combinatorial optimization problems ready for immediate application. This dual role of iMOPSE, as an open-source library, furthers research in metaheuristic optimization and provides access to sophisticated problem-solving tools, playing a vital role in the collective progress of this field.

This paper aims to present the core principles and contributions of iMOPSE, highlighting its potential role in metaheuristic optimization and its relevance to the current demands of research in this field.

At its core, iMOPSE serves as a solution to the demand for an accessible suite of optimization methods designed to solve complex NP-hard combinatorial problems frequently encountered in real-world scenarios, such as scheduling, routing, and resource allocation challenges. Consequently, it stands as a comprehensive toolkit that bridges the gap between theoretical exploration and practical application in problem-solving.

A fundamental aspect of iMOPSE's design is its emphasis on scalability and extensibility. The library's architecture has been developed to be highly adaptable, meeting a variety of scenarios. This versatility is crucial for researchers who need to tailor their approaches to specific problem characteristics and constraints. iMOPSE's flexible structure makes it a useful tool for research, as it allows for easy experimentation, comparison, and analysis of different methods within the metaheuristic field.

The rest of the article is structured as follows. In Sect. 2 a list of alternative, publicly available tools and libraries is given. The investigated iMOPSE architecture is shortly defined in Sect. 3. Next Sect. 4 contains a description of supportive utils and tools useful in research. The Sect. 5 describes a case study of iMOPSE usage – the application of a multi-objective method to a selected NP-hard Traveling Thief Problem. Lastly, the paper is concluded in Sect. 6, where we discuss further development planned for the iMOPSE library.

2 Related Work

In recent years, the field of metaheuristic optimization has seen notable advancements, resulting in the emergence of various libraries and frameworks, each crafted to cater to the specific requirements of diverse optimization problems. These tools, created across various programming languages, offer distinct features and capabilities to suit different research and practical needs. Among them, iMOPSE specializes in targeting certain problems and applications empowered by the usage of flexible encoding mechanisms. It allows iMOPSE to provide solutions that are finely tuned to the unique challenges of certain types of optimization scenarios. Below, we highlight selected libraries (developed and regularly updated – last in 2024), showcasing their features.

Opt4J [16] is a Java-based framework renowned for its modular approach. It primarily focuses on evolutionary algorithms and multi-objective optimization,

making it a go-to choice for problems that require evolutionary computation. Its Java implementation offers a familiar environment for a large segment of developers, contributing to its popularity and extensive use in academic research [11]. Opt4J implements several classic metaheuristics like Simulated Annealing (SA), Tabu Search (TS), or Differential Evolution (DE), and benchmarks like *ZDT*, and *DTLZ*. One of the standout features of Opt4J is its graphical user interface, which assists in both the configuration of optimization parameters and the visualization of the optimization process. Furthermore, Opt4J is open source and available under the MIT license, which allows for broad usage and modifications by the community.

Metaheuristics.jl [4] implemented in Julia programming language is noted for its high performance, comparable to C, making it well-suited for complex problems and numerical computations. This Julia-based implementation attracts a niche audience proficient in Julia, as it offers a robust array of features that benefit optimization tasks across various applications. The library includes a range of metaheuristic algorithms such as Genetic Algorithm (GA), DE, Particle Swarm Optimization (PSO), or Artificial Bee Colony (ABC). It also supports advanced multi-objective algorithms like Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) or Non-dominated Sorting Genetic Algorithm (NSGA-II) and includes quality measures like $InvertedGenerationalDistance$ (IGD) and $HyperVolume$ (HV). While $Metaheuristics.jl$ includes implementations of several standard test problems like *ZDT* and *DTLZ*, which serve as benchmarks for performance evaluation, it is important to note that these test problems are somewhat limited. Users looking to tackle other types of problems will need to implement these manually, which adds an extra layer of complexity when using the library for diverse optimization challenges. The framework is regularly updated and its modular structure, integration of various metaheuristic algorithms, and constant updates facilitate the optimization of complex multi-objective and constrained problems.

jMetal [3] is a prominent Java-based library utilized extensively in evolutionary algorithm research. Renowned for its broad array of multi-objective optimization algorithms, primarily focused on NSGA-II and MOEA/D, which underscores its appeal within the academic community focused on exploring multi-objective optimization scenarios. The library's commitment to a Java-based ecosystem has significantly contributed to its popularity among researchers, providing a stable and familiar development environment. The library supports classic optimization challenges like the TSP, additionally including a selection of multi-objective continuous problems, such as those found in benchmark datasets like *dtls* or the *CEC'15* competition. $jMetal's$ design facilitates extensive experimentation and research in evolutionary algorithms, and the library's robust features and ongoing updates ensure it remains at the forefront of technology in this specialized field.

PyGMO (Python Parallel Global Multi-objective Optimizer) [6] is recognized for its robust implementation in both Python and C++, making it particularly appealing for complex, large-scale optimization problems. Supported by

the European Space Agency, *PyGMO* is utilized extensively in aerospace engineering, among other fields that demand high-performance optimization solutions. It is renowned for facilitating global multi-objective optimization with a focus on continuous problems. The library includes a comprehensive array of optimization methods, with more than 15 optimization methods primarily encompassing single-objective PSO and DE. Additionally, it supports classic multi-objective algorithms such as the Strength Pareto Evolutionary Algorithm (SPEA2), NSGA-II, and MOEA/D. $PyGMO's$ repository of problems covers a broad spectrum from single- to multi-objective scenarios, incorporating both constrained and unconstrained environments.

Platypus (Multi-objective Optimization in Python) [5] is a Python library designed for solving multi-objective optimization problems, acclaimed for its user-friendliness and adaptability. It supports various evolutionary algorithms, including NSGA-II, NSGA-III, MOEA/D, and SPEA2, making it highly versatile for different optimization needs. The library's intuitive interface integrates seamlessly with Python's extensive data science tools, enhancing its usability in academic and practical contexts. The focus of *Platypus* primarily lies on continuous optimization problems and includes features for constraint handling, which broadens its applicability across various domains needing sophisticated optimization solutions. Its straightforward design is particularly appreciated within the Python community, appealing to users who require a combination of efficiency and ease of use in their optimization tasks. For developers and researchers interested in utilizing *Platypus* for their multi-objective optimization needs, the library presents a reliable and powerful tool that leverages Python's capabilities to address complex optimization challenges effectively.

Pymoo [12] pymoo is another Python-based framework, which has gained attention for its comprehensive support for multi-objective optimization. It offers a variety of state-of-the-art algorithms for multi-objective problems, including genetic algorithms and evolutionary strategies. The library is designed with usability and flexibility in mind, making it a good choice for both academic research and industry applications. $Pymoo's$ continuous updates and active development community reflect its relevance and adaptability to the evolving needs in the field of optimization. Similarly to the Platypus library, *Pymoo* also supports constraint handling and focuses on continuous problems.

PyMetaheuristic [7] is a Python library focused on metaheuristic optimization algorithms, known for its simplicity and ease of use. It caters to users seeking implementations of popular algorithms. The library encompasses a variety of algorithms, including SA, GA, and PSO. These are suitable for a broad range of optimization challenges. $pyMetaheuristic's$ approachable Python-based design lowers the barrier to entry, making it accessible for those familiar with Python, and is particularly beneficial for rapid prototyping and experimentation in diverse optimization scenarios.

Table 1 provides a comparative overview of the mentioned metaheuristic optimization libraries, highlighting some of their features. We present general characteristics of each framework to sum up related work sections and provide back-

ground for comparison. This comparative overview highlights a unique focus on special types of problems.

Table 1. Comparative Overview of Optimization Libraries and Frameworks

Library	Visual.	Impl.	Opt.	Problems
Opt4J	GUI	Java	MO/SO	Benchmark (zdt, dltz, knapsack), 6 in total
Metaheuristics.jl	No	Julia	MO/SO	Benchmark (zdt, dltz)/ Combinatorial, over 20
jMetal	No	Java	MO/SO	Benchmark (CEC2021, zcat), TSP, over 20
PyGMO	No	Python/C++	MO/SO	Benchmark (CEC2014, WFG), over 20
Platypus	Yes	Python	MO	Benchmark (zdt, dltz),
over 20 Pymoo	Yes	Python	MO/SO	Benchmark, Global Optimization, over 20
PyMetaheuristic	No	Python	SO	Test Functions, User-defined, over 20
iMOPSE	Yes	C++, adds in Python)	MO/SO	Scheduling/Combinatorial MS-RCPSP, cVRP, TSP, TTP, 5 in total

The **main motivation of the iMOPSE library** is to provide a tool versatile and accessible enough to adapt to various optimization scenarios, particularly focusing on NP-hard problems with constraints. It aims to provide a comprehensive toolkit for researchers and practitioners, enabling efficient exploration of different optimization methods and scenarios.

2.1 iMOPSE in Comparison

iMOPSE focuses itself on the competitive landscape of optimization libraries by specifically tailoring its features to handle complex NP-hard problems, which frequently have direct real-world applications. Unlike many other libraries that primarily focus on problems operating within continuous domains, iMOPSE is capable of working with binary, continuous, and permutational-based problems. This flexibility is largely due to its versatile **encoding** system, adept at managing diverse optimization challenges, and its modular architecture that allows integration of **state-of-the-art algorithms** with various problem types through the use of specialized operators. These operators are crucial for customizing the library's approach to unique problem sets, enhancing its utility across varied

optimization scenarios. This robust architectural foundation allows iMOPSE to adapt effectively across a broad spectrum of optimization tasks, positioning it as a standout choice for those needing advanced problem-solving capabilities.

Additionally, the architecture of iMOPSE features **extensibility** as well as **utilities** such as archive utils, clustering utils, and experiment logger utils, which simplify the management and analysis of optimization experiments. These tools enhance the library's functionality and user-friendliness, making it an ideal choice for both novice and experienced users in the field of optimization.

Furthermore, iMOPSE supports this technical prowess with practical tools and resources that other libraries often lack. It includes **ready-to-use instances** for included problems, comprehensive tutorials, and straightforward usage through parameters and configuration files, making it user-friendly even for those new to optimization. Additional **tools** for data collection, analysis, validation, and visualization are incorporated, providing a holistic approach to optimization projects.

The choice of **C++17** as the foundation ensures that iMOPSE benefits from fast calculations and rapid prototyping capabilities, distinguishing it from Python-based libraries in terms of performance efficiency. This makes iMOPSE particularly appealing for projects where speed and performance are critical.

In summary, iMOPSE offers a complete set of tools that aligns well with the capabilities of other leading optimization libraries, while placing a particular emphasis on specific aspects of optimization, tailoring its features to meet unique challenges within this field. Featuring a combination of advanced algorithms, specialized operators, and a design that supports users, it serves as a reliable resource for both academic research and practical implementations in the field of metaheuristic optimization. In the following sections, we will describe iMOPSE's features in more detail to present its capabilities and further support our claims.

3 iMOPSE the Main Idea and Architecture

iMOPSE comprises two primary C++ projects: the main optimizer and paretoAnalyzer. In this section, we'll delve into the main optimizer's architecture to provide a clearer understanding of iMOPSE's capabilities.

The architecture of the optimizer is designed with a focus on modularity, comprising two primary modules: the method module and the problem module, along with additional utilities (see Fig. 1). The method module contains an Operators submodule. This structure is delineated by interfaces that ensure flexibility and modularity. The utilization of the *Problem* interface of the most generic *Method* interface allows to incorporate other optimization algorithms.

Method Module: This module is the core of iMOPSE's functionality, housing the various optimization algorithms initialized from simple configuration files. It is designed to be extensible, allowing for the easy addition of new methods or enhancement of existing ones. The *Operators* submodule within this module includes a range of operators that manage neighborhoods, mutations, and

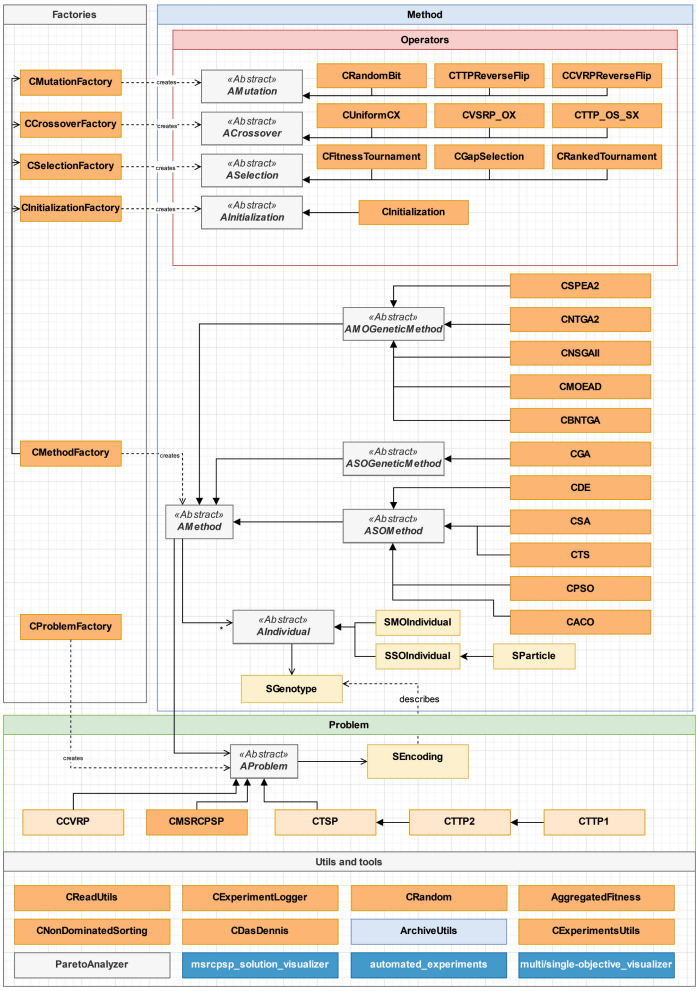


Fig. 1. The iMOPSE general schema.

crossovers, which are integral to the functioning of different optimization methods.

Problem Module: Contains the implementation of different optimization problems loaded from problem instance files. Each problem in iMOPSE adheres to a *Problem* interface, ensuring that it can be seamlessly integrated with the methods module. This setup allows for a wide variety of problems, from simple to NP-hard, to be addressed using the toolkit. The *Problem* interface includes two key methods: *Evaluate* and an *Encoding* getter. These methods enable the creation and manipulation of individuals using various operators, as well as their subsequent evaluation. This functionality is essential for effective problem-solving in the framework.

Each method and problem instance is created in the corresponding factory class, thus facilitating efficient organization and management of resources. The factory classes are responsible for the acquisition and management of the necessary components, ensuring that each part of the system is appropriately initialized and configured.

3.1 Implemented Solving Methods and Operators

To make the iMOPSE library useful for researchers, we implemented several state-of-the-art metaheuristics for single- and multi-objective optimization as well as specialized operators, which will be described in this section.

In the basic version of iMOPSE, several metaheuristics for **single-objective optimization** are implemented, such as trajectory-based SA and TS, swarm-based Ant Colony Optimization and PSO but also evolutionary population-based methods GA and DE. Moreover, DE is implemented as a hybrid DEGR [18] with a greedy heuristic applied to the Multi-Skill Resource-Constrained Project Scheduling Problem (MS-RCPSP).

In the iMOPSE library **multi-objective optimization** evolutionary-based state-of-the-art methods are implemented, such as NSGA-II [14], MOEA/D [21], SPEA2 [22] and Non-Dominated Tournament Genetic Algorithm – NTGA2 [19]. These methods are known as effective in solving multi-objective NP-hard problems with constraints, such as the Traveling Thief Problem and MS-RCPSP [19]. Additionally, in the iMOPSE library, some experimental methods have been included, such as balanced B-NTGA [8] and gene-level adaptation aB-NTGA [9].

Some of the implemented metaheuristics utilize **specialized operators** to exploit and explore the solution landscape effectively for various problems. In iMOPSE, there are implemented neighborhood operators (for TS or SA), mutations (like *RandomBit*, *ReverseFlip*, *GaussMutation*), or crossovers (e.g., *Ordering Crossover OX*, *Cycle Crossover CX*).

To direct metaheuristic in a global search, especially for evolutionary computation, **selection** operators are needed – *random* (semi-blind, without selection pressure, as reference), as well as classic *tournament* or *gapSelection* [19] for multi-objective optimization. Moreover, a predefined set of representations, operators, and selections could be easily extended as the iMOPSE library C++ interfaces are given for implementation.

3.2 Implemented Problems

In this section, we explore the problems implemented in the iMOPSE, detailing their specific representations and encodings, which are crucial for effective problem-solving. For metaheuristics, a specialized **representation** (or solution format) is essential to enable an effective search within the solution space. In iMOPSE, three primary types of representations are utilized: permutation, binary, as well as integer- and real-coded, accommodating the unique requirements of various NP-hard problems. All listed problems are recognized for their NP-hard nature.

The **Travelling Salesman Problem (TSP)** is a classic optimization challenge, the goal is to determine the shortest possible route that visits each city in a given list exactly once. The problem adheres to constraints, chiefly ensuring that no city is visited more than once. It is tackled as a single-objective optimization task, where the measure of success is the tour's length – shorter tours signify better fitness. The TSP uses a permutational solution representation, where the solution is expressed as a sequence, or permutation, of indexes of cities to visit. iMOPSE contains a popular *TSPLIB* instance set ready for experimentation.

The **Traveling Thief Problem (TTP)** [13] combines TSP with the Knapsack problem, making it more complex and reflecting real-world problems. It can be either single-objective, focusing on maximizing the profit of the collected items minus the traveling cost multiplied by the renting ratio (TTP1) [13], or multi-objective, considering both the profit and the total weight of the knapsack (TTP2) [13]. Users can apply the single-objective method in multi-objective problems by providing objective weights for fitness aggregation. TTP is characterized by a mixed representation; it combines a permutational segment for the order in which cities are visited and a binary segment to handle decisions about which items to pick up in each city. This hybrid encoding effectively captures the dual nature of routing and packing in TTP. iMOPSE is equipped with basic benchmark TTP instances of TTP (e.g. *berlin52*, *eil51*).

The **Multi-Skill Resource-Constrained Project Scheduling Problem (MS-RCPSP)**)[17] is a fundamental scheduling challenge in project management, aimed at completing a set of projects in the shortest possible time while adhering to resource limitations and project dependencies. The MS-RCPSP extends the classic RCPSP by allowing activities to be performed by resources equipped with specific skills, thus close to real-world project scheduling challenges and increasing the complexity of the problem. The constraints include the availability of resources and precedence relations between activities. Objectives encompass total project duration, cost, average cash flow, skill overuse, and resource utilization. The MS-RCPSP can be considered as a single-objective optimization problem (e.g., using a weighted sum of cost and duration of the final schedule) [18], a multi-objective opt. (e.g., balancing duration versus cost), or a many-objective opt. with five objectives [8,19]. The MS-RCPSP in the iMOPSE library is implemented twice using two different encodings: permutational or real-coded. These encodings represent the diverse approaches utilized to manage the complexities of the MS-RCPSP, complemented by the use of a greedy schedule builder specifically designed for this problem. For more detailed information, we direct readers to the iMOPSE documentation [2]. This flexibility in encoding effectively accommodates the intricacies of scheduling tasks while considering multiple skills and resource constraints. iMOPSE provides a diverse set of 265 instances for the MS-RCPSP, generated using the iMOPSE generator with varying parameters to showcase constraint influences. These instances are categorized into groups offering different complexities and characteristics.

The **Capacitated Vehicle Routing Problem (cVRP)** [1] is a complex optimization challenge that involves planning routes for a fleet of vehicles to

deliver goods or services to a set of customers, starting and ending at a central depot. Each customer has a specific demand that must be met without exceeding the carrying capacity of any vehicle. The primary goal of the cVRP is to minimize the total distance traveled by all vehicles while ensuring that each customer's demand is fulfilled and each customer is visited exactly once. In addition to the basic constraints of vehicle capacity and mandatory single visits per customer, the cVRP often incorporates other operational challenges such as time windows within which the customers must be served, varied vehicle fuel capacities, and sometimes even different vehicle types, each with its own cost and capacity implications. Currently, the fitness of a solution is evaluated based on the total distance of all routes; however, while our current model focuses solely on distance, it will be extended in the future to include additional objectives. These may include optimizing for factors such as fuel consumption, driver hours, and vehicle wear and tear, which would not only reduce operational costs but also address environmental concerns and regulatory compliance. The cVRP employs multiple permutational sections in its representation of each vehicle's route. This allows for the optimization of multiple routes simultaneously, ensuring that each customer's demand is met without exceeding vehicle capacities. For cVRP iMOPSE offers instances from *CVRPLIB* [1] instance set.

Moreover, iMOPSE's modular architecture enables easy **extension** by other optimization problems, e.g., the Job Shop Scheduling Problem (JSP). With its flexible design and support for diverse problem domains, iMOPSE can seamlessly integrate additional problems, enhancing its applicability across various optimization challenges.

4 Additional iMOPSE Tools

Apart from various utility classes that are described in detail in documentation [2], iMOPSE offers additional tools for visualization, validation, and analysis. In this section, we highlight incorporated tools.

4.1 iMOPSE Input Parameters and Configurations

The iMOPSE executable is tailored for the efficient handling of optimization problems, requiring specific input parameters for its execution. The following outlines the necessary inputs and additional capabilities:

- **Parameters:** Parameters for running the iMOPSE executable:
  1. **Method Configuration Path:** A path pointing to the method configuration file. Each method defines data needed for configuration, although they often share similarities like population size.
  2. **Problem Name:** The name of the optimization problem being addressed.
  3. **Problem Instance Path:** The path to the problem instance file.
  4. **Output Path:** The directory path where the output, including logs and results, will be stored.

5. **Number of Runs (Optional, default=1):** Users can specify the number of runs for the given configuration.
6. **Seed(Optional, default=0):** Users can set a seed value, thus enabling the reproduction of experiments. If the number of runs is greater than one, the first run starts with the provided seed, and each subsequent seed is incremented by one per run.

This structure ensures comprehensive input management and flexible execution, accommodating various optimization needs and scenarios. The more detailed instructions are accessible from the iMOPSE documentation.

4.2 Pareto Analyzer

iMOPSE includes a subproject named *ParetoAnalyzer*, which plays a crucial role in the analysis and comparison of multiple multi-objective optimization results across examined methods. This tool is designed to be used after conducting experiments and is thus not included in the optimizer project. The main features of the discussed tool include:

- **Calculation of True Pareto Front:** In multi- and many-objective optimization, the output of each method is a set of non-dominated points. The True Pareto Front (TPF) could be defined as a set of all non-dominated solutions and can be considered the best available Pareto Front Approximation (PFA). However, in practical real-world problems, TPF is usually unknown. The Pareto Analyzer tool calculates this TPF using results generated by all runs of all compared methods.
- **Nadir Point Calculation:** The tool determines the Nadir Point, which is a point with the worst possible values for all objectives.
- **Pareto Visualization:** With the use of a Python script for the visualization part, it offers capabilities to visualize specified PFA in comparison, providing a clear and intuitive understanding of the optimization outcomes (see Fig. 2).
- **Quality Measures for MO:** The tool is equipped with code for the calculation of various metrics that are essential for evaluating the quality of the PFA, including $InvertedGenerationalDistance$ (IGD), $HyperVolume$ (HV), Pareto Front Size (PFS), and $Purity$. These measures assess convergence, diversity, volume coverage, and the proportion of non-dominated solutions in the PFA, respectively.

In summary, *ParetoAnalyzer* complements the iMOPSE by providing tools for the analysis and evaluation of multi-objective optimization results. It facilitates visualization and quantification of the effectiveness of optimization algorithms, aiding researchers and practitioners in making informed decisions.

4.3 Python Scripts

Python scripts in iMOPSE are designed to enhance scientific research. While designed as ready-to-use tools for common tasks, they offer users the flexibility to analyze their results in a preferred way, assisting with common tasks.

The **msrcpsp_solution_visualizer** tool validates and visualizes MS-RCPSP solutions, aiding comprehension and refinement of optimization methods. For multi-objective optimization researchers, **multi-objective_visualizer** elucidates trade-offs between competing objectives, while **single-objective_visualizer** offers a graphical overview of fitness values. The **automated_experiments** script streamlines concurrent execution of iMOPSE, providing a simple way to run multiple different experiments.

These scripts enhance the efficiency and insightfulness of experiments, enriching the quality and depth of research within iMOPSE. Planned integration with the main C++ codebase will further bolster the software's robustness and scalability for broader scientific applications.

5 A Case Study: Conducting Experiments with iMOPSE

In this case study, we utilize the iMOPSE library for a comparative analysis between two multi-objective evolutionary algorithms: NTGA2 and MOEA/D. The investigation focused on the "eil51_n150_uncorr_01" instance from the eil52 instance set of the TTP. The **experimental configurations (setup)** for both the MOEAD and B-NTGA methods were set up using the configuration files presented in Listings 1.1, 1.2, and 1.3:

```
../../configurations/methods/MOEAD/MOEAD_TTP.cfg
TTP2
../../configurations/problems/TTP/eil51/eil51_n150_uncorr_01.ttp
../experiments/MOEAD/eil51_n150_uncorr_01/
10
0
```

Listing 1.1. Configuration for MOEAD

```
../../configurations/methods/BNTGA/BNTGA_TTP.cfg
TTP2
../../configurations/problems/TTP/eil51/eil51_n150_uncorr_01.ttp
../experiments/BNTGA/eil51_n150_uncorr_01/
10
0
```

Listing 1.2. Configuration for BNTGA

```
MethodName BNTGA
GenerationLimit 1000
Crossover TTP_OX_SX 0.7, 0.7
Mutation TTP_Reverse_Flip 0.5 0.5
PopulationSize 50
GapSelection 40
```

Listing 1.3. Selected B-NTGA Configuration File

Each algorithm was applied to the defined MS-RCPSP instance for ten repeated experiments, initialized with a seed value set to 0. The results were captured as PFA, illustrating the trade-offs between the objectives.

For **Result Analysis**, the iMOPSE framework saved the PFAs after each experiment and run. Subsequently, paretoAnalyzer was used to calculate the 'True' PFA and the associated metrics. The configurations for this process are illustrated in Listings 1.4 and 1.5:

```
../../optimizer/experiments/MOEAD
../../optimizer/experiments/BNTGA
```

Listing 1.4. Pareto Analyzer Configuration File (config.cfg)

```
../config.cfg
eil51_n150_uncorr_01
../
```

Listing 1.5. Pareto Analyzer Parameters

Pareto Analyzer saved merged PFAs for each method as well as the 'True' PFA. These fronts can be visualized using the **multi-objective visualizer** (see Listing. 1.6) script by providing the paths to the generated merged PFA files. The relevant Python code snippet for this task is shown below.

The PFAs results compare the performance of MOEA/D and B-NTGA algorithms for the "eil51_n150_uncorr_01" instance: B-NTGA ($HV = 0.85655$ and $IGD = 0.00062$), MOEA/D ($HV = 0.67808$ and $IGD = 0.00943$). B-NTGA achieves higher HV and lower IGD than MOEA/D, indicating superior PFA quality. Overall, these findings suggest that B-NTGA outperforms MOEA/D in terms of PFA quality in this instance.

```
paths = ['../paretoAnalyzer/MOEAD_merged.csv',
         '../paretoAnalyzer/BNTGA_merged.csv']
```

Listing 1.6. Python Code for Visualizing Pareto Fronts

Figure 2 presents the **visual comparison** of merged PFA for MOEA/D and B-NTGA. The B-NTGA algorithm, represented in orange, demonstrates a broader spread across the objective space, indicating a diverse set of solutions. Conversely, the MOEA/D, shown in blue, presents a more concentrated clustering of solutions. Overall, B-NTGA appears to have an advantage over MOEA/D when considering the diversity of the solutions and the extent of the coverage of the PFA.

6 Summary and Future Works

The iMOPSE library presented in this paper is implemented as an open-source C++ programming framework to support the metaheuristic research community – for researchers, students, and practitioners. The main functionalities of

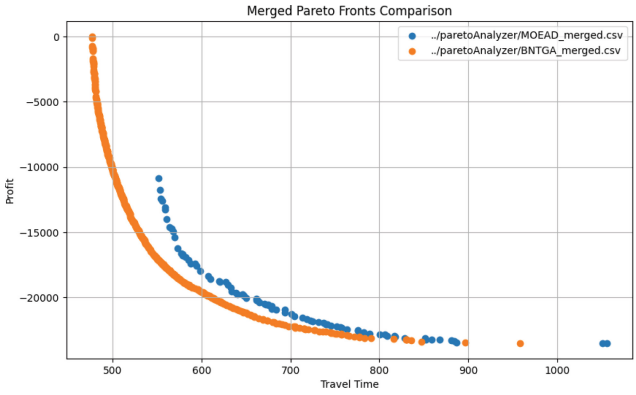


Fig. 2. Comparison of PFAs for MOEAD and B-NTGA [TTP eil51_n150_uncorr_01]

the iMOPSE library are connected to reproducible research with metaheuristics applied to solve single and multi-objective NP-hard problems.

The first version of the iMOPSE library includes commonly used metaheuristics, implementations of classical NP-hard problems, and additional utilities useful in research data analysis. Moreover, other programmers could easily extend the library – new methods and problems can be easily added. Together with various smaller improvements, we plan to incorporate other metaheuristics, like Bee Colony Optimisation or Genetic Programming, that are effective in NP-hard problem optimization. Also, the set of MOEA methods could be extended by θ-DEA [20] or HyPE [10]. To support metaheuristics in effective global searching, some local search or heuristics could be added – e.g. specific for the cVRP problem – to make the method more effective and/or build hybrid or hyper-heuristic approaches.

In the next version of iMOPSE **other optimization problems** could be included (e.q., JSP or Quadratic Assignment Problem) as well as continuous problems (e.g., to solve CEC competitions). In the first version of iMOPSE natural metaheuristic parallelism is not used – we also want to include it in the next version. Moreover, not only a multi-thread computation but also a GPU-based computation of metaheuristics should be considered to speed up computations.

Last but not least, a promising direction of iMOPSE is the **utils module extension**, where we plan to implement extra tools to support reproducible research, such as Taguchi's Design of Experiments procedure, Grey relational analysis, and various statistical tests. Finally, we plan to support solving many-objective optimization problems (i.e. 3+ obj., like MS-RCPSP [8]) by effectively calculating quality measures [15] and visualization methods for PFA with 3+ objectives.

References

1. cvrp instances. http://vrp.galgos.inf.puc-rio.br/
2. imopse library [github]. https://github.com/imopse/iMOPSE
3. jmetal. https://jmetal.readthedocs.io/
4. Metaheuristics.jl. https://github.com/jmejia8/Metaheuristics.jl
5. Platypus. https://github.com/Project-Platypus/Platypus
6. pygmo. https://esa.github.io/pygmo/
7. pymetaheuristic. https://pypi.org/project/pymetaheuristics/
8. Antkiewicz, M., Myszkowski, P.B.: Balancing pareto front exploration of non-dominated tournament genetic algorithm (b-ntga) in solving multi-objective np-hard problems with constraints. Inf. Sci. **667**, 102400 (2024)
9. Antkiewicz, M., Myszkowski, P.B., Gmyrek, K., Olech,Ł.P.: Gene-level adaptation in balanced non-dominated tournament genetic algorithm (ab-ntga) applied to versatile multi-stage weapon-target assignment problem. In: Genetic and Evolutionary Computation Conference, GECCO 2024 (2024)
10. Bader, J., Zitzler, E.: Hype: an algorithm for fast hypervolume-based many-objective optimization. Evol. Comput. **19**(1), 45–76 (2011)
11. Belli, F., Tuglular, T., Ufuktepe, E.: Unifying behavioral and feature modeling for testing of software product lines. Int. J. Softw. Eng. Knowl. Eng. (2023)
12. Blank, J., Deb, K.: pymoo: multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)
13. Bonyadi, M.R., Michalewicz, Z., Barone, L.: The travelling thief problem: the first step in the transition from theoretical problems to realistic problems. In: 2013 IEEE Congress on Evolutionary Computation, pp. 1037–1044 (2013)
14. 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)
15. Laszczyk, M., Myszkowski, P.B.: Survey of quality measures for multi-objective optimization: construction of complementary set of multi-objective quality measures. Swarm Evol. Comput. **48**, 109–133 (2019)
16. Lukasiewycz, M., Glaß, M., Reimann, F., Teich, J.: Opt4J - a modular framework for meta-heuristic optimization. In: Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011), Dublin, Ireland, pp. 1723–1730 (2011)
17. Myszkowski, P.B., Laszczyk, M.: Investigation of benchmark dataset for many-objective multi-skill resource constrained project scheduling problem. Appl. Soft Comput. **127**, 109253 (2022)
18. Myszkowski, P.B., Olech, ŁP., Laszczyk, M., Skowroński, M.E.: Hybrid differential evolution and greedy algorithm (degr) for solving multi-skill resource-constrained project scheduling problem. Appl. Soft Comput. **62**, 1–14 (2018)
19. Myszkowski, P., Laszczyk, M.: Diversity based selection for many-objective evolutionary optimisation problems with constraints. Inf. Sci. **546**, 665–700 (2021)
20. Yuan, Y., Xu, H., Wang, B., Yao, X.: A new dominance relation-based evolutionary algorithm for many-objective optimization. IEEE Trans. Evol. Comput. **20**(1), 16–37 (2016)
21. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007)
22. Zitzler, E., Laumanns, M., Thiele, L.: Spea2: Improving the strength pareto evolutionary algorithm. ETH Zurich, Computer Eng. Netw. Lab. **103** (2001)

Understanding the Importance of Evolutionary Search in Automated Heuristic Design with Large Language Models

Rui Zhang[1], Fei Liu[1], Xi Lin[1], Zhenkun Wang[2], Zhichao Lu[1(✉)], and Qingfu Zhang[1(✉)]

[1] Department of Computer Science, City University of Hong Kong, Hong Kong, China
{fliu36-c,xi.lin}@my.cityu.edu.hk, luzhichaocn@gmail.com, qingfu.zhang@cityu.edu.hk

[2] School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, China
wangzk3@sustech.edu.cn

Abstract. Automated heuristic design (AHD) has gained considerable attention for its potential to automate the development of effective heuristics. The recent advent of large language models (LLMs) has paved a new avenue for AHD, with initial efforts focusing on framing AHD as an evolutionary program search (EPS) problem. However, inconsistent benchmark settings, inadequate baselines, and a lack of detailed component analysis have left the necessity of integrating LLMs with search strategies and the true progress achieved by existing LLM-based EPS methods to be inadequately justified. This work seeks to fulfill these research queries by conducting a large-scale benchmark comprising four LLM-based EPS methods and four AHD problems across nine LLMs and five independent runs. Our extensive experiments yield meaningful insights, providing empirical grounding for the importance of evolutionary search in LLM-based AHD approaches, while also contributing to the advancement of future EPS algorithmic development. To foster accessibility and reproducibility, we have fully open-sourced our benchmark and corresponding results.

Keywords: Automated heuristic design · evolutionary program search · large language model · evolutionary computation

1 Introduction

Automated heuristic design (AHD) aims to automatically select, refine, or construct effective heuristics, thereby obviating the necessity for rich domain expertise traditionally required in manual heuristic design [1–3]. Considerable effort has been dedicated to employing machine learning techniques for AHD [4–6]. Among them, genetic programming (GP) [7] is one of the most widely used

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 185–202, 2024.
https://doi.org/10.1007/978-3-031-70068-2_12

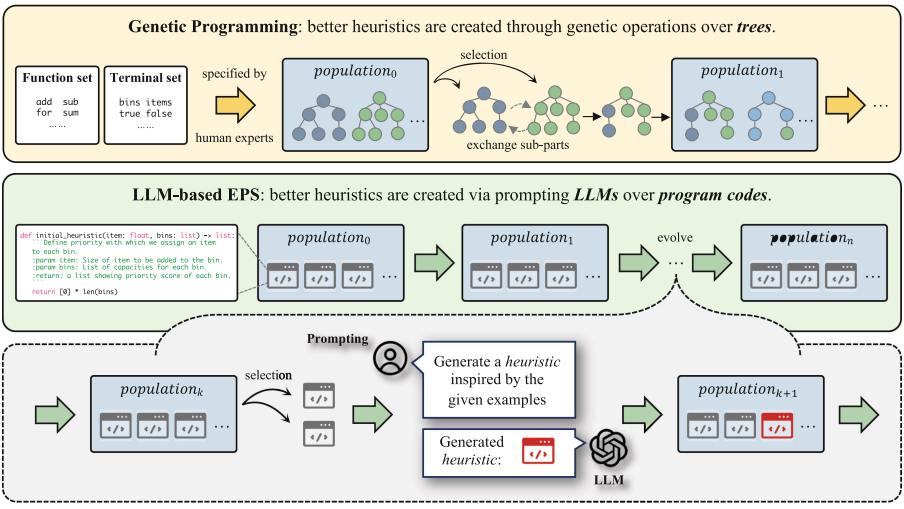


Fig. 1. An illustration of the LLM-based EPS paradigm, with respect to the GP-based paradigm (top section), for automated heuristic design.

techniques for handling AHD tasks, owing to its flexible representation and efficacy across various domains [3,8–10]. However, GP necessitates specifying a set of permissible primitives and mutation operations, which unfortunately are non-trivial and problem-dependent [11].

Recently, the advent of large language models (LLMs) has introduced novel tools for AHD. Preliminary endeavors have been made to model AHD as a program search problem, employing LLMs to aid the solution (i.e., heuristics) generation and optimization process within an evolutionary framework. For instance, FunSearch [12] evolves heuristics for mathematical problems that outperform existing solutions on cap set and admissible set problems [13]. EoH [14] and ReEvo [15] evolve heuristics for combinatorial optimization (CO) problems and consequently outperform existing AHD methods on traveling salesman problems (TSPs) [16] and online bin packing (OBP) problems [17].

These methodologies essentially adopt a canonical paradigm, referred to as *LLM-based Evolutionary Program Search (EPS)* in this work, that comprises the following key aspects: (*i*) candidate solutions (i.e., heuristics) are represented as executable computer *programs*, also referred to as codes; (*ii*) an evolutionary computation (EC) paradigm is used to evolve toward better programs; and (*iii*) LLMs are used as the main engine for driving the search, i.e., creating new programs, introducing variations to existing programs, etc. A pictorial illustration of this paradigm is provided in Fig. 1.

Despite a steady stream of promising empirical results, we have noticed three issues: (*i*) Inconsistent benchmark settings, where existing LLM-based EPS methods exhibit variations in initialization, termination criteria, and choice of LLMs; (*ii*) Inadequate baselines, where existing LLM-based EPS methods were

primarily evaluated against random search or simple heuristics derived through human intuitions; (*iii*) Lack of detailed analysis on the relative contribution of each component (e.g., choice of LLMs, prompt and search strategies, etc.) to the overall success achieved by existing LLM-based EPS methods.

To address these issues, we first develop a simple yet effective EPS baseline, dubbed (1+1)-EPS, taking inspiration from (1+1)-ES [18] and few-shot prompting [19]; and we design a unified benchmark setup comprising four LLM-based EPS methods (three existing methods and our baseline), four AHD problems, and nine different LLMs. Then, curated experiments are designed around the following two research queries: ① the necessity of coupling LLMs with search strategies for AHD, and ② the current progress made by existing LLM-based EPS methods on AHD. Detailed analyses are subsequently carried out for new knowledge and insights.

Key Takeaways: Through extensive experiments, we find that:

- The inherent generative capability of LLMs alone is insufficient, providing an empirical justification for coupling LLMs with a search mechanism, i.e., the LLM-based EPS paradigm, for tackling AHD problems effectively (Sect. 4.1).
- The performance of existing LLM-based EPS methods varies significantly across different AHD problems and LLM choices, suggesting more diverse benchmarks and applications are needed to establish a better understanding of this emergent paradigm for AHD (Sect. 4.2).

We summarize the primary contributions of this work as below:

1. **Large-scale benchmark.** This work examines all existing LLM-based EPS methods along with the proposed baseline on four AHD problems under a canonical benchmark setting. Each compared method is evaluated over nine LLM choices and five independent runs.
2. **Insights and implications for future research.** With extensive results, we provide the empirical grounding for the necessity of LLM-based EPS for AHD and suggestions for future EPS algorithmic development.
3. **Fair and reproducible evaluation.** We open-source the implementations of all compared LLM-based EPS methods, AHD problems, and interface to both open- and closed-source LLMs at https://github.com/zhichao-lu/llm-eps to foster future development.

2 Background

Automated Heuristic Design (AHD) is also known as hyper-heuristics [1, 2, 20], aiming to search over a space of heuristics rather than the solutions to a specific problem directly. Most of the AHD approaches incorporate a learning mechanism [4, 21], such as reinforcement learning [5], Bayesian learning [6], case-based reasoning [22], and evolutionary computation methods [23–26].

In particular, genetic programming (GP) [7] has emerged as a promising approach to automate the design of heuristics. In essence, GP maintains a set

of computer programs in the form of trees, instructions, graphs, etc., where better programs are evolved through genetic operations, such as crossover and mutation. GP-based AHD approaches have been applied in a number of different application domains, such as combinatorial optimization [27–29], scheduling [3,9], among other areas [30,31]. Although GP-based AHD approaches have achieved promising results, they are often criticized for the need to explicitly specify the function sets and primitive sets, which are not trivial and problem-dependent [11]. A more in-depth discussion of the connection between GP and the methods studied in this work is provided in Appx. §A.

Large Language Models (LLMs) typically refer to deep neural networks with billions or even trillions of model parameters, built upon the Transformer architecture [32]. The input query to LLMs can be any sequence of texts, such as natural language, codes, mathematical expressions, etc. As output, the LLM also provides a sequence of texts in response to the input query.

With the exponential growth in model size and training data, LLMs have improved at an impressive pace in the recent past [19,33], leading to groundbreaking performance across a wide range of tasks [34–37]. Notably, the synergy between LLMs and evolutionary computation (EC) has been successfully applied to solve various optimization problems, such as prompt optimization [38,39], algorithm design [40–42], and neural architecture search [43], to name a few. Through the lens of EC, LLMs can be viewed as an intelligent variation operator [44], yielding more diverse and novel offspring compared to conventional means, such as genetic operators, differential evolution, or particle swarm [45,46]. This has in turn translated to promising results in various domains [47–49].

Table 1. Existing LLM-based EPS methods for AHD along with the baseline, $(1+1)$-EPS, proposed in this work.

Method	Prompt Strategy	LLM	Search Strategy
FunSearch [12]	Few-shot prompting [19]	Codey [50], StarCoder [51]	Island model with re-starts
EoH [14]	CoT [52]	GPT-3.5 [19], Gemini Pro, DeepSeek [53], CodeLlama [54]	GA
ReEvo [15]	CoT [52] + Reflection [19]	2×GPT-3.5 [19]	GA
$(1+1)$-EPS (our baseline)	One-shot prompting	LLMs in Table 2	$(1+1)$-ES

Existing LLM-based EPS Methods exhibit variations mainly in the following three aspects: (i) search strategy, (ii) prompt strategy, and (iii) choice of LLMs. An overview comparison is provided in Table 1. Readers are referred to Appx. §A for elaborated descriptions.

From the perspective of search strategy, many existing EPS methods [14,15] adopt the standard genetic algorithm (GA) framework, where a population of randomly initialized heuristics is made gradually better through genetic operators (i.e., crossover and mutation) and elitist selection [55]. Sophisticated modifications (to the standard GA framework) have also been tried, in particular,

FunSearch introduces an island model (in the form of multiple distinct populations) with a re-start mechanism to promote diversity among individuals [12].

From the perspective of prompt strategy, FunSearch [12] adopts a simple strategy, i.e., few-shot prompting where the LLM outputs are conditioned on a few provided examples of heuristics [19]. More sophisticated strategies, typically variants of the chain of thought prompting (CoT) [52], have been adopted in the subsequent works. In particular, EoH [14] leverages linguistic descriptions of the heuristics (referred to as thoughts) and develops five different prompt strategies to balance the exploration and exploitation aspects of the evolutionary search; ReEvo [15] applies the reflection technique [56] to verbalize trends from past high-performant individuals into prompts.

From the choice of LLM perspective, most existing EPS methods are solely evaluated with closed-source LLMs (e.g., GPT-3.5 [19] and Codey [50]) except FunSearch which is also evaluated with an open-source LLM, i.e., StarCoder [51]. In addition, ReEvo [15] uses two LLMs – one for generating prompts and the other one for generating heuristics.

3 Preliminaries

In this section, we describe the experimental setup in terms of benchmark problems, baselines, and choices of LLMs, among other settings.

Benchmark Problems. We consider three types of applications for AHD.

① Admissible Set (AS) [13] is a variation of the cap set problem from mathematics [57]. Formally, admissible set problems, denoted as $\mathcal{A}(n, w)$, are collections of vectors in $\{0, 1, 2\}^n$ that satisfy: (1) Each vector has the same number w of non-zero elements but a unique support. (2) For any three distinct vectors there is a coordinate in which their three respective values are $\{0, 1, 2\}$, $\{0, 0, 1\}$, or $\{0, 0, 2\}$. The objective of the admissible set problem is to maximize the size of the set while fulfilling all the aforementioned criteria. In this work, we set $n = 15$ and $w = 10$, i.e., $\mathcal{A}(15, 10)$, to be consistent with prior works [12].

② Online Bin Packing (OBP). The objective of bin packing problems is to allocate a collection of items with varying sizes into the fewest possible bins of fixed capacity of C. We consider the online scenario where items are packed as they arrive, in contrast to the offline scenario where all items are known beforehand. In this work, we consider two widely used datasets for OBP: the OR dataset [58] and the Weibull dataset [59]. To guide various LLM-based EPS methods in designing heuristics, we use 20 instances where each comprises 250 items with sizes sampled from $[20, 100]$ for the OR dataset [12]; and we use five instances where each comprises 5K items with sizes sampled from a Weibull distribution of $f(45, 3)$ for the Weibull dataset [12,60]. The capacity C of each bin is set to 150 and 100 for OR and Weibull datasets, respectively.

③ Traveling Salesman Problem (TSP) aims to find the shortest route to visit all the given locations once and return to the starting location [16]. It is considered one of the most important CO problems and a widely used test bed

for heuristic design approaches. We use a set of 64 TSP100 instances [61] where the coordinates of locations to be visited are randomly sampled from $[0, 1]$ to guide the compared LLM-based EPS methods in designing heuristics [15,60].

Baseline. An adequate baseline is essential for understanding the relative improvements made by the various methods (at least empirically). Existing LLM-based EPS methods were mostly compared against random search (i.e., uniform sampling) or simple heuristics[1] based on human intuitions, yielding promising performance across diverse problems. However, we argue that a *more adequate baseline* beyond random search and intuitive heuristics is needed for a meaningful and representative comparison. To this end, inspired by the (1+1)-ES [18], we develop a simple EPS baseline, dubbed *(1+1)-EPS*. The pseudocode of the proposed baseline is provided in Algorithm 1. Given its simplistic design in both the search and prompt strategies, we envision that $(1 + 1)$-EPS should simulate the lower bound of the performance of the EPS paradigm.

Algorithm 1: $(1 + 1)$-EPS

Input : f_{LLM}: a LLM,
h_{T}: a template heuristic,
T: max. # of gens.

$h_{\text{best}} \leftarrow h_{\text{T}}$ // initialize the best heuristic (found so far) to h_{T}.
$s_{\text{best}} \leftarrow evaluate(h_{\text{best}})$ // evaluate the performance score of h_{best}.
while $t < T$ **do**
  // create input prompts for f_{LLM} by converting h to prompts via one-shot prompt engineering.
  $h \leftarrow f_{\text{LLM}}(prompts)$ // create a heuristic via a LLM.
  $s \leftarrow evaluate(h)$ // evaluate the performance of h.
  if $s < s_{\text{best}}$ **then**
    // update the best heuristic found so far and its score.
    $h_{\text{best}} \leftarrow h$, $s_{\text{best}} \leftarrow s$
  end
end
Return h_{best}, s_{best}

One-shot Prompting

Idea: Create input prompts (h) by providing the best heuristic (h_{best}) found so far as an example.

E.g.: The shaded texts below are prompts created for an online bin packing problem.

```
def h_best(
  item: float,
  bins: list) -> list:
  priority = </> #
    omitted for
    brevity
  return priority

def h(
  item: float,
  bins: list) -> list:
  priority = <...to be
     filled by a LLM
    ...>
  return priority
```

Choice of LLMs. We consider a diverse set of LLMs to investigate the impact of the choice of LLMs on the AHD performance. We include five open-source

[1] For instance, an intuitive heuristic for an OBP problem could be "place the item in the first bin with available capacity remaining".

Table 2. Overview of the LLMs evaluated in this work. We use performance on "HumanEval" [62] and "MMLU" [63] to indicate the capabilities of LLMs on code and general knowledge, respectively. Both metrics are greater the better.

Model	#P	Specialized for Code	Open Source	HumanEval (↑) [62]	MMLU (↑) [63]
UniXcoder [64]	0.3B	✗	✓	–	–
StarCoder [51]	15.5B	✓	✓	33.6%	–
CodeLlama [54]	7B	✓	✓	34.8%	42.1%
	34B	✓	✓	48.8%	53.1%
DeepSeek-Coder [53]	6.7B	✓	✓	66.1%	34.6%
	33B	✓	✓	69.2%	39.5%
GPT-3.5 [19]	–	✗	✗	60.3%	70.0%
GPT-4 [33]	1.76T	✗	✗	76.5%	86.4%
Claude 3 Opus [65]	137B	✗	✗	84.9%	86.8%

LLMs that were fine-tuned on code-related tasks and three closed-source LLMs developed for general purposes. In particular, we consider the most powerful LLM currently available, i.e., Claude 3 Opus [65], and the most capable open-source LLM for coding tasks, i.e., DeepSeek-Coder [53]. For completeness, we also include the most powerful coding language model prior to the LLM era, i.e., UniXcoder [64]. Table 2 provides an overview of the considered LLMs. For open-source LLMs, we deploy these models locally on a server with 16 NVIDIA V100 GPU cards; while for closed-source LLMs, we rely on the respective APIs provided by OpenAI and Anthropic to get responses.

Evaluation Metric. To evaluate the AHD performance, we report the mean relative distance (or gap) in performance, Δ_{d}, of the obtained heuristic with respect to the performance of the best-known performance, mathematically as follows.

$$\Delta_{\mathrm{d}} = 100\% \times \frac{1}{N_p} \sum_{p=1}^{N_p} \frac{1}{N_m} \sum_{m=1}^{N_m} \frac{(-1)^{I_p}(M_{p,m} - M_p^*)}{M_p^*}$$

where N_p is the number of compared problems, N_m is the number of considered LLMs, $M_{p,m}$ is the performance of a heuristic for the p-th problem with m-th LLM. M_p^* is the best-known performance for the p-th problem. And I_p is one if a higher value indicates better performance for the p-th problem (i.e., for maximization problems) and zero otherwise (i.e., for minimization problems).

Other Settings. We initialize all compared methods with the respective template heuristic on each problem. The details of the template heuristics are provided in Appx. §C. For EoH [60], we fill the remaining of the initial population with randomly generated heuristics. For both EoH [60] and ReEvo [15], we increase the population size as well as maximum number of evaluations. We perform ablation studies on this in Appx. §B. Table 3 summarizes the benchmark hyper-parameter settings.

4 Experimental Results and Analyses

4.1 Performance of Standalone LLMs on AHD

Motivation. Standalone LLMs have consistently showcased exceptional performance across a diverse array of AI applications, reaching a point where the research community has come to expect impressive results from them on new and challenging tasks. To this end, we wonder whether the inherent generative capability of LLMs alone (without coupling with an evolutionary search mechanism) would suffice for AHD tasks. In this work, we attempt to answer this question from the following two angles.

Table 3. Summary of benchmark settings.

Description of Setting	Value
Maximum number of function evaluations (#FE)	10,000
Population size (for EoH and ReEvo)	100
# of islands, # of samples per prompt (for FunSearch)	10, 4
Number of independent runs per experiment	5
Maximum evaluation time for each heuristic (to cope with invalid heuristics, such as infinite loops)	50 s (TSP); 20 s (others)

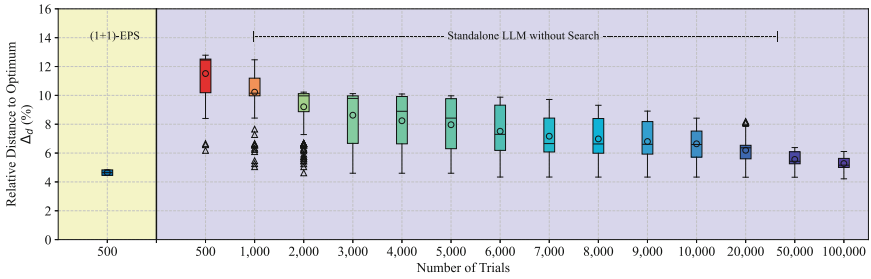
4.1.1 Angle I: Impact of Query Budget

Experimental Design. Firstly, we aim to validate the performance of standalone LLMs on AHD problems under different query budgets, i.e., maximum # of queries allowed to be sent to LLMs. Given an AHD problem, we provide the problem context along with the template heuristic (i.e., f_T in Algorithm 1) as prompts to a LLM, and we ask it to keep generating new heuristics until query budgets are exhausted. Note that we do not proactively check for duplicate heuristics simply because no effective tools for functionality-level duplicate detection are readily available.

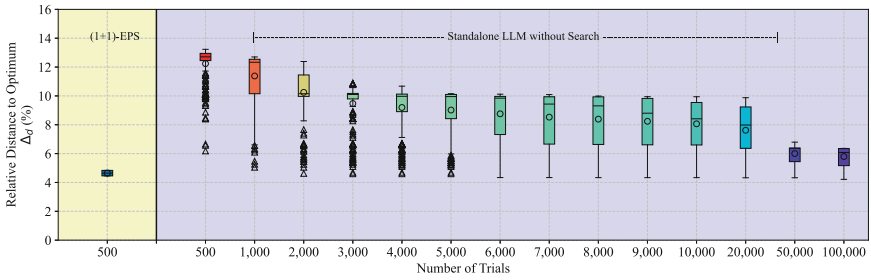
Results. Figure 2 depicts the aggregated performance (i.e., mean Δ_{d} over four AHD problems) of the heuristics generated by standalone GPT-3.5 with various query budgets. We also include the performance of the heuristics obtained by our baseline $(1+1)$-EPS as a reference. Due to space constraints, more detailed results on individual AHD problems with different LLMs are discussed elsewhere.

Our analysis reveals that while the performance of standalone LLMs on AHD problems generally improves with increasing query budgets, several critical observations emerge:

- There remains a significant gap between the performance of heuristics generated by standalone LLMs and the best-known performance (indicated by $\Delta_{\mathrm{d}} = 0$, i.e., x-axis in Fig. 2), even with a substantial query budget of 100,000.
- Although there is a steady improvement in the mean performance of the top-ranked heuristics, the performance of the best individual heuristics (represented by the lower bars of the boxes) shows minimal enhancement as query budgets increase.



(a) Top-5‰ heuristics



(b) Top-1% heuristics

Fig. 2. Box plot comparison on the performance of the top-{(a) 5‰, (b) 1%} heuristics generated by GPT-3.5 under various query budgets. The performance is measured as the relative distance to the best-known optimum (Δ_{d}) aggregated over four AHD problems and five independent runs. Lower Δ_{d} indicates better performance. The performance of the simple baseline $(1+1)$-EPS with GPT-3.5 under a small query budget of 500 is also provided as a reference.

- Standalone LLMs are highly ineffective[2] on AHD problems, even when granted an order of magnitude more queries.

In summary, these observations suggest that merely increasing the number of attempts by a standalone LLM is insufficient for effectively addressing AHD problems. This underscores *the need to integrate LLMs with search methods to enhance their efficacy in AHD contexts.*

4.1.2 Angle II: Impact of More Capable LLMs

Experimental Design. Next, we attempt to understand the relationship between LLMs' capacity and their performance on AHD problems. In this work,

[2] The ineffectiveness is in the sense that a simple EPS baseline achieves better mean performance with much lower variances than standalone LLMs with an order of magnitude more query budget, as depicted in Fig. 2.

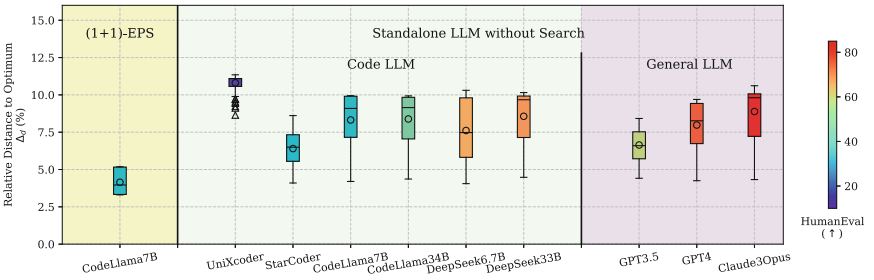


Fig. 3. Box plot comparison on the performance of the top-5‰ heuristics generated by LLMs with varying capacities under 10,000 query budgets. We group LLMs into two categories: (1) LLMs specialized for coding tasks (with background shaded in) and (2) general-purpose LLMs (with background shaded in). Then, the LLMs are arranged in the order of ascending model size within each group. The color scale of the boxes corresponds with the scores on HumanEval [62]. The performance is measured as the relative distance to the best-known optimum (Δ_d) aggregated over four AHD problems and five independent runs. Lower Δ_d indicates better performance. The performance of the simple baseline $(1+1)$-EPS with CodeLlama-7B is also provided as a reference.

Table 4. The performance of the top-1 heuristics generated by LLMs with varying capacities. The performance is measured as the relative distance to the best-known optimum (Δ_d) aggregated over four AHD problems and five independent runs. Lower Δ_d indicates better performance.

UniXcoder	DeepSeek -6.7B	CodeLlama -7B	StarCoder 15.5B	DeepSeek -33B	CodeLlama -34B	GPT-3.5	Claude 3 Opus	GPT-4
9.15%	4.24%	4.32%	4.21%	4.59%	4.48%	4.31%	4.63%	4.44%

we consider the model size (in terms of # of parameters), the coding performance (in terms of HumanEval scores [62]), and the general performance across many tasks (in terms of MMLU scores [63]) as proxy indicators for measuring a LLM's capacity. A diverse set of nine different LLMs is considered, with more details provided in Table 2. Specifically, given an AHD problem, we provide the problem context along with the template heuristic (i.e., f_T in Algorithm 1) as prompts to a LLM, and we ask it to keep generating new heuristics until the query budget of 10,000 is exhausted.

Results. Figure 3 depicts the aggregated performance (i.e., mean Δ_d over four AHD problems) of the heuristics generated by LLMs with varying capacities. We also include the performance of the heuristics obtained by our baseline $(1+1)$-EPS with CodeLlama-7B (i.e., a small-capacity LLM) as a reference. In addition, we provide the performance of the top-1 heuristics generated by various LLMs in Table 4. Constrained by space, more elaborated results on individual AHD problems with different thresholds on filtering top heuristics are provided in Appx. §B. Evidently, we make the following observations:

- LLMs with more capacity (i.e., more # of model parameters, better HumanEval and MMLU scores) do not necessarily lead to better performance on AHD problems.
- LLMs fine-tuned for coding tasks (i.e., the group with background shaded in) are not statistically better than general purpose LLMs (i.e., the group with background shaded in).
- Standalone LLMs with large overall model capacity are still highly ineffective[3] on AHD problems.
- Conventional LLMs, i.e., variants of BERT [66] such as UniXcoder [64], are significantly inferior to modern LLMs (e.g., GPTs) on AHD problems.

In summary, the above observations suggest that simply importing more capable LLMs is insufficient for tackling AHD problems, reinforcing *the need to integrate LLMs with search methods to enhance their efficacy in AHD contexts.*

4.1.3 Summary and Implications

Observations from the previous sections have converged to a consensus that the inherent generative capability of **LLMs alone is insufficient for AHD** problems, which holds true under increased query budget (Sect. 4.1.1 Angle I) and model capacity (Sect. 4.1.2 Angle II), suggesting the **necessity of coupling LLMs with a search strategy** to tackle AHD problems effectively.

Given the modular yet flexible framework, we believe that the LLM-based EPS paradigm, **synergizing LLMs with an evolutionary search strategy**, is a meaningful approach to addressing the general AHD problems.

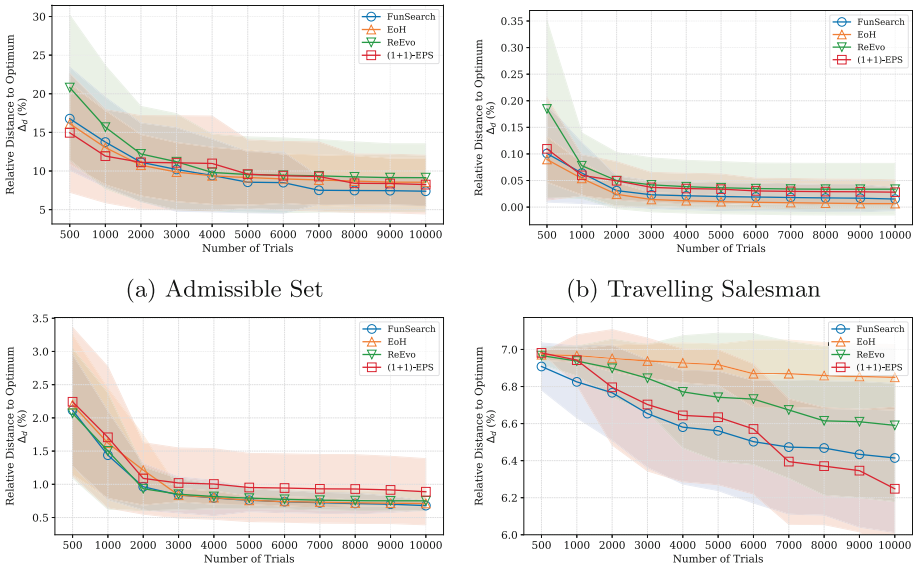
4.2 Performance of Existing LLM-Based EPS Methods on AHD

We decompose our investigations into the following two angles to establish an empirical understanding of the progress made by the existing LLM-based EPS methods on AHD.

4.2.1 Angle I: Relative Improvements over Adequate Baseline

Motivation. Existing LLM-based EPS methods incorporate a variety of complications in the search and the prompt components (see Sect. 2 for more details). The relative improvements contributed by these modifications are primarily evaluated against random search or simple heuristics derived through human intuitions. On the one hand, whether the observed improvements over these naive baselines meaningfully capture the advancement in algorithmic design remains questionable; while on the other hand, the general utility of the enhancements introduced by various EPS methods also remains to be further evaluated.

[3] The ineffectiveness is in the sense that a low-capacity LLM coupled with a simple EPS baseline significantly standalone LLMs with orders of magnitude more model capacities (e.g., GPT-4 and Claude 3 Opus), as depicted in Fig. 3.



(a) Admissible Set (b) Travelling Salesman

(c) Online Bin Packing (Weibull) (d) Online Bin Packing (OR)

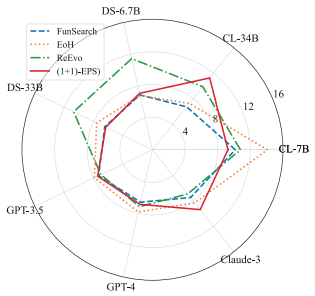
Fig. 4. Convergence curve comparison on the performance of the top-1 heuristics achieved by various EPS methods. The mean relative distances to the best-known optimum (Δ_d) averaged over five independent runs are denoted with markers, while the standard deviations of Δ_d are shown with the shaded regions.

Experimental Design. We benchmark existing LLM-based EPS methods (i.e., FunSearch [12], EoH [60], and ReEvo [15]) against the proposed baseline (1 + 1)-EPS on four AHD problems with seven LLMs[4]. We repeat each experiment five times with different random seeds. All other benchmark settings are identical to those described in Sect. 3 unless otherwise specified.

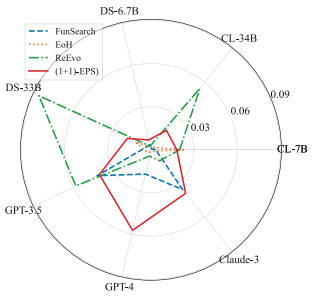
Results. Figure 4 compares the aggregated performance (i.e., mean Δ_d over seven LLMs and five independent runs) among existing EPS methods and the proposed baseline on four AHD problems. Evidently, we observe that:

- Performance varies significantly across different problems for all existing LLM-based EPS methods, with no single method demonstrating consistent superiority.
- Specifically, the EoH method consistently outperforms all others in the TSP problem throughout the search process, while the simple baseline (1 + 1)-EPS shows competitive performance, except in the OBP (Weibull) problem.

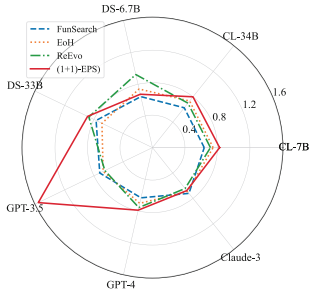
[4] We exclude UniXcoder and StarCoder from Table 2 as they are mainly designed for code completion, which are not compatible with EoH and ReEvo that also require comprehension of natural languages.



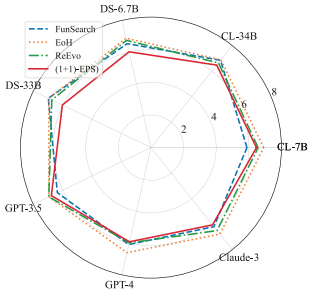
(a) Admissible Set



(b) Travalling Salesman



(c) Online Bin Packing (Weibull)



(d) Online Bin Packing (OR)

Fig. 5. Radar plot comparison on the performance of the top-1 heuristics achieved by various EPS methods with different choices of LLMs. The radius of each vertex is calculated by the mean relative distances to the best-known optimum (Δ_{d}) averaged over five independent runs; hence, a smaller radius/enclosed area indicates better performance. We use "CL" and "DS" to denote the CodeLlama and DeepSeek models, respectively.

These empirical findings suggest that there may not be universally effective and efficient LLM-based EPS method for all AHD problems, reinforcing the applicability of the "no free lunch" (NFL) theorem to AHD.

4.2.2 Angle II: Dependency on the Choice of LLMs

Motivation. Existing LLM-based EPS methods are typically evaluated using only one particular choice of LLMs [15,60,67]. This raises uncertainty regarding the extent to which performance enhancements suggested by these methods can be applied to other LLM choices. Compounding this issue, the predominant LLM utilized in these EPS methods, i.e., GPT-3.5, is closed-source in nature. Should the efficacy of existing EPS methods hinge significantly on closed-source LLMs, the geographically restricted access to APIs may impede future development built upon these methods.

Experimental Design. The experimental setup is identical to those described in Sect. 4.2.1, except on the utilization of different LLMs. In this case, we do not aggregate experiments across various LLMs. Instead, we aim to directly compare the performance under different LLM choices for each EPS method.

Results. Figure 5 compares the final performance of various EPS methods under different LLMs across four AHD problems. From this comparison, we draw two main observations:

- There are significant variances in performance attributable to the choice of LLM for all EPS methods, with the notable exception of the OBP (OR) problem where this variance is marginal.
- Specifically, the EoH method shows stable and robust performance on the TSP problem across all LLMs, whereas the $(1 + 1)$-EPS's performance varies considerably due to its greedy nature.

These findings underscore the dependence of EPS methods' performance on the specific LLMs employed.

4.2.3 Summary and Implications

The empirical observations from previous sections jointly suggest that the LLM-based EPS algorithmic development is still in the early stages. We hypothesize that *more diverse benchmarks and applications are needed to establish a better understanding of this emergent paradigm for AHD*. Nevertheless, these preliminary results also prompt us to (*i*) rethink the general efficacy of various components (such as prompt engineering and search strategy) within the overall paradigm, (*ii*) consider incorporating domain knowledge to LLM-based EPS algorithm design, and (*iii*) use a variety of LLMs to gain a more robust evaluation of the performance of EPS methods.

5 Conclusion

This work presents a large-scale benchmark study comprising all existing LLM-based EPS methods along with a new proposed baseline and four AHD problems over (up-to) nine different LLMs and five independent runs. Based on the analyses from multiple comparison angles, we reveal novel insights into the necessity and the current progress of the LLM-based EPS paradigm for AHD. On top of them, we summarize a few tangible implications for future research directions for LLM-based EPS, along with the fully released source codes for fostering future development.

Acknowledgements. The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (GRF Project No. CityU11215622), the National Natural Science Foundation of China (Grant No. 62106096), the Natural Science Foundation of Guangdong Province (Grant No. 2024A1515011759), the National Natural Science Foundation of Shenzhen (Grant No. JCYJ20220530113013031).

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

References

1. Burke, E.K., et al.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**, 1695–1724 (2013)
2. Stützle, T., López-Ibáñez, M.: Automated design of metaheuristic algorithms. In: Handbook of Metaheuristics, pp. 541–579 (2019)
3. Wu, X., Consoli, P., Minku, L., Ochoa, G., Yao, X.: An evolutionary hyper-heuristic for the software project scheduling problem. In: International Conference on Parallel Problem Solving from Nature (2016)
4. Chen, T., et al.: Learning to optimize: a primer and a benchmark. J. Mach. Learn. Res. **23**(189), 1–59 (2022)
5. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Practice and Theory of Automated Timetabling (2001)
6. Mockus, J.: Application of bayesian approach to numerical methods of global and stochastic optimization. J. Global Optim. **4**, 347–365 (1994)
7. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Stat. Comput. **4**, 87–112 (1994)
8. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-662-04726-2
9. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. IEEE Trans. Evol. Comput. **28**(1), 147–167 (2024)
10. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Importance-aware genetic programming for automated scheduling heuristics learning in dynamic flexible job shop scheduling. In: International Conference on Parallel Problem Solving from Nature (2022)
11. O'Neill, M., Vanneschi, L., Gustafson, S., Banzhaf, W.: Open issues in genetic programming. Genet. Program. Evol. Mach. **11**(3), 339–363 (2010)
12. Romera-Paredes, B., et al.: Mathematical discoveries from program search with large language models. Nature **625**(7995), 468–475 (2024)
13. Tao, T., Vu, V.H.: Additive Combinatorics. Cambridge University Press, Cambridge (2006)
14. Liu, F., et al.: Evolution of heuristics: towards efficient automatic algorithm design using large language model. In: International Conference on Machine Learning (2024)
15. Ye, H., Wang, J., Cao, Z., Song, G.: Reevo: large language models as hyper-heuristics with reflective evolution. arXiv preprint arXiv:2402.01145 (2024)
16. Matai, R., Singh, S.P., Mittal, M.L.: Traveling salesman problem: an overview of applications, formulations, and solution approaches. Travel. Salesman Prob. Theory Appl. **1**(1), 1–25 (2010)

17. Seiden, S.S.: On the online bin packing problem. J. ACM **49**(5), 640–671 (2002)
18. Hansen, N.: The CMA evolution strategy: a tutorial. arXiv preprint arXiv:1604.00772 (2016)
19. Brown, T., et al.: Language models are few-shot learners. Adv. Neural Inf. Process. Syst. (2020)
20. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A Classification of Hyper-heuristic Approaches. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 146, pp. 449–468. Springer, Boston (2010)
21. He, X., Zhao, K., Chu, X.: Automl: a survey of the state-of-the-art. Knowl.-Based Syst. **212**, 106622 (2021)
22. Burke, E.K., Petrovic, S., Qu, R.: Case-based heuristic selection for timetabling problems. J. Sched. **9**, 115–132 (2006)
23. Ross, H.-L. F.P., Corne, D.: A promising hybrid GA/heuristic approach for open-shop scheduling problems. In: European Conference on Artificial Intelligence (1994)
24. Hart, E., Ross, P., Nelson, J.: Solving a real-world problem using an evolving heuristically driven schedule builder. Evol. Comput. **6**(1), 61–80 (1998)
25. Terashima-Marín, H., Flores-Alvarez, E., Ross, P.: Hyper-heuristics and classifier systems for solving 2d-regular cutting stock problems. In: Annual Conference on Genetic and Evolutionary Computation (2005)
26. Rodríguez, J.V., Petrovic, S., Salhi, A.: A combined meta-heuristic with hyper-heuristic approach to the scheduling of the hybrid flow shop with sequence dependent setup times and uniform machines. In: Multidisciplinary International Conference on Scheduling: Theory and Applications. MISTA: Paris, France (2007)
27. Burke, E.K., Hyde, M.R., Kendall, G.: Evolving bin packing heuristics with genetic programming. In: International Conference on Parallel Problem Solving from Nature (2006)
28. Duflo, G., Kieffer, E., Brust, M.R., Danoy, G., Bouvry, P.: A GP hyper-heuristic approach for generating tsp heuristics. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (2019)
29. Rego, C., Gamboa, D., Glover, F., Osterman, C.: Traveling salesman problem heuristics: leading methods, implementations and latest advances. Eur. J. Oper. Res. **211**(3), 427–441 (2011)
30. Drechsler, R., Becker, B.: Learning heuristics by genetic algorithms. In: ASP-DAC'95/CHDL'95/VLSI'95 with EDA Technofair (1995)
31. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated design of production scheduling heuristics: a review. IEEE Trans. Evol. Comput. **20**(1), 110–124 (2015)
32. Vaswani, A., et al.: Attention is all you need. Adv. Neural Inf. Process. Syst. (2017)
33. Achiam, J., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
34. Zhao, W.X., et al.: A survey of large language models. arXiv preprint arXiv:2303.18223 (2023)
35. Tian, H., et al.: chatgpt the ultimate programming assistant–how far is it?. arXiv preprint arXiv:2304.11938 (2023)
36. Yu, C., Liu, X., Tang, C., Feng, W., Lv, J.: GPT-NAS: neural architecture search with the generative pre-trained model. arXiv preprint arXiv:2305.05351 (2023)
37. Zhang, S., Gong, C., Wu, L., Liu, X., Zhou, M.: Automl-GPT: automatic machine learning with gpt. arXiv preprint arXiv:2305.02499 (2023)
38. Zhou, Y., et al.: Large language models are human-level prompt engineers. arXiv preprint arXiv:2211.01910 (2022)

39. Wang, X., et al.: Promptagent: strategic planning with language models enables expert-level prompt optimization. arXiv preprint arXiv:2310.16427 (2023)
40. Zelikman, E., Lorch, E., Mackey, L., Kalai, A.T.: Self-taught optimizer (stop): recursively self-improving code generation. arXiv preprint arXiv:2310.02304 (2023)
41. Liu, S., Chen, C., Qu, X., Tang, K., Ong, Y.-S.: Large language models as evolutionary optimizers. arXiv preprint arXiv:2310.19046 (2023)
42. Liu, F., et al.: Large language model for multi-objective evolutionary optimization. arXiv preprint arXiv:2310.12541 (2023)
43. Chen, A., Dohan, D., So, D.: EvoPrompting: language models for code-level neural architecture search. Adv. Neural Inf. Process. Syst. (2024)
44. Meyerson, E., et al.: Language model crossover: variation through few-shot prompting. arXiv preprint arXiv:2302.12170 (2023)
45. Hemberg, E., Moskal, S., O'Reilly, U.-M.: Evolving code with a large language model. arXiv preprint arXiv:2401.07102 (2024)
46. Yang, C., et al.: Large language models as optimizers. arXiv preprint arXiv:2309.03409 (2023)
47. Guo, Q., et al.: Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. arXiv preprint arXiv:2309.08532 (2023)
48. Lehman, J., Gordon, J., Jain, S., Ndousse, K., Yeh, C., Stanley, K.O.: Evolution through large models (2022)
49. Wu, X., Wu, S.-H., Wu, J., Feng, L., Tan, K.C.: Evolutionary computation in the era of large language model: survey and roadmap. arXiv preprint arXiv:2401.10034 (2024)
50. Code models overview (2023)
51. Li, R., et al.: Starcoder: may the source be with you!. arXiv preprint arXiv:2305.06161 (2023)
52. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural Inf. Process. Syst. (2022)
53. Guo, D., et al.: Deepseek-coder: when the large language model meets programming–the rise of code intelligence. arXiv preprint arXiv:2401.14196 (2024)
54. Roziere, B., et al.: Code llama: open foundation models for code. arXiv preprint arXiv:2308.12950 (2023)
55. Holland, J.H.: Genetic algorithms. Sci. Am. **267**(1), 66–73 (1992)
56. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: language agents with verbal reinforcement learning. Adv. Neural Inf. Process. Syst. (2024)
57. Grochow, J.: New applications of the polynomial method: the cap set conjecture and beyond. Bull. Am. Math. Soc. **56**(1), 29–64 (2019)
58. Beasley, J.E.: Or-library: distributing test problems by electronic mail. J. Oper. Res. Soc. **41**(11), 1069–1072 (1990)
59. Castiñeiras, I., De Cauwer, M., O'Sullivan, B.: Weibull-based benchmarks for bin packing. In: International Conference on Principles and Practice of Constraint Programming (2012)
60. Liu, F., et al.: An example of evolutionary computation+ large language model beating human: design of efficient guided local search. arXiv preprint arXiv:2401.02051 (2024)
61. Kool, W., Van Hoof, H., Welling, M.: Attention, learn to solve routing problems!. arXiv preprint arXiv:1803.08475 (2018)
62. Chen, M., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021)
63. Hendrycks, D., et al.: Measuring massive multitask language understanding. arXiv preprint arXiv:2009.03300 (2020)

64. Guo, D., Lu, S., Duan, N., Wang, Y., Zhou, M., Yin, J.: UniXcoder: unified cross-modal pre-training for code representation. In: Annual Meeting of the Association for Computational Linguistics (2022)
65. Anthropic. The claude 3 model family: Opus, sonnet, haiku (2024)
66. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Conference of the North American Chapter of the Association for Computational Linguistics (2019)
67. Ma, Y.J., et al.: Eureka: human-level reward design via coding large language models. In: International Conference on Learning Representations (2024)

Numerical Optimization

Warm Starting of CMA-ES for Contextual Optimization Problems

Yuta Sekino(✉), Kento Uchida, and Shinichi Shirakawa

Yokohama National University, Yokohama, Japan
sekino-yuta-cs@ynu.jp, {uchida-kento-fz,shirakawa-shinichi-bg}@ynu.ac.jp

Abstract. Several practical applications of evolutionary computation possess objective functions that receive the design variables and externally given parameters. Such problems are termed contextual optimization problems. These problems require finding the optimal solutions corresponding to the given context vectors. Existing contextual optimization methods train a policy model to predict the optimal solution from context vectors. However, the performance of such models is limited by their representation ability. By contrast, warm starting methods have been used to initialize evolutionary algorithms on a given problem using the optimization results on similar problems. Because warm starting methods do not consider the context vectors, their performances can be improved on contextual optimization problems. Herein, we propose a covariance matrix adaptation evolution strategy with contextual warm starting (CMA-ES-CWS) to efficiently optimize the contextual optimization problem with a given context vector. The CMA-ES-CWS utilizes the optimization results of past context vectors to train the multivariate Gaussian process regression. Subsequently, the CMA-ES-CWS performs warm starting for a given context vector by initializing the search distribution using posterior distribution of the Gaussian process regression. The results of the numerical simulation suggest that CMA-ES-CWS outperforms the existing contextual optimization and warm starting methods.

Keywords: contextual optimization · warm starting · covariance matrix adaptation evolution strategy · Gaussian process regression · initialization

1 Introduction

In practical applications of evolutionary algorithms, the evaluation value of the given objective function is often determined using the design variables and externally given parameters. These optimization problems are referred to as contextual optimization problems and require finding the optimal solution corresponding to the given external parameter, called context vector. Contextual optimization problems arise in many applications, such as the optimization of the controller model of a robot for target motion [2,4] and nuclear fusion control using

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 205–220, 2024.
https://doi.org/10.1007/978-3-031-70068-2_13

plasma states as context vectors [6]. For example, in the controller model of robot locomotion task [2], the target locomotion speed and the locomotion direction are possible choices of the context vector, and the objective of optimization is to design a controller model that realizes the locomotion of the robot with the target speed.

Various contextual optimization methods have been proposed to efficiently optimize these contextual optimization problems [1,3,13]. These methods solve contextual optimization problems by training the policy model that receives the context vector and predicts the corresponding optimal solution. To reduce the training cost of the policy model in contextual optimizations, several studies [3,4] have focused on the update rules of the covariance matrix adaptation evolution strategy (CMA-ES) [11], which employs a multivariate Gaussian distribution as the sampling distribution and iteratively updates the distribution parameters. The contextual CMA-ES [3] extends the update rules of the CMA-ES to update the policy model. Notably, because many contextual optimization methods use linear models as the policy model, their ability to predict the optimal solution is limited.

Meanwhile, warm starting methods have been used to start the optimization of evolutionary algorithms from a good state on a given problem. The warm starting methods initialize the evolutionary algorithm using the optimization results on similar problems [14,18]. For example, the warm starting CMA-ES (WS-CMA-ES) initializes the multivariate Gaussian distribution using the set of superior solutions achieved in the optimization of a given similar task. However, the existing warm starting methods do not consider the context vectors. We are of the view that the performance of warm starting methods on contextual optimization problems can be improved using the information of context vectors. A related idea was introduced in the contextual bandit problems [19], which utilize a dataset consisting of a set of context vectors provided in advance by experts and a set of reward vectors for each action.

In this study, we propose the CMA-ES with contextual warm starting (CMA-ES-CWS) to efficiently obtain the optimal solution corresponding to a given context vector. CMA-ES-CWS requires the best solutions corresponding to some context vectors achieved in past optimizations. Given the optimization results for past context vectors, CMA-ES-CWS trains the multi-output Gaussian process regression (GPR) to predict the optimal solution from the context vector. Next, CMA-ES-CWS performs warm starting for a newly obtained context vector by initializing the search distribution using the predictive distribution of the GPR.

We evaluate the performance of the CMA-ES-CWS using the benchmark functions and control task of the robot arm. In the experiment with benchmark functions, we transform the search space of existing benchmark functions based on the context vector. Experimental results show that, with nonlinear and noisy transformations, CMA-ES-CWS can obtain the optimal solution corresponding to the given context vector more efficiently than the contextual CMA-ES and WS-CMA-ES. In addition, we confirm that the performance of CMA-ES-CWS

is improved by increasing the amount of past optimization results. Further, we evaluate the performance of CMA-ES-CWS in robot control tasks.

2 Preliminaries

2.1 CMA-ES

The CMA-ES is a probabilistic model-based black-box continuous optimization method using the multivariate Gaussian distribution $\mathcal{N}\left(\boldsymbol{m}, \sigma^2 \boldsymbol{C}\right)$ which is parameterized by mean vector $\boldsymbol{m}^{(t)} \in \mathbb{R}^N$, covariance matrix $\boldsymbol{C}^{(t)} \in \mathbb{R}^{N \times N}$, and step-size $\sigma^{(t)} \in \mathbb{R}_{>0}$.

In each iteration, the CMA-ES generates λ solutions $\boldsymbol{x}^{\langle 1 \rangle}, \cdots, \boldsymbol{x}^{\langle \lambda \rangle}$ from the Gaussian distribution as

$$\boldsymbol{z}^{\langle i \rangle} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad , \quad \boldsymbol{y}^{\langle i \rangle} = \sqrt{\boldsymbol{C}^{(t)}} \boldsymbol{z}^{\langle i \rangle} \quad \text{and} \quad \boldsymbol{x}^{\langle i \rangle} = \boldsymbol{m}^{(t)} + \sigma^{(t)} \boldsymbol{y}^{\langle i \rangle} \ . \tag{1}$$

Next, the CMA-ES computes the ranking of the solutions on the objective function. We denote the index of the i-th best solution as $i{:}\lambda$.

Next, the CMA-ES updates two evolution paths: $\boldsymbol{p}_c \in \mathbb{R}^N$ and $\boldsymbol{p}_\sigma \in \mathbb{R}^N$. These evolution paths are initialized as $\boldsymbol{p}_\sigma^{(0)} = \boldsymbol{p}_c^{(0)} = \mathbf{0}$. The CMA-ES computes two weighted sums, namely, $\boldsymbol{\Delta}_{\boldsymbol{z}} = \sum_{i=1}^{\mu} w_i \boldsymbol{z}^{\langle i:\lambda \rangle}$ and $\boldsymbol{\Delta}_{\boldsymbol{y}} = \sum_{i=1}^{\mu} w_i \boldsymbol{y}^{\langle i:\lambda \rangle}$ of μ best solutions, to update the evolution paths as

$$\boldsymbol{p}_\sigma^{(t+1)} = (1 - c_\sigma) \boldsymbol{p}_\sigma^{(t)} + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} \cdot \boldsymbol{\Delta}_{\boldsymbol{z}} \tag{2}$$

$$\boldsymbol{p}_c^{(t+1)} = (1 - c_c) \boldsymbol{p}_c^{(t)} + h_\sigma^{(t+1)} \sqrt{c_c (2 - c_c) \mu_{\text{eff}}} \cdot \boldsymbol{\Delta}_{\boldsymbol{y}} \ , \tag{3}$$

where $c_\sigma, c_c \in (0, 1]$ are accumulation factors, and $\mu_{\text{eff}} = (\sum_{i=1}^{\mu} w_i^2)^{-1}$ is the variance effective selection mass. The Heaviside function $h_\sigma \in \{0, 1\}$ is set to $h_\sigma = 1$ when it satisfies

$$\frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|}{\sqrt{1 - (1 - c_\sigma)^{2(t+1)}}} < \left(1.4 + \frac{2}{N + 1}\right) \chi_N \ , \tag{4}$$

where $\chi_N = \sqrt{N}\left(1 - \frac{1}{4N} + \frac{1}{21N^2}\right)$ denotes the approximated value of $\mathbb{E}[\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|]$.

Finally, the CMA-ES updates the distribution parameters. The update rule of the mean vector is given as

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + c_m \sigma^{(t)} \boldsymbol{\Delta}_{\boldsymbol{y}} \ , \tag{5}$$

where the learning rate $c_m \in (0, 1]$ is usually set as $c_m = 1$. The update of the covariance matrix consists of the rank-μ update and the rank-one update. The rank-μ update uses the weighted sum of the μ best solutions, whereas the

rank-one update uses the evolution path $\boldsymbol{p}_c$. The update rule of the covariance matrix is given by

$$\boldsymbol{C}^{(t+1)} = \left(1 + (1 - h_\sigma^{(t+1)}) c_1 c_c (2 - c_c)\right) \boldsymbol{C}^{(t)} + \underbrace{c_\mu \sum_{i=1}^{\mu} w_i \left(\boldsymbol{y}^{\langle i:\lambda \rangle} \left(\boldsymbol{y}^{\langle i:\lambda \rangle}\right)^{\mathrm{T}} - \boldsymbol{C}^{(t)}\right)}_{\text{rank-}\mu\text{ update}} + \underbrace{c_1 \left(\boldsymbol{p}_c^{(t+1)} \left(\boldsymbol{p}_c^{(t+1)}\right)^{\mathrm{T}} - \boldsymbol{C}^{(t)}\right)}_{\text{rank-one update}} , \quad (6)$$

where $c_\mu, c_1 \in (0, 1]$ denote the learning rates. The update rule of the step-size is given as

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|}{\chi_N} - 1\right)\right) , \quad (7)$$

where $d_\sigma \in \mathbb{R}_{>0}$ denotes the damping factor. The CMA-ES has well-tuned recommended settings for each hyperparameter, as shown in [9,10].

The initial distribution parameters $\boldsymbol{m}^{(0)}$, $\boldsymbol{C}^{(0)}$, and $\sigma^{(0)}$ significantly influence the optimization performance. However, because the precise general-purpose method to determine these important hyperparameters does not exist, they are manually set (or sometimes manually tuned) for the target problem.

2.2 Multi-output Gaussian Process Regression

The GPR is a non-parametric regression method, which expresses the prediction using the posterior distribution of the function following a Gaussian process. The Gaussian process is a function g of the distribution whose evaluation values at arbitrary n points $\mathcal{D}_n = \{\boldsymbol{x}_i\}_{i=1}^n$ follow the multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$.

The prediction of GPR uses the conditional distribution of the multivariate Gaussian distribution. We denote sample $\boldsymbol{x}$, mean vector $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma}$ as

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{a,a} & \boldsymbol{\Sigma}_{a,b} \\ \boldsymbol{\Sigma}_{b,a} & \boldsymbol{\Sigma}_{b,b} \end{pmatrix} . \quad (8)$$

Then, the distribution of $\boldsymbol{x}_b$ conditioned on $\boldsymbol{x}_a$ is given by a multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}_{b|a}$ and covariance matrix $\boldsymbol{\Sigma}_{b|a}$ as

$$\boldsymbol{\mu}_{b|a} = \boldsymbol{\mu}_b + \boldsymbol{\Sigma}_{b,a}\boldsymbol{\Sigma}_{b,b}^{-1}(\boldsymbol{x}_b - \boldsymbol{\mu}_b) \quad \text{and} \quad \boldsymbol{\Sigma}_{b|a} = \boldsymbol{\Sigma}_{b,b} - \boldsymbol{\Sigma}_{b,a}\boldsymbol{\Sigma}_{a,a}^{-1}\boldsymbol{\Sigma}_{a,b} . \quad (9)$$

Here, we consider the multi-output GPR that predicts a function g with L outputs. In this case, the vector $\boldsymbol{g}_n = (g(\boldsymbol{x}_1)^{\mathrm{T}}, \cdots, g(\boldsymbol{x}_n)^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{Ln}$ that consists of evaluation values at points in $\mathcal{D}_n$ follows the (Ln)-dimensional multivariate Gaussian distribution whose covariance matrix is given by the Gram matrix $\boldsymbol{K}_L(\mathcal{D}_n) \in \mathbb{R}^{(Ln)\times(Ln)}$. There are several methods to construct the Gram matrix

for multi-output GPR. We use the linear model of coregionalization (LMC) [12]. The LMC introduces Q matrices $\boldsymbol{B}_1, \cdots, \boldsymbol{B}_Q \in \mathbb{R}^{L \times L}$ and Q kernels $k_1, \cdots, k_Q$ and computes the Gram matrix as

$$\boldsymbol{K}_L(\mathcal{D}_n) = \sum_{q=1}^{Q} \boldsymbol{K}(\mathcal{D}_n; k_q) \otimes \boldsymbol{B}_q = \begin{pmatrix} \boldsymbol{K}_L(\mathcal{D}_{n-1}) & \boldsymbol{K}_* \\ \boldsymbol{K}_*^{\mathrm{T}} & \boldsymbol{K}_{**} \end{pmatrix} \ , \tag{10}$$

where the operation $\otimes$ is the Kronecker product, and $\boldsymbol{K}_* \in \mathbb{R}^{Ln \times L}$ and $\boldsymbol{K}_{**} \in \mathbb{R}^{L \times L}$ are components of the decomposition of the Gram matrix. The covariance matrix $\boldsymbol{K}(\mathcal{D}_n; k_q)$ using the kernel function $k_q : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is given by

$$\boldsymbol{K}(\mathcal{D}_n; k_q) = \begin{pmatrix} k_q(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k_q(\boldsymbol{x}_n, \boldsymbol{x}_1) \\ \vdots & & \vdots \\ k_q(\boldsymbol{x}_1, \boldsymbol{x}_n) & \cdots & k_q(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix} \ . \tag{11}$$

The LMC computes the predictive distribution of the function output at $\boldsymbol{x}_n$ as a L-dimensional multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}_n), \boldsymbol{\Sigma}(\boldsymbol{x}_n))$, where

$$\boldsymbol{\mu}(\boldsymbol{x}_n) = \boldsymbol{K}_*^{\mathrm{T}} \left(\boldsymbol{K}_L(\mathcal{D}_{n-1})\right)^{-1} \boldsymbol{g}_{n-1} \tag{12}$$

$$\boldsymbol{\Sigma}(\boldsymbol{x}_n) = \boldsymbol{K}_{**} - \boldsymbol{K}_*^{\mathrm{T}} \left(\boldsymbol{K}_L(\mathcal{D}_{n-1})\right)^{-1} \boldsymbol{K}_* \ . \tag{13}$$

In the prediction of LMC, the d-th element of the function output $g(\boldsymbol{x}_n)$ can be interpreted as the weighted sum of QR Gaussian processes with Q kernels as

$$(g(\boldsymbol{x}))_d = \sum_{q=1}^{Q} \sum_{r=1}^{R} a_{d,q}^r u_q^r(\boldsymbol{x}) \ , \tag{14}$$

where $u_q^r : \mathbb{R}^N \to \mathbb{R}$ is the sample from the Gaussian process with kernel k_q. Coefficient $a_{d,q}^r \in \mathbb{R}$ is set such that as to satisfy $(\boldsymbol{B}_q)_{d,d'} = \sum_{r=1}^{R} a_{d,q}^r a_{d',q}^r$.

3 Problem Definition

We consider a contextual optimization problem whose objective function $f(\boldsymbol{x}, \boldsymbol{\alpha})$ is determined by context vector $\boldsymbol{\alpha} \in \mathbb{R}^{N_\alpha}$. The objective of the contextual optimization problem is to obtain an optimal solution corresponding to the given context vector.

Objective of Existing Contextual Optimization Methods: The existing contextual optimization methods [7,8,13] aim to train the policy model $\pi_{\boldsymbol{w}}$ to predict the optimal solution $\boldsymbol{x}^*(\boldsymbol{\alpha})$ from the context vector $\boldsymbol{\alpha}$. Their target is to obtain the optimal parameter $\boldsymbol{w}^*$ of policy model that minimizes the expected objective function value under the distribution p_α of the context vector as

$$\boldsymbol{w}^* = \mathop{\arg\min}_{\boldsymbol{w} \in \mathcal{W}} \int_{\boldsymbol{x}} \int_{\boldsymbol{\alpha}} p_\alpha(\boldsymbol{\alpha}) \pi_{\boldsymbol{w}}(\boldsymbol{x} \mid \boldsymbol{\alpha}) f(\boldsymbol{x}; \boldsymbol{\alpha}) \mathrm{d}\boldsymbol{\alpha} \mathrm{d}\boldsymbol{x} \ . \tag{15}$$

Algorithm 1. CMA-ES with contextual warm-starting (CMA-ES-CWS)

Input: Pairs of context and best solution found previously $\mathcal{D} = \{(\boldsymbol{\alpha}_1, \boldsymbol{x}_1^{\text{best}}), \cdots, (\boldsymbol{\alpha}_{M_{\text{prev}}}, \boldsymbol{x}_{M_{\text{prev}}}^{\text{best}})\}$

Input: Target context vector $\boldsymbol{\alpha}_{\text{new}}$

1: Compute the mean vector $\boldsymbol{\mu}(\boldsymbol{\alpha}_{\text{new}})$ and covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\alpha}_{\text{new}})$ of the predictive distribution using (12) and (13).

2: Set initial distribution parameters to $\boldsymbol{m}^{(0)} = \boldsymbol{\mu}(\boldsymbol{\alpha}_{\text{new}})$, $\boldsymbol{C}^{(0)} = \mathbf{I}$, and $\sigma^{(0)} = \text{clip}(\sqrt{\text{Tr}(\boldsymbol{\Sigma}(\boldsymbol{\alpha}_{\text{new}}))/N}, \sigma_{\min}, \sigma_{\max})$.

3: Run CMA-ES with the initial distribution parameters $\boldsymbol{m}^{(0)}, \sigma^{(0)}, \boldsymbol{C}^{(0)}$.

There are different scenarios of the training in contextual optimization: one scenario is contextual policy search [7,13] where the optimizer receives the context vectors stochastically generated, and another is the active contextual policy search [8], where the optimizer can determine the context vector.

Objective of this Study: Different from the objective of the existing contextual optimization methods, the objective of this study is to achieve efficient optimization of objective function $f(\boldsymbol{x}; \boldsymbol{\alpha}_{\text{new}})$ for $\boldsymbol{x}$ after we receive a target context vector $\boldsymbol{\alpha}_{\text{new}}$. The optimal solution corresponding to a target context vector $\boldsymbol{\alpha}_{\text{new}}$ is formulated as

$$\boldsymbol{x}^*(\boldsymbol{\alpha}_{\text{new}}) = \underset{\boldsymbol{x} \in \mathbb{R}^N}{\arg\min}\, f(\boldsymbol{x}; \boldsymbol{\alpha}_{\text{new}}) \ . \tag{16}$$

In this study, the domain of design variables $\boldsymbol{x}$ is the continuous space $\mathbb{R}^N$. We assume that M_{prev} best solutions $\boldsymbol{x}_1^{\text{best}}, \cdots, \boldsymbol{x}_{M_{\text{prev}}}^{\text{best}}$ for context vectors $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{M_{\text{prev}}}$ are obtained in advance.

4 Proposed Method: CMA-ES-CWS

In this study, we propose CMA-ES with contextual warm-starting (CMA-ES-CWS), which introduces warm starting for contextual optimization problems to CMA-ES. The CMA-ES-CWS utilizes optimization results for past context vectors to train the multi-output GPR. Subsequently, it performs warm starting for a given context vector by initializing the sampling distribution using the predictive distribution of the GPR. Algorithm 1 shows the pseudocode of CMA-ES-CWS.

4.1 Predictive Distribution for Optimal Solution

The CMA-ES-CWS utilizes M_{prev} pairs of context vector and optimization result $\mathcal{D} = \{(\boldsymbol{\alpha}_1, \boldsymbol{x}_1^{\text{best}}), \cdots (\boldsymbol{\alpha}_{M_{\text{prev}}}, \boldsymbol{x}_{M_{\text{prev}}}^{\text{best}})\}$ to predict the distribution of the optimal solution $\boldsymbol{x}_{\text{new}}^*$ corresponding to a newly given context vector $\boldsymbol{\alpha}_{\text{new}}$. The multi-output GPR represents the predictive distribution using a multivariate Gaussian distribution as

$$p(\boldsymbol{x}_{\text{new}}^* \mid \boldsymbol{\alpha}_{\text{new}}, \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{\alpha}_{\text{new}}), \boldsymbol{\Sigma}(\boldsymbol{\alpha}_{\text{new}})) \ . \tag{17}$$

We use LMC to compute the predictive distribution. We apply three kernels for the computation of the Gram matrix in (11): linear kernel k_1, radial basis function (RBF) kernel k_2, and Matern 5/2 kernel k_3 as

$$k_1(\boldsymbol{x}, \boldsymbol{x}') = \sigma_1^2 \boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}' \tag{18}$$

$$k_2(\boldsymbol{x}, \boldsymbol{x}') = \sigma_2^2 \exp\left(-\frac{1}{2}(r_2(\boldsymbol{x}, \boldsymbol{x}'))^2\right) \tag{19}$$

$$k_3(\boldsymbol{x}, \boldsymbol{x}') = \sigma_3^2 \left(1 + \sqrt{5} r_3(\boldsymbol{x}, \boldsymbol{x}') + \frac{5}{3}(r_3(\boldsymbol{x}, \boldsymbol{x}'))^2\right) \exp\left(-\sqrt{5} r_3(\boldsymbol{x}, \boldsymbol{x}')\right) \ , \tag{20}$$

where $\sigma_q \in \mathbb{R}_{>0}$ and $\ell_{q,i} \in \mathbb{R}_{>0}$ are hyperparameters. The RBF and Matern 5/2 kernels are functions of $r_q \in \mathbb{R}_{>0}$ for $q = 2, 3$ defined as $r_q(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{\sum_{i=1}^{N} (x_i - x_i')^2 / \ell_{q,i}^2}$. Matrix $\boldsymbol{B}_q$ is determined using hyperparameters $\boldsymbol{a}_q^r \in \mathbb{R}^N$ and $\kappa \in \mathbb{R}_{>0}$ as

$$\boldsymbol{B}_q = \sum_{r=1}^{R} \boldsymbol{a}_q^r (\boldsymbol{a}_q^r)^{\mathrm{T}} + \kappa \mathbf{I} \ . \tag{21}$$

We set $R = 1$. These hyperparameters of each kernel are optimized through marginal likelihood maximization.[1]

4.2 Warm Starting Using Predictive Distribution

The CMA-ES-CWS uses the mean vector $\boldsymbol{\mu}(\boldsymbol{\alpha}_{\mathrm{new}})$ and covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\alpha}_{\mathrm{new}})$ of the predictive distribution to obtain the initial values of the probability distribution parameters as follows:

$$\boldsymbol{m}^{(0)} = \boldsymbol{\mu}(\boldsymbol{\alpha}_{\mathrm{new}}) \tag{22}$$

$$\sigma^{(0)} = \mathrm{clip}\left(\sqrt{\frac{\mathrm{Tr}(\boldsymbol{\Sigma}(\boldsymbol{\alpha}_{\mathrm{new}}))}{N}}, \sigma_{\min}, \sigma_{\max}\right) \tag{23}$$

$$\boldsymbol{C}^{(0)} = \mathbf{I} \tag{24}$$

where $\mathrm{clip}(a, b, c) = \min\{\max\{a, b\}, c\}$ and $\sigma_{\min} = 10^{-2}, \sigma_{\max} = 2$. This clipping prevents the inefficient optimization caused by a too small initial step-size. The CMA-ES-CWS uses these initial values to search for the optimal solution $\boldsymbol{x}^*(\boldsymbol{\alpha}_{\mathrm{new}})$ through CMA-ES.

Note that CMA-ES-CWS does not use the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\alpha}_{\mathrm{new}})$ of the predictive distribution to initialize the covariance matrix $\boldsymbol{C}$ in CMA-ES. This is because the appropriate distribution shape in CMA-ES depends on the functional shape of the objective function rather than the location of the optimal solution, whereas the predictive distribution given by the multiple-output GPR uses only information of the best solution, not the functional shape.

[1] `Gpy 1.10.0` [17] was used to implement the proposed method.

5 Experiment Using Benchmark Functions

In this section, we present the result of the performance evaluation of CMA-ES-CWS using benchmark functions. In Sect. 5.3, we compare the CMA-ES-CWS with existing methods using benchmark functions. In Sect. 5.4, we verify the relationship between the number of past optimization results M_{prev} and performance of CMA-ES-CWS.

5.1 Comparative Methods

Contextual CMA-ES: Contextual CMA-ES [3] is a contextual optimization method that uses the update rules of CMA-ES. As a traditional problem setting of contextual optimization described in Sect. 3, contextual CMA-ES assumes that the context vector is given randomly before generating a solution.

The contextual CMA-ES trains a policy model to predict the optimal solution corresponding to the context vector $\boldsymbol{\alpha}$. It predicts the optimal solution using the linear model $\boldsymbol{m}^{(t)}(\boldsymbol{\alpha}) = \boldsymbol{A}^{(t)}\varphi(\boldsymbol{\alpha})$ with parameter $\boldsymbol{A} \in \mathbb{R}^{N \times N_\varphi}$, where $\varphi : N_\alpha \to N_\varphi$ determines the context features. In our experiment, we set $\varphi(\boldsymbol{\alpha}) = (\boldsymbol{\alpha}^{\mathrm{T}}, 1)^{\mathrm{T}}$ and $N_\varphi = N + 1$, in accordance with the settings in reference [3]. The contextual CMA-ES acquires the policy model by repeatedly generating samples from the Gaussian distribution $\mathcal{N}(\boldsymbol{m}(\boldsymbol{\alpha}), \sigma^2\boldsymbol{C})$ and updating the parameters $\boldsymbol{A}, \sigma$, and $\boldsymbol{C}$. The mean vector of Gaussian distribution is gained from the predictions of the linear model.

WS-CMA-ES: Warm starting CMA-ES (WS-CMA-ES) [14] uses the evaluated solution set obtained in the optimization of a similar task to initialize CMA-ES for a newly given target task. It uses the best $K_\gamma = \lfloor \gamma K \rfloor$ solutions $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{K_\gamma}$ out of the K solutions obtained in a similar task to compute the promising distribution for the target task as

$$p(\boldsymbol{x}) = \frac{1}{K_\gamma} \sum_{i=1}^{K_\gamma} \mathcal{N}(\boldsymbol{x}_i, \alpha^2 \mathbf{I}) \ , \tag{25}$$

where $\gamma = 0.1$ and $\alpha = 0.1$. The WS-CMA-ES optimizes the target task from a Gaussian distribution that has the smallest Kullback-Leibler divergence from the promising distribution. The mean vector and covariance matrix of this Gaussian distribution are analytically given as follows:

$$\boldsymbol{m}^* = \frac{1}{K_\gamma} \sum_{i=1}^{K_\gamma} \boldsymbol{x}_i \quad \text{and} \quad \boldsymbol{\Sigma}^* = \alpha^2 \mathbf{I} + \frac{1}{K_\gamma} \sum_{i=1}^{K_\gamma} (\boldsymbol{x}_i - \boldsymbol{m}^*)(\boldsymbol{x}_i - \boldsymbol{m}^*)^{\mathrm{T}} \tag{26}$$

The WS-CMA-ES initializes the Gaussian distribution to satisfy $\boldsymbol{m}^{(0)} = \boldsymbol{m}^*$ and $(\sigma^{(0)})^2\boldsymbol{C}^{(0)} = \boldsymbol{\Sigma}^*$. Note that WS-CMA-ES does not consider the existence of context vectors.

Table 1. Definitions of benchmark functions

No.	Name	Definition
1	Sphere	$f_1(\boldsymbol{y}) = \sum_{i=1}^{N} y_i^2$
2	Rosenbrock	$f_2(\boldsymbol{y}) = \sum_{i=1}^{N-1} \left(100(y_{i+1} - y_i^2)^2 + (1 - y_i)^2\right)$
3	Easom	$f_3(\boldsymbol{y}) = -\cos(y_1)\cos(y_2)\exp(-((y_1 - \pi)^2 + (y_2 - \pi)^2)) + 1$

5.2 Experimental Setting

The experimental setting partially followed the setting used in [3]. We used the benchmark functions $f(\boldsymbol{x}; \boldsymbol{\alpha}) = f_i(\phi_j(\boldsymbol{x}; \boldsymbol{\alpha}))$ constructed by applying a context vector dependent transformation $\phi_j(\boldsymbol{x}, \alpha)$ to the existing black-box continuous benchmark function f_i. Table 1 shows the definitions of benchmark functions.

The sphere function is a unimodal benchmark function that is easy to optimize. The Rosenbrock function is an ill-scale and non-separable benchmark function that requires covariance matrix adaptation to fit the function landscape. The Easom function has many shallow local optima and requires starting optimization with initial distribution around the optimal solution. In this experiment, we set the number of dimensions to $N = 20$ for the sphere and Rosenbrock functions, and $N = 2$ for the Easom function. We used three transformations depending on the context vector as follows:

- Linear Shift: $\phi_1(\boldsymbol{x}; \boldsymbol{\alpha}) = \boldsymbol{x} - \boldsymbol{G}\boldsymbol{\alpha}$
- Nonlinear Shift: $\phi_2(\boldsymbol{x}; \boldsymbol{\alpha}) = \boldsymbol{x} - \boldsymbol{G}(\boldsymbol{\alpha} \circ \boldsymbol{\alpha})$
- Noisy Shift: $\phi_3(\boldsymbol{x}; \boldsymbol{\alpha}) = \boldsymbol{x} - \boldsymbol{G}\boldsymbol{\alpha} + \epsilon^2\mathcal{N}$

Each element of constant matrix $\boldsymbol{G} \in \mathbb{R}^{N \times N_\alpha}$ was assigned according to the standard normal distribution $\mathcal{N}(0, 1)$ and shared in the optimizations for the target and previously appeared M_{prev} context vectors. Note that operation $\circ$ is an element-wise product of vectors and $\epsilon = 0.25$. Noise vector $\mathcal{N} \in \mathbb{R}^N$ was generated from the multivariate standard normal distribution $\boldsymbol{\mathcal{N}}(\boldsymbol{0}, \boldsymbol{I})$ every time the context vector was given. For every context, these functions have a unique global minimum whose evaluation value is zero. We set the number of dimensions for the context vector to $N_\alpha = 2$ and the number of optimization results for warm starting to $M_{\text{prev}} = 10$.

Because CMA-ES-CWS uses the past optimization results, we pre-optimized M_{prev} problems corresponding to $M_{\text{prev}} = 10$ context vectors generated on $[-2, 2]^{N_\alpha}$ uniformly at random. For these pre-optimizations, we ran CMA-ES with the initial step-size, and covariance matrix was given by $\sigma^{(0)} = 2$ and $\boldsymbol{C}^{(0)} = \mathbf{I}$. The initial mean vector was given on $[-1, 1]^N$ uniformly at random. The maximum number of evaluations was 1×10^4 for the sphere and Easom functions, and 4×10^4 for the Rosenbrock function. We restarted CMA-ES when the maximum eigenvalue of $\sigma^2\boldsymbol{C}$ was less than 10^{-10} and terminated the optimization when the best evaluation value was less than 10^{-8} or the number of evaluations reached its maximum value.

The contextual CMA-ES was evaluated using the model obtained in pre-optimization. In pre-optimization, the context vector was given on $[-2, 2]^{N_\alpha}$ uniformly at random before generating each solution. For a fair comparison with CMA-ES-CWS, the maximum number of evaluations was $M_{\mathrm{prev}} \times 10^4$ for the sphere and Easom functions and $4M_{\mathrm{prev}} \times 10^4$ for the Rosenbrock function.

For each of the objective functions corresponding to M_{prev} context vectors, the initialization of WS-CMA-ES was performed with the set of solutions that were generated on $[-2, 2]^N$ uniformly at random. For a fair comparison with CMA-ES-CWS, the number of solutions was 1×10^4 for the sphere and Easom functions and 4×10^4 for the Rosenbrock function. To select a similar task from M_{prev} tasks, we measured the Euclid distances between the target and M_{prev} context vectors and selected the task corresponding to the nearest context vector to the target context vector.

We also ran CMA-ES for comparison with CMA-ES-CWS. The initial step-size and covariance matrices were given as $\sigma^{(0)} = 2$ and $\boldsymbol{C}^{(0)} = \mathbf{I}$, and the initial mean vector was given on $[-1, 1]^N$ uniformly at random. Target context vector $\boldsymbol{\alpha}_{\mathrm{new}}$ was given on $[-2, 2]^{N_\alpha}$ uniformly at random. We regarded an optimization as successful when the best evaluation value on the objective function $f(\boldsymbol{x}; \boldsymbol{\alpha}_{\mathrm{new}})$ corresponding to this context vector was less than 10^{-8}. We performed 20 independent trials for each experimental setting.

5.3 Experimental Result

Figures 1 shows the transitions of the best evaluation values on each benchmark function. To evaluate the multi-output GPR in CMA-ES-CWS, we presented the evaluation value for the mean vector of the predictive distribution. The contextual CMA-ES and the multi-output GPR in CMA-ES-CWS do not require an additional optimization for the target context vector. Note that these evaluation values were not plotted for values less than 10^{-8}.

First, comparing the performance of CMA-ES-CWS with those of CMA-ES and WS-CMA-ES, CMA-ES-CWS consistently outperformed these methods under all setting. Particularly on the Rosenbrock function, CMA-ES-CWS reached the target evaluation value with approximately 1/4 of number of the evaluations of CMA-ES and WS-CMA-ES. The WS-CMA-ES exhibited no significant advantage over CMA-ES and increased the upper quartile range for the Easom function. From these results, we confirmed the effectiveness of warm starting in CMA-ES-CWS using the context vector.

Next, comparing the performance of CMA-ES-CWS with that of the contextual CMA-ES, the performance of CMA-ES-CWS in the initial iteration was better than that of the contextual CMA-ES when the nonlinear and noisy shifts were applied. This is because the contextual CMA-ES utilized a linear model whereas CMA-ES-CWS used the multiple-output GPR that can capture nonlinear relationships. With the linear shift, the initial evaluation value of CMA-ES-CWS was significantly worse than these model outputs. This is because the lower bound on the initial step-size in (23) gave a relatively large initial value when compared with the distance between the initial mean vector and optimal

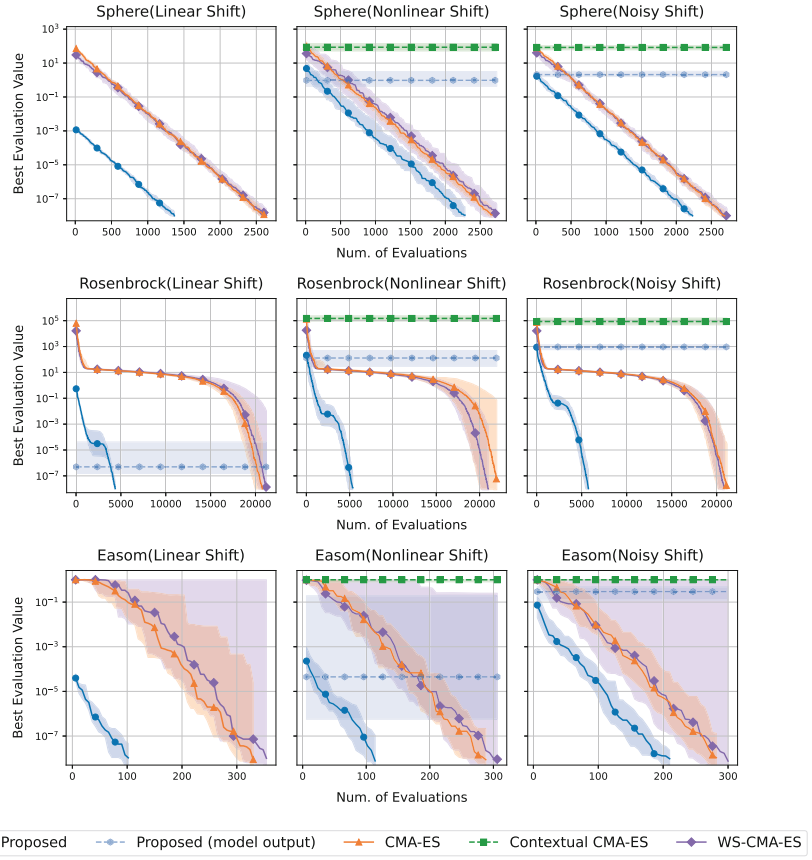


Fig. 1. Transitions of best evaluation values on the sphere, Rosenbrock, and Easom functions. We plot the medians and interquartile ranges. Dash lines show the evaluation values for outputs of the policy model in contextual CMA-ES and multi-output GPR in CMA-ES-CWS. Note that the evaluation values less than 10^{-8} are not plotted.

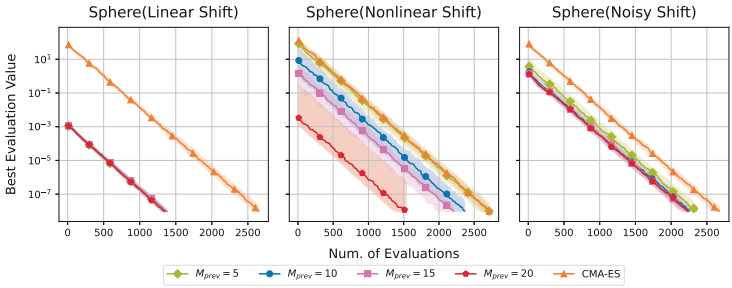


Fig. 2. Transitions of the best evaluation values with various past optimization results M_{prev}. We plotted the medians and interquartile ranges of 20 trials on the sphere function.

solution. On the Rosenbrock function with the linear shift, the evaluation value for the mean vector of the predictive distribution was not less than 10^{-8} because of the failure of pre-optimizations.

Finally, we compared the results of CMA-ES-CWS for various shift types. On the sphere function, the linear shift required fewer evaluations than the nonlinear and noisy shifts. With the Rosenbrock function, the number of evaluations did not change significantly regardless of which transformation was applied. Note that the CMA-ES-based methods must adapt the covariance matrix to fit the landscape of the Rosenbrock function. For all transformations, CMA-ES-CWS achieved the initial distribution that required a small number of evaluations for covariance matrix adaptation on the Rosenbrock function.

5.4 Effect of Number of Pre-optimizations

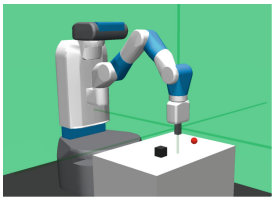
We evaluated the performance of CMA-ES-CWS on the sphere function with various numbers of pre-optimizations $M_{\mathrm{prev}}(M_{\mathrm{prev}} = 5, 10, 15, 20)$. Other experiment settings were the same as in Sect. 5.2. Figure 2 shows the transitions of the best evaluation values. We also plotted the result of the CMA-ES to consider the case where no pre-optimization result was obtained. With the linear shift, the optimization performance did not change regardless of the number of pre-optimizations; because, owing to the lower bound on the initial step-size in (23), the initial step-size was set to a large value such that the performance differences in initial mean vectors were disappeared. When applying the nonlinear shift, the optimization performance improved as the number of pre-optimizations increased. Particularly, with $M_{\mathrm{prev}} = 5$, the optimization performance was the same as that of CMA-ES. For problems with nonlinear dependencies on context vectors, CMA-ES-CWS may be more effective with a larger number of pre-optimization results. Finally, focusing on the results with the noisy shift, although the optimization performance dropped slightly when $M_{\mathrm{prev}} = 5$, almost the same optimization performance was observed with various numbers of pre-optimizations. Because the results with the noisy shift were worse than the results with the linear shift, CMA-ES-CWS cannot deal with the noise even with a large number of pre-optimization results. Developing new noise handling capabilities for warm starting in noisy contextual optimization is a future work.

6 Evaluation Experiment in Robot Control Task

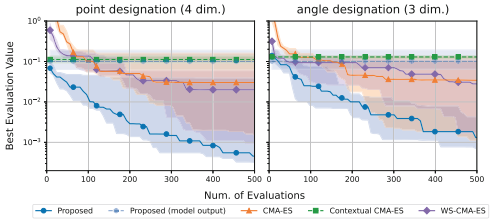
6.1 Experimental Setting

We used two robot control tasks provided by OpenAI Gym [5,16],

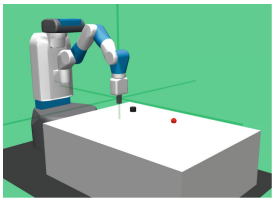
- **FetchPush-v2**: Control the robot arm to push a box to a target position.
- **FetchSlide-v2**: Control the robot arm to slide a box to a target position that is out of reach for the arm.



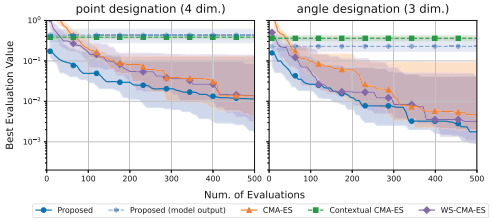
(a) FetchPush-v2



(b) Result of algorithm comparison on FetchPush-v2



(c) FetchSlide-v2



(d) Result of algorithm comparison on FetchSlide-v2

Fig. 3. Images of FetchSlide-v2 and FetchPush-v2 and comparison results

Following reference [15], we considered a trajectory parameter space to reduce the number of dimensions of the problem. The objective of optimization was to designate the parameters that determine the trajectory of the arm. The trajectory comprised two movements: putting the arm close to the box and sliding the arm to move the box. The trajectory was determined using coordinates $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^2$. The first point $\boldsymbol{x}_1$ determined where to put the arm, whereas the second point $\boldsymbol{x}_2$ determined where to slide the arm from the first position to push or slide the box. The maximum amount of movement for each time step was set to (1,1,1), whereas OpenAI Gym set time step to 1/25th of a second. The simulator calculated the arm trajectory with the above limits.

We prepared two trajectory parameter spaces: point and angle designation spaces. In point designation space, there were four design variables that determined points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The ranges of design variables were set as $\boldsymbol{x}_1, \boldsymbol{x}_2 \in [-0.2, 0.2]^2$ for FetchPush and $\boldsymbol{x}_1 \in [-0.2, 0] \times [-0.2, 0.2]$ and $\boldsymbol{x}_2 \in [0, 0.4] \times [-0.4, 0.4]$ for FetchSlide. The angle designation space contained three design variables, point $\boldsymbol{x}_2$, and additional parameter θ that determined the angle for approaching to the box. The point $\boldsymbol{x}_1$ was set on the circle with the radius 0.07 around the initial box position and determined using parameter θ. The ranges of the three design variables were set as $\theta \in [0, 2\pi]$ and $\boldsymbol{x}_2 \in [-0.2, 0.2]^2$ for FetchPush and $\theta \in [0, \pi]$ and $\boldsymbol{x}_2 \in [0, 0.4] \times [-0.4, 0.4]$ for FetchSlide. We used the final distance between the box and the target position as the evaluation value in each task. When the design variable was out of the range, we computed the evaluation value using the design variable clipped into the range and, as a penalty, added the sum of the Euclidean distances from the range to the evaluation value.

In this experiment, we used four-dimensional context vector $\boldsymbol{\alpha} \in \mathbb{R}^4$. The first two dimensions determined the initial position of the box and other dimensions determined the target position. We set the range of the context vector to $[-0.15, 0.15]^4$ for FetchPush and $[-0.1, 0.1]^2 \times [-0.3, 0.3]^2$ for FetchSlide.

The number of pre-optimizations was $M_{\text{prev}} = 10$. The maximum number of evaluations in each pre-optimization was 5×10^2 for CMA-ES-CWS, CMA-ES, and WS-CMA-ES. For a fair comparison, we set the maximum number of evaluations for contextual CMA-ES to $5M_{\text{prev}} \times 10^2$. Other experimental settings were the same as in the experiment using benchmark functions in Sect. 5.

6.2 Experimental Result

Figure 3 shows the transitions of the best evaluation values for the two tasks. Note that these best evaluation values included penalty, and we confirmed that only CMA-ES and WS-CMA-ES included penalty until approximately 100 evaluations, whereas other methods did not include the penalty.

In all settings, CMA-ES-CWS consistently outperformed other methods. Particularly, in the FetchPush task, CMA-ES-CWS significantly performed better than did the other methods. It is evident that CMA-ES and WS-CMA-ES started optimizations from out of the range for the design variables, and these methods consumed evaluations to generate solutions in the range. By contrast, CMA-ES-CWS could start the optimization from in the range for the design variables. We can confirm that the warm starting of CMA-ES-CWS was also effective in robotic tasks.

7 Conclusion

In this study, we proposed CMA-ES-CWS, which incorporates a warm starting for contextual optimization problems. The CMA-ES-CWS uses a multi-output GPR to compute the predictive distribution of the optimal solution corresponding to a newly given context vector from the best solutions corresponding to previously given context vectors. Subsequently, it initializes the sampling distribution of CMA-ES using the predictive distribution. Based on experimental results using benchmark functions, we confirmed that CMA-ES-CWS exhibited better performance than that of the contextual CMA-ES and WS-CMA-ES when applying nonlinear or noisy transformations of the search space. In addition, the CMA-ES-CWS outperformed also the other methods in the robot control tasks. In future, we will extend CMA-ES-CWS to discrete and mixed-integer optimization methods by introducing Bayesian estimation using discrete probability distributions. The development of a reasonable initialization method for the covariance matrix is also planned.

References

1. Abdolmaleki, A., Lau, N., Paulo Reis, L., Neumann, G.: Contextual stochastic search. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, GECCO 2016 Companion, pp. 29–30. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2908961.2909012
2. Abdolmaleki, A., Lau, N., Reis, L.P., Peters, J., Neumann, G.: Contextual policy search for generalizing a parameterized biped walking controller. In: 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, pp. 17–22 (2015). https://doi.org/10.1109/ICARSC.2015.43
3. Abdolmaleki, A., Price, B., Lau, N., Reis, L.P., Neumann, G.: Contextual covariance matrix adaptation evolutionary strategies. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pp. 1378–1385 (2017). https://doi.org/10.24963/ijcai.2017/191
4. Abdolmaleki, A., Simões, D., Lau, N., Reis, L.P., Neumann, G.: Contextual direct policy search. J. Intell. Rob. Syst. **96**(2), 141–157 (2019). https://doi.org/10.1007/s10846-018-0968-4
5. Brockman, G., et al.: Openai gym (2016)
6. Char, I., et al.: Offline contextual Bayesian optimization. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)
7. Deisenroth, M.P., Neumann, G., Peters, J.: A survey on policy search for robotics (2013). https://doi.org/10.1561/2300000021
8. Fabisch, A., Metzen, J.H.: Active contextual policy search. J. Mach. Learn. Res. **15**(97), 3371–3399 (2014). http://jmlr.org/papers/v15/fabisch14a.html
9. Hansen, N.: The CMA evolution strategy: a tutorial. CoRR arxiv:1604.00772 (2016)
10. Hansen, N., Auger, A.: Principled design of continuous stochastic search: from theory to practice. In: Borenstein, Y., Moraglio, A. (eds.) Theory and Principled Methods for the Design of Metaheuristics. NCS, pp. 145–180. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-33206-7_8
11. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 312–317 (1996). https://doi.org/10.1109/ICEC.1996.542381
12. Journel, A., Huijbregts, C.: Mining Geostatistics. Academic Press, London (1978)
13. Kupcsik, A., Deisenroth, M., Peters, J., Neumann, G.: Data-efficient generalization of robot skills with contextual policy search. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 27, no. 1, pp. 1401–1407 (2013). https://doi.org/10.1609/aaai.v27i1.8546
14. Nomura, M., Watanabe, S., Akimoto, Y., Ozaki, Y., Onishi, M.: Warm starting CMA-ES for hyperparameter optimization. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 10, pp. 9188–9196 (2021). https://doi.org/10.1609/aaai.v35i10.17109
15. Pinsler, R., Karkus, P., Kupcsik, A., Hsu, D., Lee, W.S.: Factored contextual policy search with bayesian optimization. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 7242–7248 (2019). https://doi.org/10.1109/ICRA.2019.8793808
16. Plappert, M., et al.: Multi-goal reinforcement learning: challenging robotics environments and request for research (2018)

17. The GPy authors: GPy: a gaussian process framework in python (2012). http://github.com/SheffieldML/GPy
18. Watanabe, S., Awad, N., Onishi, M., Hutter, F.: Speeding up multi-objective hyperparameter optimization by task similarity-based meta-learning for the tree-structured Parzen estimator. In: International Joint Conference on Artificial Intelligence (2023)
19. Zhang, C., Agarwal, A., Iii, H.D., Langford, J., Negahban, S.: Warm-starting contextual bandits: robustly combining supervised and bandit feedback. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 7335–7344. PMLR (2019)

A Potential Function for a Variable-Metric Evolution Strategy

Stephan Frank(✉) and Tobias Glasmachers

Fakultät für Informatik, Institut für Neuroinformatik, Ruhr-Universität Bochum, Bochum, Germany
{stephan.frank,tobias.glasmachers}@ini.rub.de

Abstract. This paper works towards an analysis of a variable-metric evolution strategy by means of drift analysis. Drift analysis has been effective for proving convergence and analyzing the runtime of a simple (1+1)-ES. We make a first step towards including covariance matrix adaptation (CMA). To this end, we develop a novel class of potential functions for the (1+1)-CMA-ES optimizing two-dimensional convex quadratic functions. We leverage invariances to efficiently sample a representative space of states. We use simulations to gain an empirical estimate of the expected minimal drift induced by the candidate potential function and to tune potential function parameters. Our results indicate that the tuned potential function is negative and uniformly bounded away from zero, which yields linear convergence.

1 Introduction

Variable metric evolution strategies like the covariance matrix adaptation evolution strategy (CMA-ES) [8,9,11] are among the best performing methods for difficult black-box optimization problems [3,7]. However, due to their randomized nature and the lack of convergence guarantees, they are sometimes considered unreliable heuristics. The lack of analytical convergence guarantees can be a barrier to the adoption of state-of-the-art methods like CMA-ES. Even though the developers of such algorithms have a good understanding of how the methods will perform on a problem, it can be difficult to convey this understanding to practitioners. We therefore believe that developing theoretical performance guarantees is an important line of research. On that route, we pursue theory-guided empirical analysis as an intermediate goal.

Runtime analysis of evolutionary algorithms is a well-established field [16]. It is very well developed for optimization in discrete domains, where nearly all recent results were established by means of drift analysis techniques [12,13]. The desire to understand the optimization behavior of evolution strategies is not new [4]. In recent years, there was significant progress in transferring drift techniques to the analysis of continuous optimization [1,2,5,10,14,15]. We witnessed an impressive generalization in terms of problems, starting from the simple sphere function and arriving at large function classes like all strongly convex functions

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 221–235, 2024.
https://doi.org/10.1007/978-3-031-70068-2_14

with Lipschitz gradient. However, in terms of algorithms, only rather simplistic evolution strategies without covariance matrix adaptation (CMA) were analyzed. While being quite flexible in principle, the apparent challenge of the drift-based approach is to identify a suitable potential function. The present paper is concerned with the question of how to design a suitable potential for a variable metric ES.

In the present paper, we aim to make progress towards analyzing variable-metric evolution strategies by means of drift. We believe that analyzing convergence through empirical means and the help of drift analysis can be valuable in addition to the traditional method of using analytical proofs. The natural first step in the analysis is the quest for a Lyapunov potential capturing the quite involved algorithm dynamics sufficiently well. We propose to address the problem of designing a suitable potential function with an iterative method based on empirical analysis. Our approach can lead to a better understanding of potential functions and guide the development of analytical proofs, by allowing for piecewise advancements on potential functions, supported by empirical performance data. The approach can be adapted rather easily to a wide range of algorithms and objective functions, expanding the range of problems for which runtime and convergence guarantees can be established.

Algorithms. Our general methodology is not bound to a specific algorithm. It can hence be applied, e.g., to a fully fledged state-of-the-art implementation of CMA-ES. Instead, we use a simplified version of the (1+1)-CMA-ES [9] as outlined in Algorithm 1. This is a natural choice if we wish to leverage existing results, since the literature on analyzing evolution strategies with drift is focused on elitist selection algorithms.

Algorithm 1: Simplified variant of (1+1)-CMA-ES

Input: $d \in \mathbb{N}$, $f : \mathbb{R}^d \to \mathbb{R}$, $m \in \mathbb{R}^d$, $\sigma > 0$, $c_{\text{cov}} \in (0, 1]$
$p_{\text{target}} = \frac{2}{11}$
while *stopping condition not met* **do**
  $z \sim \mathcal{N}(0, C)$
  $x \leftarrow m + \sigma \cdot z$
  if $f(x) \leq f(m)$ **then**
    $m \leftarrow x$; $p_{\text{succ}} \leftarrow 1$; $C \leftarrow (1 - c_{\text{cov}}) \cdot C + c_{\text{cov}} \cdot (Az)(Az)^T$
  else
    $p_{\text{succ}} \leftarrow 0$
  $\sigma \leftarrow \sigma \cdot \exp\left(\frac{1}{d} \cdot \left(\frac{p_{\text{succ}} - p_{\text{target}}}{1 - p_{\text{target}}}\right)\right)$

Contributions. In this paper, we make the following contributions:

- We propose an experimental methodology supporting the design of a potential function capturing the dynamics of a variable-metric evolution strategy.
- We introduce the target step size of the (1+1)-CMA-ES as a key concept for constructing a suitable potential function.
- Based on the target step size, we define a potential function.
- We provide systematic empirical evidence for the suitability of the novel potential function. At the same time, we are in the position to highlight its weak spots, which might need to be addressed in future work.

Taking the above together, we provide a drift potential function that *potentially* can give rise to an analysis of the optimization behavior of a variable metric ES.

2 Theoretical Background

This section introduces the necessary background. First, we give a brief bird's eye introduction to drift analyses and its challenges. We then turn to invariance properties of CMA-ES, which are instrumental to the design of a Lyapunov potential.

2.1 Drift Analysis

Drift analysis goes back to Hajek [6]. It was later adapted to the specific needs of runtime analysis of randomized search heuristics [12,13]. The general idea of a drift theorem is to connect a statement about the expected single-step reduction of a potential function to the expected number of steps it takes to reduce the potential to a target value. Drift is a powerful concept, since the analysis is reduced to statements about the single-step behavior of the algorithm. Furthermore, drift theorems are not limited to expected values – they can also bound quantiles and hence control the tails of the runtime distribution. For further details, we refer the interested reader to [12,13].

Applying a drift argument amounts to the following steps: we define a potential function ϕ, show that the algorithm exhibits a certain type of expected progress with respect to that potential, and apply a drift theorem to turn the stepwise progress into a runtime bound. Let S denote the state space of the algorithm, s_t the sequence of algorithm states, and $\phi : S \to \mathbb{R}$ the potential. In the simplest case, the expected progress $\mathbb{E}[\phi(s_{t+1}) - \phi(s_t) | s_t = s]$ is bounded from below by a negative constant $-b$. If the progress is also bounded (or its tails controlled in a suitable way), then the so-called additive drift applies, yielding an expected runtime of $\mathbb{E}[T] \leq \frac{a}{b} + \text{const}$, where $a = \phi(s_0) - \phi_{\text{target}}$ is the potential difference to be crossed and T is the so-called first hitting time of the event $\phi(s_T) \leq \phi_{\text{target}}$.

Defining a suitable potential function is an art, not a science. The job of the potential is to capture progress of the algorithm across the whole state space S. For an evolution strategy, this is a non-trivial task because even elitist algorithms make nearly no progress if the step size is either much too small or much too

large, or if the covariance matrix is unsuitable. Then with high probability, the algorithm adapts its distribution parameters towards more suitable values, hence bringing them closer to the regime where progress towards the optimum is achieved. Strategy adaptation does not yield immediate progress in terms of objective function improvement, but rather in terms of the potential to achieve such improvements in the future. Therefore, a suitable potential function needs to capture not only the goal of minimizing the objective function when the step size is well adapted, but also the goal of adapting step size and covariance matrix towards a regime where this is the case. We will discuss a corresponding potential function design in Sect. 3.

Compared with a simple (1+1)-ES, this task is considerably harder when covariance matrix adaptation is involved. The ability to adapt the covariance matrix has the benefit of gaining invariance to affine transformations, which yields more general results in terms of the class of objective functions covered. The price to pay is that it takes away some symmetries, which increases the dimension of the normalized state. Moreover, CMA interacts with step size adaptation in non-trivial ways. However, depending on the tightness of the resulting bound, we may or may not need to capture all of these dependencies in a potential function.

2.2 Invariances

The goal of this section is to reduce the dimension of the state space. We will describe the reduced space by means of a normal form with easy-to-interpret state variables. The reduction also makes sampling a grid of states feasible.

We consider the (1+1)-CMA-ES with parameters (m, C) of its multi-variate Gaussian sampling distribution $\mathcal{N}(m, C)$,[1] optimizing an objective function $f : \mathbb{R}^d \to \mathbb{R}$. The parameters (m, C) and the objective function f define a state of the algorithm, in the sense that this information determines the distribution of successor states. Therefore, we pack them into the tuple $\theta = (m, C, f)$. Given a state θ, we denote the state after a single iteration of (1+1)-CMA-ES as $\theta' = (m', C', f)$. The following definition captures the invariance properties of the (1+1)-CMA-ES algorithm:

Definition 1. *We say that two tuples $\theta_1 = (m_1, C_1, f_1)$ and $\theta_2 = (m_2, C_2, f_2)$ are equivalent, and we write $\theta_1 \sim_T \theta_2$, if there exist an affine transformation $T(x) = Ax + b$ and a strictly monotonically increasing function $h : \mathbb{R} \to \mathbb{R}$ such that it holds*

1. $m_2 = T(m_1) = Am_1 + b$, *(affine invariance, mean)*
2. $C_2 = A^T C_1 A$, *(affine invariance, covariance matrix)*
3. $f_2 = h \circ f_1 \circ T^{-1}$. *(strictly monotone fitness transformations)*

[1] Under slight misuse of notation, we incorporate the step size into the covariance matrix at this point, writing C instead of $\sigma^2 C$ from now on. The parameter σ is re-introduced in the normal form, see equation (1).

In other words, given a representative $\theta = (m, C, f)$ of an equivalence class, then all other members of that class are of the form $(Am + b, A^T CA, h \circ f \circ T^{-1})$. The following lemma clarifies how the definition relates to invariance:

Lemma 1. *Consider a sequence of points $x_1, \ldots, x_n$ ordered by their f_1-ranking: $f_1(x_1) \leq f_1(x_2) \leq \cdots \leq f_1(x_n)$. Then the transformed points $y_1 = T(x_1), \ldots, y_n = T(x_n)$ have an equivalent f_2-ranking, i.e., it holds $f_2(y_1) \leq f_2(y_2) \leq \cdots \leq f_2(y_n)$.*

Proof. The proof amounts to plugging the definition into the formulas of the lemma. We obtain $f_2(y_k) = f_2(T(x_k)) = h(f_1(T^{-1}(T(x_k)))) = h(f_1(x_k))$ and we note that h does not change the ranking. □

Plugging properties (1) and (2) of the definition into the PDF of the multivariate normal distribution yields the same result as the transformation theorem for densities applied to T. Hence, the PDFs are simply transformed into each other by means of T. Together with the above lemma, this implies that performing a step from θ_1 to θ_1' and the three-step sequence of transforming θ_1 into θ_2, performing a step from θ_2 to θ_2', and finally transforming θ_2' back into θ_1', yield the same result if the same randomness is used, and the same distributions in any case. In short: if we understand the algorithm dynamics starting in θ_1 then the insight immediately transfers to θ_2 and to the whole equivalence class.

We use this notion of invariance in two different ways, namely to formulate a potential that respects invariances, and for efficient sampling. In both cases, the goal is to reduce the number of cases for the subsequent analysis. This is achieved by analyzing only one state per equivalence class.

From now on, we consider a general convex quadratic objective, unction $f(x) = \frac{1}{2}(x - x^*)^T H(x - x^*) + c$ with optimizer x^*, Hessian H, and optimal value c. This case is of great interest, since it is a second order approximation of a local optimum of a twice continuously differentiable objective function. The first application of invariance is to simplify f. We can set $c = 0$ since the offset does not impact the ranking. Setting $T(x) = Ax + b$ with $A = H^{-1/2}$ and $b = -Ax^*$, we see that (m, C, f) is equivalent to $(Am + b, ACA^T, s)$, where $s(x) = \frac{1}{2}\|x\|$ is the sphere function. In other words, in order to understand the optimization behavior of a variable-metric evolution strategy on all convex quadratic objective functions, it suffices to consider the sphere function, as long as we consider general initial conditions. This greatly simplifies the task of designing a drift potential.

The remaining invariances are known as rotation and scale invariance. They refer to transformations for which A is a scaled orthogonal matrix and b is zero. Since the objective function is fixed, we write $\theta = (m, C)$ in the following, with Θ forming the space of all algorithm states. For the two-dimensional case, we define a section through the quotient space $\Theta/\sim_T$. Equivalently, it can be considered a normal form for states. First of all, we use the scaling degree of freedom to turn m into a unit vector. We then use the rotation degree of freedom to diagonalize the covariance matrix C. By swapping the axes of the coordinate system and flipping the axes individually, we can transform m to the form $m = (\cos(\alpha), \sin(\alpha))$ with

$\alpha \in [0, \pi/2]$. Furthermore, we can decompose the (diagonal) covariance matrix into scale and shape components. Hence, without loss of generality, we can write all relevant states in the form

$$m = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix} \qquad C = \sigma^2 \begin{pmatrix} \kappa & 0 \\ 0 & \frac{1}{\kappa} \end{pmatrix} \tag{1}$$

with *normalized step size* $\sigma > 0$, *approach angle* $\alpha \in [0, \pi/2]$, and *eccentricity* $\kappa \geq 1$ encoding the shape of the distribution. An algorithm for transforming any state into the normal form is found in the online supplement[2].

3 Construction of a Lyapunov Potential Function

To construct a potential function that yields drift everywhere in the state space, each part of the algorithm's actions needs to be rewarded. By design (elitism), the distribution mean never moves away from the optimum. This ensures that the algorithm will never lose progress in terms of distance to the optimum. However, for increasingly bad parameter settings (too small or large σ or too large κ) the progress rate quickly approaches zero. In order to avoid vanishing drift, we want the potential function to account for the quality of the strategy parameters. We design building blocks for such a function in the following.

3.1 Target σ

For each setting of α and κ we define a *target* σ. The meaning of the target is that the algorithm adjusts the parameter towards this value while (artificially) keeping the other parameters fixed. This does not mean that the target value is optimal in terms of optimization progress; instead it reflects the adaptation actually performed by the algorithm. We denote the target as a function $\sigma^*(\alpha, \kappa)$. We omit the parameters in the following.

3.2 Potential Function

Our potential function is defined in terms of the normal form:

$$V(\theta) = V(m, \kappa, \sigma) = \underset{(1)}{\log(||m||)} + \underset{(2)}{v_1 \cdot |\log(\kappa)|} + \underset{(3)}{v_2 \cdot f_\kappa\left(\left|\log\left(\frac{\sigma}{\sigma^*}\right)\right|\right)}$$

$$f_\kappa(x) = \begin{cases} 1 - \exp\left(-s \cdot \log\left(\frac{x}{w_\kappa}\right)^2\right) & \text{if } x \geq w(\kappa), \quad s = \frac{1}{100}, w_\kappa = 1 + \frac{\log(\kappa)}{10} \\ 0 & \text{else} \end{cases}$$

[2] https://github.com/RUB-INI-Theory-of-Machine-Learning/EmpiricalDriftAnalysis/blob/main/Empirical_Drift_Analysis___Supplements.pdf.

Here, $v_1, v_2 > 0$ are tuning parameters to be determined later. There are three dimensions in which the CMA-ES can make progress, namely the distance to the optimum m, the eccentricity of the matrix κ, and the (normalized) step size σ. Those dimensions are taken care of in the potential function separately:

m: The term $\log(||m||)$ describes optimization progress by finding a point that is closer to the optimum. For "well-adapted" parameters of the algorithm, it should make significant expected progress in this sense. We refer to this term as the $\log(||m||)$-*term*.

κ: The second term of the potential function determines progress by adjusting the eccentricity κ towards the minimal eccentricity of 1, encoding an isotropic distribution. We refer to this term as the κ-*term*.

σ: The third term of the potential function measures progress by adjusting the step size σ towards the stable step size σ^*. The term should become dominant if σ is far away from σ^*. We refer to this term as the σ-*term*. The activation function f_κ asymptotically approaches the identity, but it is flat in a neighborhood of zero, with a differentiable transition (see Fig. 1).

4 Experiments

This section describes the experimental setup to gain empirical data on two types of quantities: the target step size and a lower bound of the expected drift of the potential function.

4.1 Target Step Size

To obtain values for the target step size σ^*, we conduct the following experiment: First we prepare a lattice of 64 linearly spaced α-values from 0 to $\pi/2$ and 2048 geometrically spaced κ-values from 1 to 2000. Then, for every point on the lattice, we transform the normal form into the parameter form (σ is set to 1 in the beginning) and initialize the algorithm with this state. We then conduct a step of the algorithm and transform the resulting state back into the normal form, however, omitting the rescaling to $|m| = 1$, and resetting α and κ to their initial value. After 50,000 iterations for reaching the limit distribution we record σ for further 1,000,000 iterations. In the end we compute the geometric mean of the recorded values. We end up with a lattice of target step sizes σ^*. For parameters that are not on the grid, we compute σ^* with bilinear interpolation.

4.2 Drift Experiments

We perform three experiments, one where the focus lies on understanding the drifts of the potential function around sensible values for (α, κ, σ) and the other two, with a much wider but less dense grid for investigating the potential function's boundary behaviour, i.e., when σ or κ approach ∞ or σ approaches 0.

Sampling Parameters. We define a grid for each parameter (α, κ, σ) and combine those into a three-dimensional product grid. The parameters for the dense grid are shown in Table 1 and for the wide grids (κ-grid, σ-grid) in Table 2. We then take each of these states and perform a Monte-Carlo simulation to obtain the expected value of the drift at that point.

Table 1. Dense Grid κ

param.	range	steps	spacing
α	[0, $\pi/2$]	24	linear
κ	[1, 10]	128	logarithmic
σ	[0.1, 10]	256	logarithmic

Table 2. Wide Grids (κ-grid, σ-grid)

param.	range	steps (κ/σ)	spacing
α	[0, $\pi/2$]	12/12	linear
κ	[1, 1000]	512/24	logarithmic
σ	[0.01, 100]	24/512	logarithmic

Significance and Precision. Since we work with a fixed number of random samples to estimate the drift, we are interested in the quality of those estimations. To that end we perform a one-sided t-test. Let d denote the (observed) population mean. For a fixed candidate precision ϵ, the null hypothesis is that the distributions of d and $d + \epsilon$ are overlapping, while the alternative hypothesis is that $d + \epsilon$ is significantly larger than d. We then perform a golden-section search to find a value of ϵ where the t-test returns a p-value of at most 0.001. This establishes a 99.9% confidence that the observed drifts are not smaller than the precision ϵ. However, due to multiple testing, this does not imply that the overall confidence is 99.9%. In the dense grid run, we evaluate 786,432 samples, out of which we would statistically expect the null hypothesis to be true for approximately 787.

Potential Function Parameters. To determine the optimal weights v_1, v_2 of the second and third term in the potential function we used CMA-ES. We used the sum of the experimentally obtained drift-values and the precision-values as drift-values and optimized for the largest drift-value to be minimal, i.e., for the gap between uniform drift and zero to be maximal.

5 Results

In this section, we show our empirical results. We start with the target step size σ^*. Following that, we analyze the results of the dense grid dataset (Table 1). Finally, we present the results of the wide grid experiments (Table 2) to examine the boundary behaviour.

5.1 Target Parameters

In Fig. 2 we present the target step size σ^* for 64 linearly spaced values of α between 0 and $\pi/2$. We notice significant differences with respect to the approach angle α, and a decreasing trend for growing eccentricity κ.

5.2 Drift Analysis

We observe that the difference between close α values is small, indicating that the drift is continuous. Furthermore, it is also monotonic for the most part. For that reason, we will only present the extreme cases of $\alpha = 0$ and $\alpha = \pi/2$ in the following figures, as this provides a sufficient impression of the results. The plots do not include the precision values added to them, since these are always at least two orders of magnitude smaller than the drift values. The plots of the full data as well as the precision values can be examined in the supplementary material.[3]

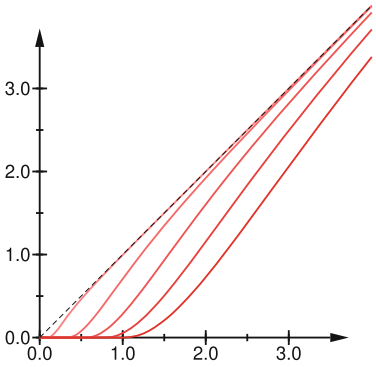


Fig. 1. Graphs of the "activation" function f_κ for $\kappa \in \{1, 10, 100, 1000, 10000\}$, from left to right.

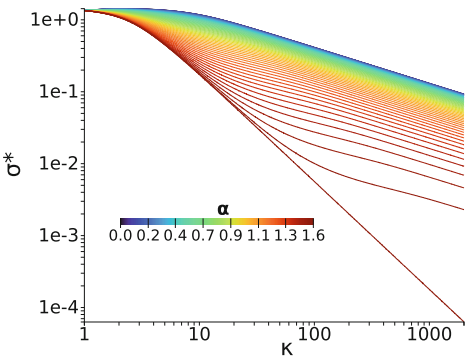


Fig. 2. Graphs of Target Sigma, as a Function Functions of the Remaining Parameters.

Figure 3 shows the legend used for the following heatmap plots. The color scale of each plot is scaled to the minimum and maximum drift for that plot. A blue color indicates negative values, i.e., positive drift (desirable), while red color indicates positive values, i.e., negative drift (undesirable). Values close to 0 appear white.

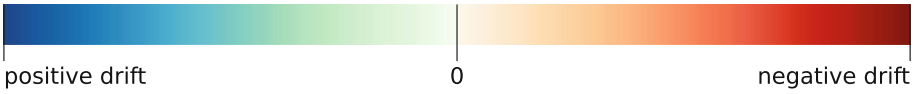
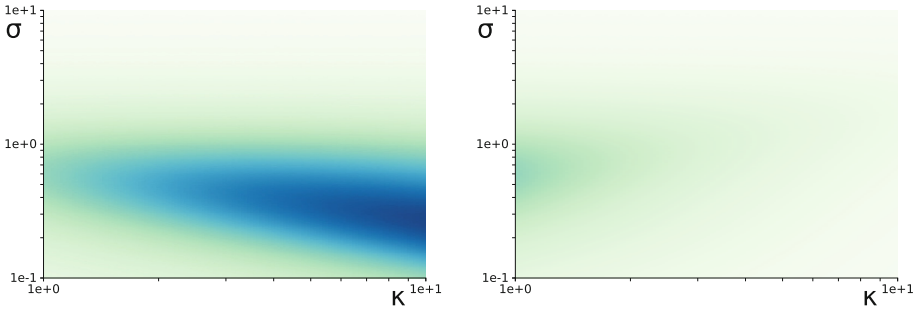


Fig. 3. The Heatmap Scale

[3] https://github.com/RUB-INI-Theory-of-Machine-Learning/EmpiricalDriftAnalysis/tree/main/plots.

Weights. The resulting weights from the CMA-ES optimization for the drift terms differ for each experiment, however $v_1 = 1.7$ and $v_2 = 3.14$ prove to yield good results across all datasets. All following plots use these weights.

The Log(||m||)-Term-Optimization Progress. We first present the result of the overall drift from the dense grid run. We see that for $\alpha = 0$ the overall drift is positive. Furthermore, for large values of κ we notice that for $0.05 < \sigma < 1$ a region with especially large drift emerges. For $\alpha = \pi/2$ we also see an overall positive drift with slightly larger drift values for very small κ values and $0.3 < \sigma < 1$. The minimal drift (gap to zero) in this dataset is ≈ -0.00198 (Fig. 4).



(a) $\alpha = 0$ (b) $\alpha = \pi/2$

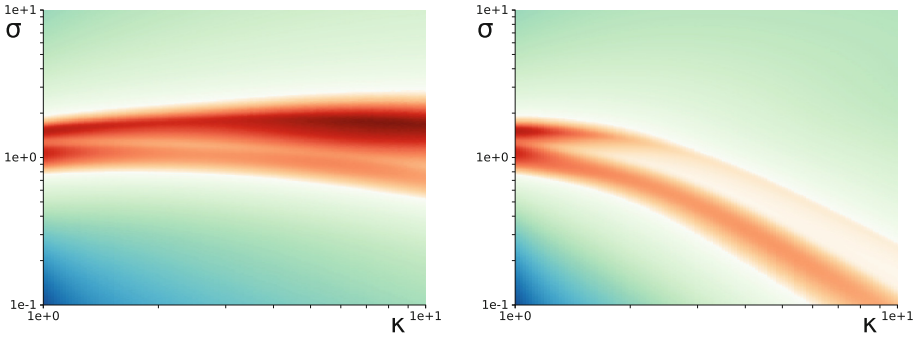
Fig. 4. The log($||m||$)-term Progress

However, we notice for both α values and for increasing σ values decreasing drift, while for very small σ the drift diminishes. Since the goal of (additive) drift analysis is to find a potential function that provides a lower bound on the drift, this trend defeats that purpose. Therefore we will now look at the results where the σ-term of the potential function is added.

Adding the σ-term – Behaviour on the Boundaries. The σ-term of the potential function rewards progress for changing the σ value towards σ^*. Figure 5 shows the drift of the σ-term in isolation.

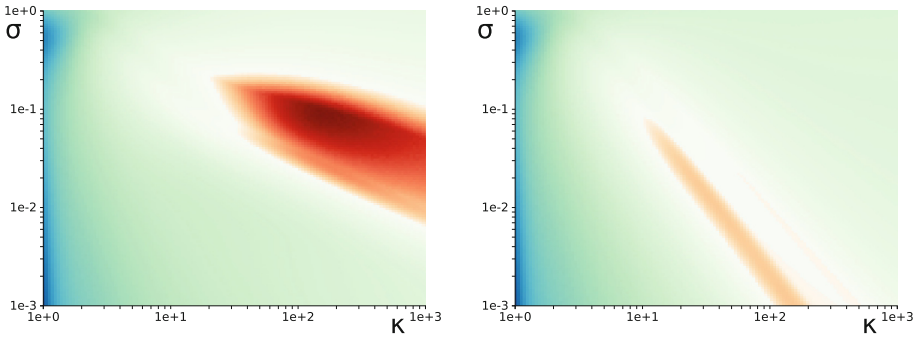
We notice a large red strip in the results where the drift is negative. This corresponds to the target σ values. When the algorithm's σ parameter is already at or very close to the target value σ^* then any change results in negative drift.

The negative drift around σ^* stems from a *moving target* problem. Even though the algorithm adapts its σ parameter towards σ^*, in the new algorithm state that value has changed (because α and κ were adapted by the algorithm). This effect is present everywhere, however in the cases where the drift becomes negative this effect is so strong that it dominates the otherwise favourable σ adaptation of the algorithm. We used a filter f_k in this term to alleviate this effect. Although the effect is still present, it produces far less negative drift than without the filter.



(a) $\alpha = 0$ (b) $\alpha = \pi/2$

Fig. 5. Sigma Progress

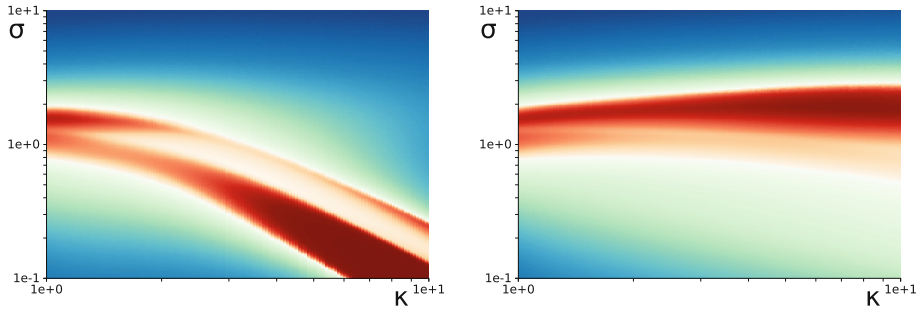


(a) $\alpha = 1.5404$ (b) $\alpha = \pi/2$

Fig. 6. Drift of the $\log(||m||)$-term and the σ-term combined. Note that this graph shows a larger grid to present the problematic areas.

Besides that, we also witness that for large values of σ the σ-term shows a stable drift. In Fig. 6 we see the combined drift of the $\log(||m||)$-term and the σ-term. In Fig. 7 we plotted the influence of the σ term on the overall drift. We notice that for large σ the third term dominates the overall drift. This is what we hoped to see, as this counteracts the diminishing drift of the $\log(||m||)$-term. This also makes sense from the perspective of an ES, since the search distribution is misaligned and needs to improve. The same effect can be observed for small σ in the bottom left corner, where the moving target problem does not dominate the drift. In Fig. 6a and 6b, we see that for large α, large κ and small σ the drift from the first term is not large enough to make up for the negative drift of the third term. Because of that the κ-term is necessary, which we will add next.

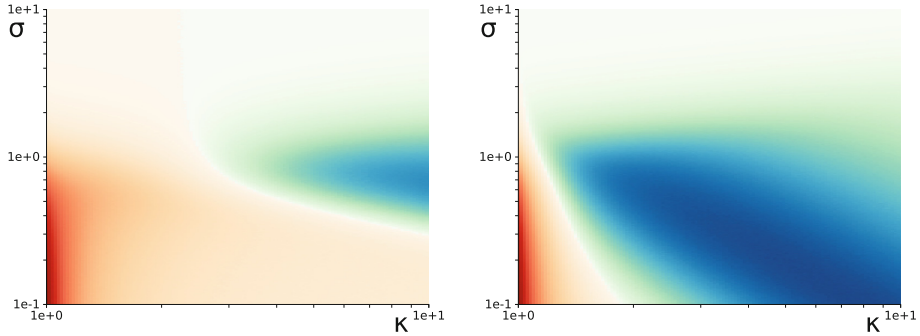
Adding the κ-term – Fixing Moving Targets. In Fig. 8 we present the drift for the κ-term in isolation. Similar to the negative drifts for the σ-term, there also exist regions with negative drift. Since this term only gratifies the eccentric-



(a) $\alpha = 0$ (b) $\alpha = \pi/2$

Fig. 7. The Sigma Influence in a Percentile View. The influence is the drift of the σ-term divided by the sum of the absolute values of the $\log(||m||)$-term and the σ-term.

ity to become smaller, naturally there are configurations where the algorithm has negative drift. This is due to the fact that for small α and reasonable σ the algorithm is making better progress by becoming more eccentric. When κ becomes exceedingly large the algorithm does not profit from eccentricity anymore and is inclined to make κ smaller. For large α this point is reached for smaller κ. Furthermore, when $\kappa = 1$ the algorithm can only get worse. This corresponds the red regions on the bottom left.

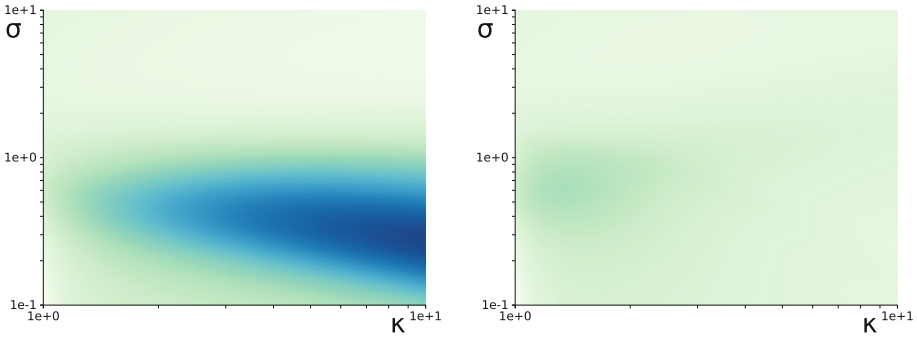


(a) $\alpha = 0$ (b) $\alpha = \pi/2$

Fig. 8. Kappa Progress

The Final Result-Adding It All Together. We now add the κ term to see the drift of the complete potential function, which is shown in Fig. 9. For small α we continue to see a region with strong drift for larger κ and $\sigma < 1$. Overall we see a moderate amount of drift everywhere, and no regions with negative drift. The minimal drift value for the complete potential function is ≈ -0.0063 which

is three times as much as the minimal drift value of just the $\log(||m||)$-term in the same region of the state space.

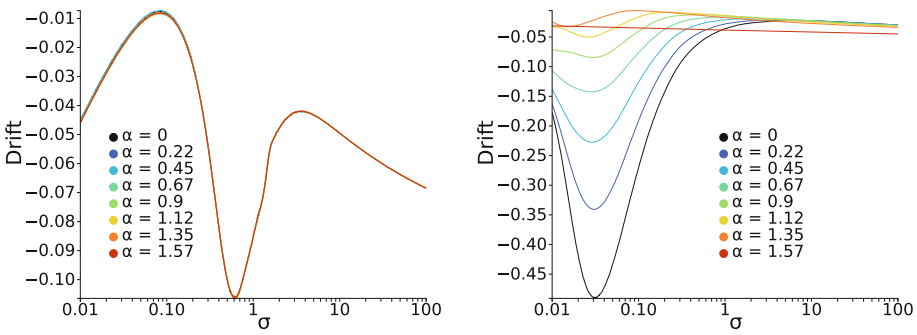


(a) $\alpha = 0$ (b) $\alpha = \pi/2$

Fig. 9. The Drift of the Complete Potential Function $V(\theta)$

5.3 Asymptotic Behaviour

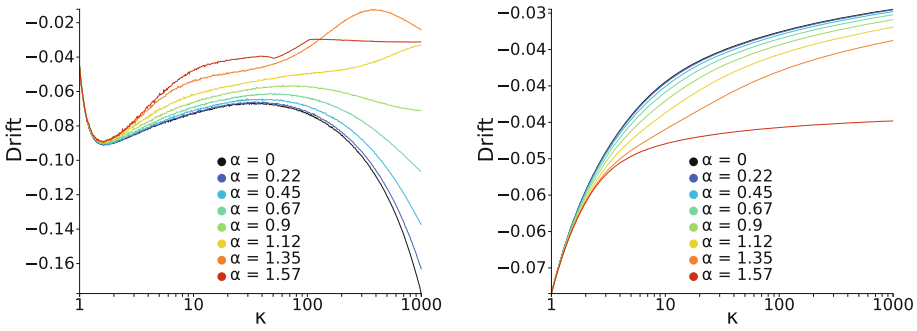
Even though we observed an improved drift for extreme values of κ and σ, there is still a small decrease visible as the parameters become more extreme. We believe that this trend will saturate such that we can guarantee a lower bound for the drift. In Figs. 10 and 11 we present the result of the experiments that probe deeper into the parameter space (Table 2).



(a) $\kappa = 1$ (b) $\kappa = 1000$

Fig. 10. The Drift for a Single κ along the σ Axis

We leave Fig. 10 for visual inspection to the reader. However, we believe to identify the asymptotic behaviour of the drift. Even though in Fig. 10b towards smaller σ the drift is decreasing, we suspect that if the experiment had an even wider grid, we would observe a behaviour similar to the one in Fig. 10a.



(a) $\sigma = 0.01$ (b) $\sigma = 100$

Fig. 11. The Drift for a Single σ along the κ Axis

Figure 11a displays increasing drifts for most α and at least stable drifts for some α, while Fig. 11b shows decreasing drifts at $\sigma = 100$ and at least suggests an asymptotic behaviour towards large κ. A more detailed investigation of asymptotic effects needs better noise handling, and will be subject to future work.

6 Conclusion

We have filled a gap in the analysis of evolution strategies by proposing a drift potential function for a variable metric evolution strategy. In general, designing a potential for drift analysis is a difficult task. Our function involves an auxiliary function, namely target states of the scale parameter σ of the search distribution, which is interesting to investigate in its own right. Our empirical analysis shows that the novel potential works well in the sense that it yields negative drift everywhere, and that it is bounded away from zero.

Naturally, our result has limitations. We consider only two-dimensional search spaces since our sampling-based approach scales badly to higher dimensions. While Monte Carlo simulations leave space for random effects, our huge sample size yields high confidence. Inter- and extrapolation from a fixed parameter grid may induce inaccuracies. However, the smoothness of all observed effects indicates that the grid is well-chosen, and that our results do indeed generalize—at least qualitatively—to the full continuous and unbounded state space.

Although our study is empirical in nature we believe that it can serve three distinct goals: it increases the trust into the reliability of variable metric evolution strategies like CMA-ES, it enhances our understanding of their behavior, and it paves the way for an actual mathematical convergence proof based on drift arguments.

References

1. Akimoto, Y., Auger, A., Glasmachers, T.: Drift theory in continuous search spaces: expected hitting time of the (1+1)-ES with 1/5 success rule. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 801–808 (2018)
2. Akimoto, Y., Auger, A., Glasmachers, T., Morinaga, D.: Global linear convergence of evolution strategies on more than smooth strongly convex functions. SIAM J. Optim. **32**(2), 1402–1429 (2022)
3. Bennet, P., Doerr, C., Moreau, A., Rapin, J., Teytaud, F., Teytaud, O.: Nevergrad: black-box optimization platform. ACM SIGEVOlution **14**(1), 8–15 (2021)
4. Beyer, H.-G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001). https://doi.org/10.1007/978-3-662-04378-3
5. Correa, C.R., Wanner, E.F., Fonseca, C.M.: Lyapunov design of a simple step-size adaptation strategy based on success. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 101–110. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45823-6_10
6. Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. Adv. Appl. Probab. **14**(3), 502–525 (1982)
7. Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 1689–1696 (2010)
8. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
9. Igel, C., Suttorp, T., Hansen, N.: A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), vol. 1, pp. 453–460 (2006)
10. Jägersküpper, J.: Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. Theoret. Comput. Sci. **379**(3), 329–347 (2007)
11. Kern, S., Müller, S.D., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms-a comparative review. Nat. Comput. **3**(1), 77–112 (2004)
12. Lehre, P.K., Witt, C.: General drift analysis with tail bounds. arXiv preprint arXiv:1307.2559 (2013)
13. Lengler, J.: Drift analysis. In: Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 89–131 (2020)
14. Morinaga, D., Akimoto, Y.: Generalized drift analysis in continuous domain: linear convergence of (1+ 1)-ES on strongly convex functions with lipschitz continuous gradients. In: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, pp. 13–24 (2019)
15. Morinaga, D., Fukuchi, K., Sakuma, J., Akimoto, Y.: Convergence rate of the (1+ 1)-evolution strategy on locally strongly convex functions with lipschitz continuous gradient and their monotonic transformations. Technical Report arXiv:2209.12467, arXiv.org (2022)
16. Oliveto, P.S., Yao, X.: Runtime analysis of evolutionary algorithms for discrete optimization. In: Theory of Randomized Search Heuristics: Foundations and Recent Developments, pp. 21–52. World Scientific (2011)

CMA-ES for Discrete and Mixed-Variable Optimization on Sets of Points

Kento Uchida[1(✉)], Ryoki Hamano[2], Masahiro Nomura[2], Shota Saito[1,3], and Shinichi Shirakawa[1]

[1] Yokohama National University, Yokohama, Japan
{uchida-kento-fz,shirakawa-shinichi-bg}@ynu.ac.jp, saito-shota-bt@ynu.jp
[2] CyberAgent, Inc., Shibuya, Japan
{hamano_ryoki_xa,nomura_masahiro}@cyberagent.co.jp
[3] SKILLUP NeXt Ltd., Chiyoda, Japan

Abstract. Discrete and mixed-variable optimization problems have appeared in several real-world applications. Most of the research on mixed-variable optimization considers a mixture of integer and continuous variables, and several integer handlings have been developed to inherit the optimization performance of the continuous optimization methods to mixed-integer optimization. In some applications, acceptable solutions are given by selecting possible points in the disjoint subspaces. This paper focuses on the optimization on sets of points and proposes an optimization method by extending the covariance matrix adaptation evolution strategy (CMA-ES), termed the CMA-ES on sets of points (CMA-ES-SoP). The CMA-ES-SoP incorporates margin correction that maintains the generation probability of neighboring points to prevent premature convergence to a specific non-optimal point, which is an effective integer-handling technique for CMA-ES. In addition, because margin correction with a fixed margin value tends to increase the marginal probabilities for a portion of neighboring points more than necessary, the CMA-ES-SoP updates the target margin value adaptively to make the average of the marginal probabilities close to a predefined target probability. Numerical simulations demonstrated that the CMA-ES-SoP successfully optimized the optimization problems on sets of points, whereas the naive CMA-ES failed to optimize them due to premature convergence.

Keywords: CMA-ES · discrete optimization · mixed-variable optimization · adaptation

1 Introduction

Mixed-variable optimization methods have been actively developed due to the significant demand in real-world applications. Most of the existing works have focused on mixed-integer optimization problems that contain both continuous and integer variables. One of the major approaches is applying integer handlings

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 236–251, 2024.
https://doi.org/10.1007/978-3-031-70068-2_15

to powerful continuous optimization methods to address integer variables [9,10,14]. For example, the reference [9] focused on the covariance matrix adaptation evolution strategy (CMA-ES) [13] and proposed the CMA-ES with margin by incorporating the margin correction. The margin correction uses a margin that is a lower bound of the marginal probabilities for integer variables and prevents premature convergence caused by the original update of the CMA-ES. These integer handlings consist of coordinate-wise operations for each integer variable.

However, because the integer handling assumes the set of possible values are given on grid space, they cannot be applied to other kinds of sets of possible values. For example, when optimizing the location for the construction of wind turbines [2], the user makes a set of possible locations (pairs of latitude and longitude) and requests an optimizer to select the best location from the set. In this case, the existing integer handling cannot be applied. In addition, when optimizing both the location and forms of winds that are represented by continuous variables, this problem is formulated as a mixed-variable optimization problem. We term this problem structure as an optimization problem on the sets of points, and we formulate this problem as an optimization problem on the search space consisting of several subspaces. Each subspace contains multiple possible points where the objective function value can be computed. In mixed-variable optimization, some of the subspace is treated as continuous space. The optimization problems on the sets of points have been found in several real-world applications such as design optimization of vehicle [15,16] and facility layout optimization [8]. We note that the naive CMA-ES fails to optimize such optimization problems on the sets of points, which will be observed in our experimental results.

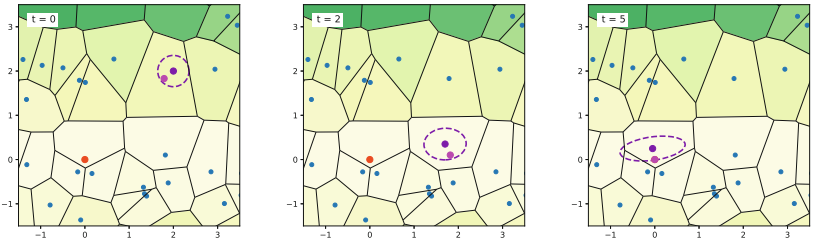


Fig. 1. Illustration of the search space and the optimization process of CMA-ES-SoP on the two-dimensional Ellipsoid function. The acceptable solutions are depicted as blue points. The red and magenta points are the optimum and closest points, respectively. The CMA-ES-SoP uses the Voronoi diagram to encode the samples and adjust the margin value. (Color figure online)

In this paper, we tailor the CMA-ES for optimization on the sets of points and propose CMA-ES-SoP (CMA-ES on sets of points). Figure 1 shows the conceptual image of the optimization with the CMA-ES-SoP. The CMA-ES-SoP incorporates three handlings: sample encoding, margin correction, and margin

adaptation. In the encoding process, the samples generated from a multivariate Gaussian distribution are projected to the closest points to the samples. In the margin correction, the covariance matrix is modified to maintain the marginal generation probability beyond the mid-points between the mean vector and neighboring points in the Voronoi diagram above the margin. Finally, to prevent an unnecessary increase of marginal probabilities after the margin adaptation, the margin is adjusted so that the average of marginal probabilities is maintained close to the target value for the margin.

We evaluated the performance of CMA-ES-SoP using numerical simulations with benchmark functions. In the experiment with discrete optimization on sets of points, the CMA-ES-SoP successfully optimized the benchmark functions with high probability, while the CMA-ES failed to optimize them. In the experiment with mixed-variable optimization, the CMA-ES-SoP outperformed the CMA-ES in most functions, especially in high-dimensional problems.

Notations. The functions $\Phi_{\mathrm{cdf}} : \mathbb{R} \to (0,1)$ and $\Phi_{\mathrm{ppf}} : (0,1) \to \mathbb{R}$ are the cumulative density function of the standard normal distribution $\mathcal{N}(0,1)$ and the inverse function of Φ_{cdf} called the percentile point function, respectively. We denote the concatenation of n vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n$ as $\textsc{Concat}(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n) = (\boldsymbol{v}_1^{\mathrm{T}}, \cdots, \boldsymbol{v}_n^{\mathrm{T}})^{\mathrm{T}}$.

2 Related Works

Evolutionary Algorithms for Discrete Optimization: Integer optimization, which is the optimization of integer variables, is a related problem to the optimization on sets of points. Several evolutionary algorithms (EAs) have been developed for integer optimization. The coordinate-wise mutation is a widely-used approach for integer optimization [6,7]. Particularly, (1+1)-EA with self-adjusting mutation is a promising method with theoretical guarantee [7]. Although there are several other approaches, including binary encoding [19] and the probabilistic model-based approach [4], there is no approach that can directly be applied to the optimization on sets of points.

The optimization problem with categorical variables is another topic for EAs [1,5]. As the points in the set can be treated as categories, these methods can be applied to the optimization on sets of points. However, as the positional relationship between points is not addressed, the optimization performance is limited. In addition, these EAs cannot deal with mixed-variable optimization problems containing both discrete and continuous variables.

Integer Handling for Mixed-Integer Optimization: Another related work is optimization methods for mixed-integer optimization. Several powerful optimization methods have been developed by introducing integer handlings to powerful continuous optimization methods such as the CMA-ES. The study [10] injects the integer mutation vector into the generation process of candidate solutions in the CMA-ES. The CMA-ES with margin [9] incorporates the margin correction to

maintain the marginal probabilities associated with the integer variables above a certain value. The DX-NES-ICI [14] leaps the elements of the mean vector corresponding to integer variables to overcome the performance deterioration of CMA-ES with margin when the evaluation value is more influenced by continuous variables than integer variables. Although these are powerful methods for mixed-integer optimization problems, they do not adequately handle scenarios involving a mix of continuous variables and variables on a set of points.

3 Baseline Algorithm: CMA-ES

CMA-ES [13] is a powerful black-box optimization method on continuous space. The CMA-ES employs a multivariate Gaussian distribution as a sampling distribution of the candidate solutions and updates the distribution parameters to generate better solutions. The multivariate Gaussian distribution is parameterized by the mean vector $\boldsymbol{m} \in \mathbb{R}^N$, the step-size $\sigma \in \mathbb{R}_{>0}$, and the covariance matrix $\boldsymbol{C} \in \mathbb{R}^{N \times N}$. The CMA-ES also employs two evolution paths $\boldsymbol{p}_c \in \mathbb{R}^N$ and $\boldsymbol{p}_\sigma \in \mathbb{R}^N$ that are initialized to zero vectors.

The update procedure of the CMA-ES is as follows. First, the CMA-ES generates λ solutions $\boldsymbol{x}^{\langle 1 \rangle}, \cdots, \boldsymbol{x}^{\langle \lambda \rangle}$ as

$$\boldsymbol{z}^{\langle i \rangle} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \ , \quad \boldsymbol{y}^{\langle i \rangle} = \sqrt{\boldsymbol{C}^{(t)}} \boldsymbol{z}^{\langle i \rangle} \ , \text{ and } \quad \boldsymbol{x}^{\langle i \rangle} = \sigma^{(t)} \boldsymbol{y}^{\langle i \rangle} + \boldsymbol{m}^{(t)} \ . \tag{1}$$

Subsequently, the generated solutions are evaluated on the objective function $f : \mathbb{R}^N \to \mathbb{R}$ to be optimized. We denote the index of the i-th best solution as $i : \lambda$.

Next, the CMA-ES updates the evolution paths as

$$\boldsymbol{p}_\sigma^{(t+1)} = (1 - c_\sigma) \boldsymbol{p}_\sigma^{(t)} + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} \cdot \sum_{i=1}^{\mu} w_i \boldsymbol{z}^{\langle i:\lambda \rangle} \tag{2}$$

$$\boldsymbol{p}_c^{(t+1)} = (1 - c_c) \boldsymbol{p}_c^{(t)} + h_\sigma^{(t+1)} \sqrt{c_c (2 - c_c) \mu_{\text{eff}}} \cdot \sum_{i=1}^{\mu} w_i \boldsymbol{y}^{\langle i:\lambda \rangle} \ , \tag{3}$$

where $\{w_i\}_{i=1}^{\mu}$ are predefined positive weights, $c_\sigma, c_c \in \mathbb{R}_{>0}$ are the accumulation rates of the evolution paths, and $\mu_{\text{eff}} = (\sum_{i=1}^{\mu} w_i^2)^{-1}$ is the variance effective selection mass. The Heaviside function $h_\sigma^{(t+1)} \in \{0, 1\}$ becomes $h_\sigma^{(t+1)} = 1$ if and only if it satisfies:

$$\frac{\| \boldsymbol{p}_\sigma^{(t+1)} \|}{\sqrt{1 - (1 - c_\sigma)^{2(t+1)}}} < \left(1.4 + \frac{2}{N + 1} \right) \chi_N \ , \tag{4}$$

where $\chi_N = \sqrt{N} \left(1 - \frac{1}{4N} + \frac{1}{21N^2} \right)$ is the approximated value of the expectation $\mathbb{E}[\| \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \|]$. Otherwise, it becomes $h_\sigma^{(t+1)} = 0$.

Finally, the CMA-ES updates the distribution parameters of the multivariate Gaussian distribution as

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + c_m \sum_{i=1}^{\mu} w_i \left(\boldsymbol{x}^{\langle i:\lambda \rangle} - \boldsymbol{m}^{(t)} \right) \tag{5}$$

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|}{\chi_N} - 1 \right) \right) \tag{6}$$

$$\begin{aligned} \boldsymbol{C}^{(t+1)} = \left(1 + \delta(h_\sigma^{(t+1)})\right) \boldsymbol{C}^{(t)} + c_1 \left(\boldsymbol{p}_c^{(t+1)} \left(\boldsymbol{p}_c^{(t+1)} \right)^{\mathrm{T}} - \boldsymbol{C}^{(t)} \right) \\ + c_\mu \sum_{i=1}^{\mu} w_i \left(\boldsymbol{y}^{\langle i:\lambda \rangle} \left(\boldsymbol{y}^{\langle i:\lambda \rangle} \right)^{\mathrm{T}} - \boldsymbol{C}^{(t)} \right) \quad , \end{aligned} \tag{7}$$

where $c_m, c_1, c_\mu \in \mathbb{R}_{>0}$ are the learning rates, $d_\sigma \in \mathbb{R}_{>0}$ is the damping factor, and $\delta(h) = (1-h)c_1 c_c(2-c_c)$. The CMA-ES has well-tuned default values for all hyperparameters. Details are available in the literature [11,12].

4 Target Problem

In this study, we consider the search space $\mathcal{X} \subseteq \mathbb{R}^N$ consisting of K subspaces as

$$\mathcal{X} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_K \quad . \tag{8}$$

In discrete optimization problems on sets of points, each subspaces $\mathcal{S}_k \subseteq \mathbb{R}^{N_k}$ is given by sets of L_k points, i.e., $\mathcal{S}_k = \{\boldsymbol{s}_{k,1}, \cdots, \boldsymbol{s}_{k,L_k}\}$, where $N = \sum_{k=1}^{K} N_k$. We assume that the sets of points are accessible for the optimization methods. The candidate solutions are constructed by selecting points from subspaces. In mixed-variable optimization problems, a part of search space is given by the continuous space, and other space is given by sets of points, i.e., $\mathcal{X} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_{K-1} \times \mathbb{R}^{N_{\mathrm{co}}}$.

Such problems can be found in applications of system design and manufacturing. For example, in the design optimization of vehicle [15,16], the optimization is sometimes performed by selecting available parts of machines. In the optimization of position for constructing wind turbines [2], the potential places can be listed in advance, and the optimal place is selected from the listed possible places. Additionally, when optimizing both the position and form of the wing, which is controlled by continuous variables, the problem becomes a mixed-variables optimization problem. Similar problem structures can also be found in facility layout problems [8] and location-routing problems [18].

As well on the mixed-integer optimization problems, the naive CMA-ES usually fails to optimize the optimization on the sets of points. This is because the original update rule of the CMA-ES leads to premature convergence around non-optimal points, as observed in our experimental results shown in Sect. 6.

5 Proposed Method: CMA-ES-SoP

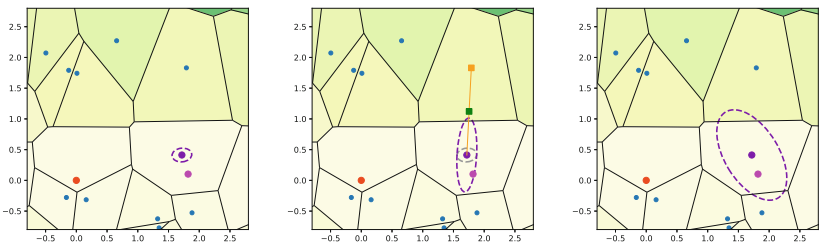
We propose a novel variant of the CMA-ES for optimization on sets of points, termed CMA-ES-SoP. The CMA-ES-SoP encodes the samples generated from the multivariate Gaussian distribution to obtain the candidate solution on the search space. After updating the distribution parameters, the CMA-ES-SoP corrects the covariance matrix $\boldsymbol{C}^{(t+1)}$ to maintain the generation probability beyond the mid-points from the neighboring points above the margin value $\alpha^{(t)}$. In addition, we adapt the margin value $\alpha_k^{(t)} \in \mathbb{R}_{>0}$ to prevent an unnecessary increase of the marginal probabilities for a part of neighboring points. Algorithm 1 shows the pseudo-code of the CMA-ES-SoP.

5.1 Sample Encoding

The CMA-ES-SoP transforms the samples $\boldsymbol{x}^{\langle 1 \rangle}, \cdots, \boldsymbol{x}^{\langle \lambda \rangle}$ generated from the multivariate Gaussian distribution into the candidate solutions on the search space. For the elements $\boldsymbol{x}_k^{\langle i \rangle}$ of $\boldsymbol{x}^{\langle i \rangle}$ corresponding to the k-th subspace, the closest points $\textsc{Enc}_k(\boldsymbol{x}_k^{\langle i \rangle})$ in $\mathcal{S}_k$ are selected as a part of encoded candidate solution, where $\textsc{Enc}_k : \mathbb{R}^{N_k} \to \mathcal{S}_k$ is defined as

$$\textsc{Enc}_k(\boldsymbol{x}_k) = \underset{\boldsymbol{s} \in \mathcal{S}_k}{\arg\min} \|\boldsymbol{x}_k - \boldsymbol{s}\| \ . \tag{9}$$

Then, the encoded candidate solution $\tilde{\boldsymbol{x}}^{\langle i \rangle} = \textsc{Concat}(\textsc{Enc}_1(\boldsymbol{x}_1^{\langle i \rangle}), \cdots, \textsc{Enc}_K(\boldsymbol{x}_K^{\langle i \rangle}))$ is evaluated on the objective function. The closest point is determined by the Voronoi region containing the corresponding elements of sample $\boldsymbol{x}_k^{\langle i \rangle}$ on the subspace.



(a) Before correction (b) A step of correction (c) After correction

Fig. 2. Illustration of margin correction in CMA-ES-SoP. The orange and green square points in the center figure represent the neighboring point and mid-point, respectively. The gray and violet ellipses in the center figure correspond to the covariance matrices of the multivariate Gaussian distribution before and after a single step of margin correction, respectively. (Color figure online)

Algorithm 1. CMA-ES on Sets of Points

Input: The objective function $f : \mathbb{R}^N \to \mathbb{R}$ and subspaces $\mathcal{S}_1, \cdots, \mathcal{S}_K$.
Input: Initial distribution parameters $\boldsymbol{m}^{(0)}, \boldsymbol{C}^{(0)}, \sigma^{(0)}$.
Input: Hyperparameters $\alpha_{\text{target}} = 1/(\lambda N)$ and $\beta = 1 + 1/N$.
Initialize the margin value as $\alpha_k^{(0)} = \alpha_{\text{target}}$.
while termination condition is not met **do**
  for $i = 1, \cdots, \lambda$ **do**
    Generate $\boldsymbol{x}^{\langle i \rangle}$ from the multivariate Gaussian distribution using (1).
    Encode $\boldsymbol{x}^{\langle i \rangle}$ to $\tilde{\boldsymbol{x}}^{\langle i \rangle}$ by concatenating the nearest points as (9).
    Evaluate $\tilde{\boldsymbol{x}}^{\langle i \rangle}$ on the objective function f.
  end for
  Update $\boldsymbol{m}^{(t)}, \boldsymbol{C}^{(t)}, \sigma^{(t)}$ using the update rules (5), (6), and (7) with samples $\boldsymbol{x}^{\langle 1 \rangle}, \cdots, \boldsymbol{x}^{\langle \lambda \rangle}$ before encoding.
  for $k = 1, \cdots, K$ **do**
    Compute the neighboring points $\mathcal{S}_k^{\text{neighbor}}$ to the mean vector in k-th subspace.
    for $\boldsymbol{s}_{k,b}^{\text{neighbor}} \in \mathcal{S}_k^{\text{neighbor}}$ **do**
      Compute the marginal probability $p_{k,b}^{(t+1)}$ in (12).
      if $p_{k,b}^{(t+1)} < \alpha_k^{(t)}$ **then**
        Correct $\boldsymbol{C}^{(t+1)}$ by margin correction with $\alpha_k^{(t)}$ as (15).
      end if
    end for
    Adjust the margin $\alpha_k^{(t)}$ using the probabilities $p_{k,1}^{(t+1)}, \cdots, p_{k,B^{(t+1)}}^{(t+1)}$ as (16).
  end for
  $t \leftarrow t + 1$
end while

5.2 Margin Correction

The margin correction aims to maintain the generation probability of neighboring points above the margin value to prevent early convergence. Because the exact computation of generation probability over a non-linearly constrained region is intractable, we develop our margin correction with an alternative tail probability. Figure 2 shows an example of margin correction.

We consider the closest point $\boldsymbol{s}_k^{\text{close}} = \textsc{Enc}_k(\boldsymbol{m}_k^{(t+1)})$ to the mean vector on the k-th subspace. We then compute the neighboring points around $\boldsymbol{s}_k^{\text{close}}$ on the Voronoi diagram as

$$\mathcal{S}_k^{\text{neighbor}} = \{\boldsymbol{s}_{k,1}^{\text{neighbor}}, \cdots, \boldsymbol{s}_{k,B_k^{(t+1)}}^{\text{neighbor}}\} \subseteq \mathcal{S}_k \ , \tag{10}$$

where $B_k^{(t+1)}$ is the number of neighboring points. With a neighboring point $\boldsymbol{s}_{k,b}^{\text{neighbor}}$, the mid-point $\boldsymbol{s}_{k,b}^{\text{mid}}$ between the mean vector $\boldsymbol{m}_k^{(t+1)}$ and $\boldsymbol{s}_{k,b}^{\text{neighbor}}$ is computed as

$$\boldsymbol{s}_{k,b}^{\text{mid}} = \frac{\boldsymbol{m}_k^{(t+1)} + \boldsymbol{s}_{k,b}^{\text{neighbor}}}{2} \ . \tag{11}$$

Then, we consider the marginal distribution along the direction $\boldsymbol{s}_{k,b}^{\mathrm{mid}} - \boldsymbol{m}_k^{(t+1)}$ to compute the alternative tail probability to exact generation probability. We aim to maintain the generation probability $p_{k,b}^{(t+1)}$ beyond the mid-point on the marginal distribution above the margin value $\alpha_k^{(t)}$. The generation probability is computed as

$$p_{k,b}^{(t+1)} = \Phi_{\mathrm{cdf}}\left(-d_{k,b}^{(t+1)}\right) \quad , \tag{12}$$

where $d_{k,b}^{(t+1)}$ is the Mahalanobis distance between $\boldsymbol{s}_{k,b}^{\mathrm{mid}}$ and $\boldsymbol{m}_k^{(t+1)}$ on the k-th subspace as

$$d_{k,b}^{(t+1)} = \left\| \sqrt{\left(\boldsymbol{C}^{(t+1)}\right)^{-1}} \cdot \xi_{k,b}^{(t+1)} \right\| \tag{13}$$

with a vector $\xi_{k,b}^{(t+1)}$ defined by two zero vectors $\mathbf{0}_k^{\mathrm{ant}}$ and $\mathbf{0}_k^{\mathrm{post}}$ with the lengths of $\sum_{j=1}^{k-1} N_j$ and $\sum_{j=k+1}^{K} N_j$ as

$$\xi_{k,b}^{(t+1)} = \textsc{Concat}\left(\mathbf{0}_k^{\mathrm{ant}}, \left(\frac{\boldsymbol{s}_{k,b}^{\mathrm{mid}} - \boldsymbol{m}_k^{(t+1)}}{\sigma^{(t+1)}}\right), \mathbf{0}_k^{\mathrm{post}}\right) \quad . \tag{14}$$

When $p_{k,b}$ is smaller than $\alpha_k^{(t)}$, the covariance matrix is modified to

$$\boldsymbol{C}^{(t+1)} \leftarrow \boldsymbol{C}^{(t+1)} + \frac{(d_{k,b}^{(t+1)})^2 - (\gamma_\alpha^{(t)})^2}{(d_{k,b}^{(t+1)})^2 (\gamma_\alpha^{(t)})^2} \cdot \xi_{k,b}^{(t+1)} \left(\xi_{k,b}^{(t+1)}\right)^{\mathrm{T}} \quad , \tag{15}$$

where $\gamma_\alpha^{(t)} = \Phi_{\mathrm{ppf}}(1 - \alpha_k^{(t)})$. This modification maintains the Mahalanobis distance $d_{k,b}^{(t+1)}$ at most $\gamma_\alpha^{(t)}$, which ensures $p_{k,b}^{(t+1)} \geq \alpha_k^{(t)}$ (see Appendix A). We note that $\xi_{k,b}^{(t+1)}$ can contain a non-zero value on the elements corresponding to k-th subspace, and the margin correction does not change the variance on other subspaces. The CMA-ES-SoP shuffles the neighboring points before each margin correction and applies the correction for each neighboring point in turn.

5.3 Margin Adaptation

In the margin correction explained in the previous subsection, a single step in the correction of the covariance matrix in Eq. (15) may increase the marginal probabilities for other neighboring points more than necessary. To prevent the performance deterioration due to such unnecessary increase of the marginal probabilities, we adjust the margin value $\alpha_k^{(t)}$ so that the average of probabilities $p_{k,1}^{(t+1)}, \cdots, p_{k,B_k^{(t+1)}}^{(t+1)}$ on marginal distribution is maintained close to the target margin value α_{target}. We realize this adjustment by employing the update rule for $\alpha_k^{(t)}$ given by

$$\alpha_k^{(t+1)} = \begin{cases} \alpha_k^{(t)} / \beta & \text{if} \quad \alpha_{\text{target}} \leq \frac{1}{B_k^{(t+1)}} \sum_{b=1}^{B_k^{(t+1)}} p_{k,b}^{(t+1)} \\ \beta \cdot \alpha_k^{(t)} & \text{otherwise} \end{cases} . \tag{16}$$

We set the target margin value and the increasing and decreasing factor as $\alpha_{\text{target}} = 1/(N\lambda)$ and $\beta = 1 + 1/N$, respectively. We note the target margin value follows the reference [9].

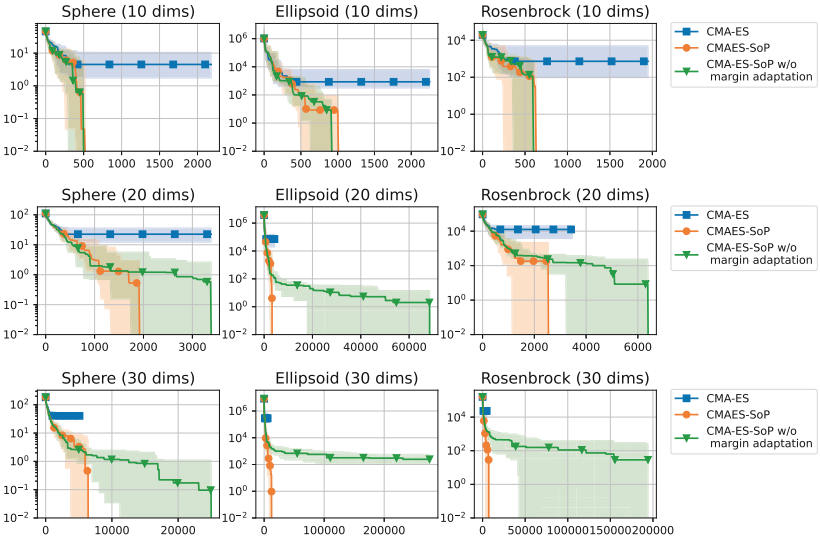


Fig. 3. Transitions of the best evaluation values on the discrete optimization problems with $(N_k, L_k) = (2, 10)$. We plot the median and interquartile ranges over 25 independent trials.

6 Experiment

We evaluated the optimization performance of the CMA-ES-SoP on the discrete optimization problems on the sets of points in Sect. 6.2 and the mixed-variable optimization problems in Sect. 6.3. The code of the CMA-ES-SoP will be made available at https://github.com/CyberAgentAILab/cmaes [17].

6.1 Experimental Setting

We prepared four benchmark functions as follows:

- Sphere: $f(\boldsymbol{x}) = \sum_{i=1}^{N} x_i^2$
- Ellipsoid: $f(\boldsymbol{x}) = \sum_{i=1}^{N} \left(1000^{\frac{i-1}{N-1}} x_i\right)^2$

Table 1. Success rate (SR) and SP1 in discrete optimization on sets of points.

Problem Setting		Method	Sphere		Ellipsoid		Rosenbrock	
			SR	SP1	SR	SP1	SR	SP1
$N_k = 2$ $L_k = 10$	$N = 10$	CMA-ES	0.20	**1410.0**	0.00	–	0.24	**1069.4**
		CMA-ES-SoP	**1.00**	1611.2	**0.96**	**1406.6**	**0.96**	1282.1
	$N = 20$	CMA-ES	0.00	–	0.00	–	0.00	–
		CMA-ES-SoP	**1.00**	**3811.6**	**1.00**	**5002.5**	**1.00**	**6043.6**
	$N = 30$	CMA-ES	0.00	–	0.00	–	0.00	–
		CMA-ES-SoP	**1.00**	**9456.1**	**1.00**	**12291.4**	**0.96**	**12534.9**
$N_k = 5$ $L_k = 40$	$N = 10$	CMA-ES	0.32	277.3	0.12	805.5	0.44	**78.5**
		CMA-ES-SoP	**1.00**	**213.2**	**1.00**	**541.6**	**1.00**	134.8
	$N = 20$	CMA-ES	0.00	–	0.00	–	0.04	4800.0
		CMA-ES-SoP	**1.00**	**765.6**	**1.00**	**4431.3**	**0.96**	**1679.6**
	$N = 30$	CMA-ES	0.00	–	0.00	–	0.00	–
		CMA-ES-SoP	**1.00**	**2107.28**	**1.00**	**7458.6**	**1.00**	**2667.2**

- Reversed Ellipsoid: $f(\boldsymbol{x}) = \sum_{i=1}^{N} \left(1000^{\frac{N-i}{N-1}} x_i\right)^2$
- Rosenbrock: $f(\boldsymbol{x}) = \sum_{i=1}^{N-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$

Sphere, Ellipsoid, and Rosenbrock are well-known benchmark functions. We added Reversed Ellipsoid for comparison with Ellipsoid, which deepens our discussions in mixed-variable optimization.

The sets of points $\mathcal{S}_k$ was given by $L_k - 1$ samples generated from the uniform distribution on $[-5, 5]^{N_k}$. We then added the optimal solution of benchmark functions to $\mathcal{S}_k$. We performed two settings for N_k and L_k as $(N_k, L_k) \in \{(2, 10), (5, 40)\}$. We varied the total number of dimensions as $N = 10, 20, 30$.

We ran the naive CMA-ES explained in Sect. 3 (with the sample encoding in Sect. 5.1) as a comparative method. We also compared the CMA-ES-SoP without the margin adaptation, in which $\alpha_k^{(t)} = \alpha_{\text{target}}$ was fixed for all iterations. For both CMA-ES and CMA-ES-SoP, the initial mean vector $\boldsymbol{m}^{(0)}$ was given uniformly at random on $[1, 5]^N$. The initial covariance and step-size were given by $\boldsymbol{C}^{(0)} = \mathbf{I}$ and $\sigma^{(0)} = 2$, respectively. We terminated the optimization when one of the following four conditions was met: 1) the successful condition was satisfied, 2) the number of evaluations reached $N \times 10^4$, 3) the minimum eigenvalue of $(\sigma^{(t)})^2 \boldsymbol{C}^{(t)}$ was updated less than 10^{-30}, or 4) a numerical error occurred.

6.2 Experimental Result in Discrete Optimization on Sets of Points

First, we show the results with the discrete optimization problems on sets of points. In this setting, the search space consisted of $K = N/N_k$ sets of points.

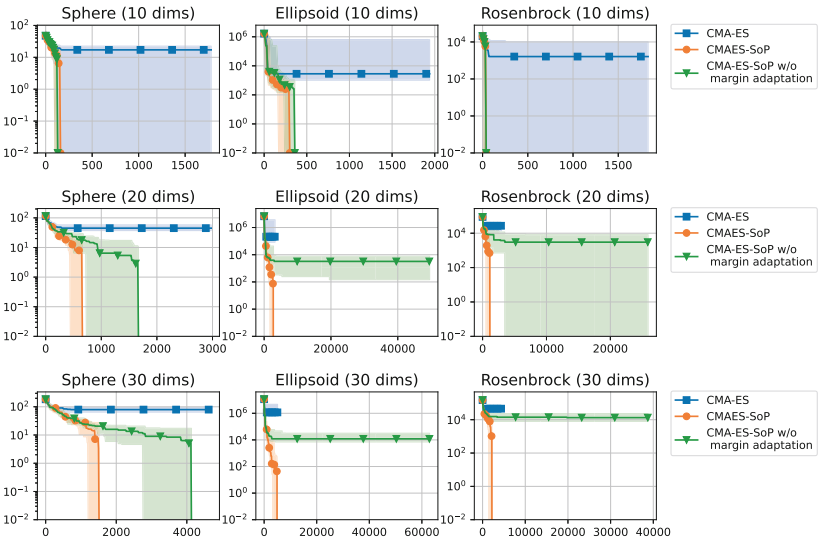


Fig. 4. Transitions of the best evaluation values on the discrete optimization problems with $(N_k, L_k) = (5, 40)$. We plot the median and interquartile ranges over 25 independent trials.

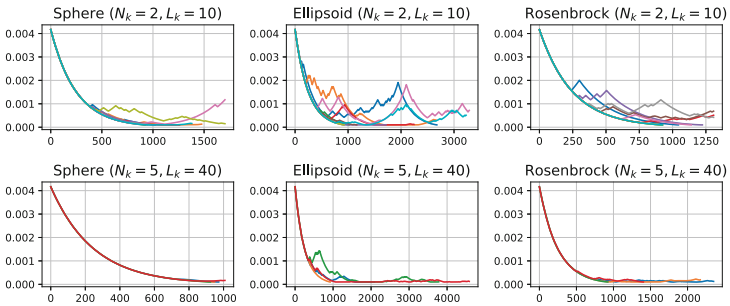


Fig. 5. Transitions of margins for each subspace. They were observed in a typical trial of the CMA-ES-SoP on discrete 20-dimensional optimization problems.

We regarded a trial as successful when the optimal solution was found. We ran 25 independent trials on Sphere, Ellipsoid, and Rosenbrock for each setting.

Figures 3 and 4 show the transitions of the best evaluation values with $(N_k, L_k) = (2, 10)$ and $(N_k, L_k) = (5, 40)$, respectively. We observed that the CMA-ES-SoP successfully optimized all benchmark problems while the CMA-ES often stagnated on all benchmark functions. Note that the optimization of the CMA-ES is usually terminated due to a too small eigenvalue of the covariance matrix on the search distribution. We consider that the margin correction in CMA-ES-SoP prevented such premature convergence. We also observed that margin adaptation improved the optimization performance for high-dimensional

problems. Figure 5 shows the transition of the margins in a typical trial. We can see that the dynamics of the margin change depending on the function, subspace dimension, and number of data, highlighting the importance of margin adaptation.

Table 1 shows the success rates and SP1 values computed with 25 trials. SP1 is the average number of evaluations over successful trials divided by the success rate [3]. We can see that the success rates of the CMA-ES-SoP were significantly better than those of the CMA-ES in all problem settings. Focusing on SP1, however, the CMA-ES was superior to the CMA-ES-SoP on some low-dimensional problems. We note that, although the CMA-ES failed to optimize in most of trials, it sometimes quickly converged to the optimum solution.

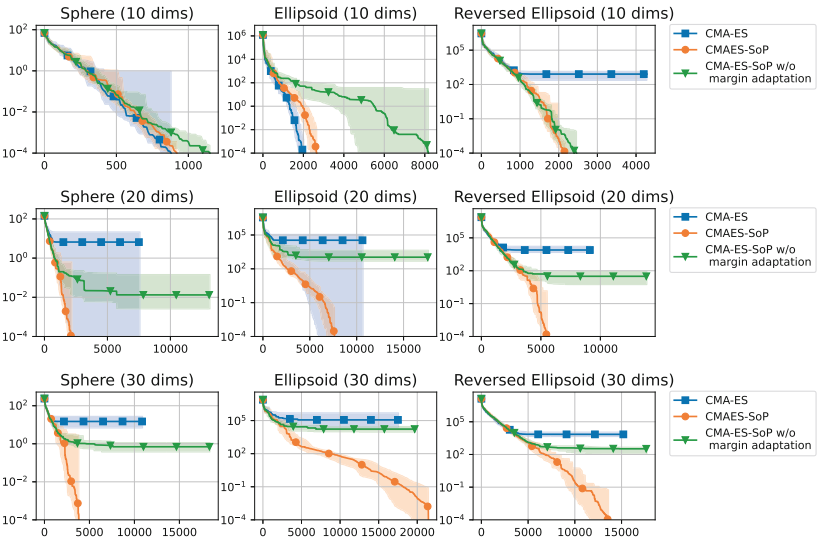


Fig. 6. Transitions of the best evaluation values on the mixed-variable optimization problems with $(N_k, L_k) = (2, 10)$. We plot the median and interquartile ranges over 25 independent trials.

6.3 Experimental Result in Mixed-Variable Optimization

Next, we show the results with the mixed-variable optimization problems. In this setting, we set the number of sets of points as $N_{\mathrm{set}} = \lfloor N/N_k/2 \rfloor$. In this experimental setting, the search space of the first $N_{\mathrm{set}}N_k$ design variables was given by the sets of points, while the remaining $N_K = N - N_{\mathrm{set}}N_k$ design variables were treated as continuous variables. We regarded a trial as successful when the best evaluation value reached 10^{-4}. We ran 25 independent trials on Sphere, Ellipsoid, and Reversed Ellipsoid for each setting.

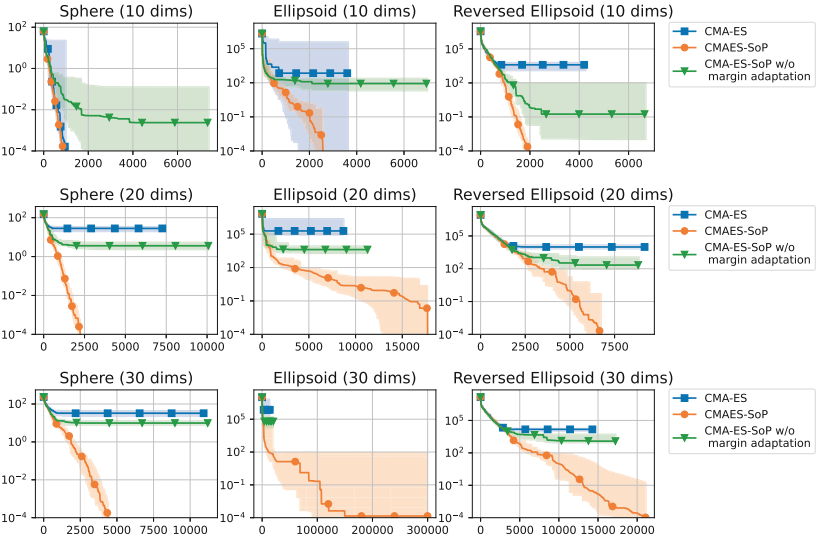


Fig. 7. Transitions of the best evaluation values on the mixed-variable optimization problems with $(N_k, L_k) = (5, 40)$. We plot the median and interquartile ranges over 25 independent trials.

Figures 6 and 7 show the transitions of the best evaluation values with $(N_k, L_k) = (2, 10)$ and $(N_k, L_k) = (5, 40)$, respectively. On 20- and 30-dimensional problems, the CMA-ES-SoP outperformed the CMA-ES. In contrast, the CMA-ES was sometimes competitive or superior to the CMA-ES-SoP on the 10-dimensional problems. We note the number of possible points for discrete variables was significantly low on the 10-dimensional problems, which was 100 when $(N_k, L_k) = (2, 10)$ and 40 when $(N_k, L_k) = (5, 40)$. This is why the CMA-ES could find the optimum points on the sets of points and showed competitive and superior performance compared to the CMA-ES-SoP. As well as the result in the previous subsection, the margin adaptation improved the optimization performance on high-dimensional problems. Because we did not tune the increasing and decreasing factor β, tuning it may improve the optimization performance.

Table 2 shows the success rates and SP1 values computed with 25 trials. In addition to the discrete optimization in the previous subsection, we can see that the success rates of CMA-ES-SoP were significantly higher compared to CMA-ES in all problem settings. In addition, the CMA-ES-SoP achieved smaller SP1 values than that of CMA-ES under all settings on Sphere and Reversed Ellipsoid. On Ellipsoid, however, the CMA-ES was sometimes better than the CMA-ES-SoP with respect to SP1. We consider the following reason: when optimizing the Ellipsoid by the CMA-ES, the variance corresponding to the continuous variables converged faster than the variance corresponding to the discrete variables. Therefore, unlike on the other functions, the premature convergence of the CMA-ES in discrete subspace was relatively prevented.

Table 2. Success rate (SR) and SP1 in mixed-variable optimization.

Problem Setting		Method	Sphere		Ellipsoid		Reversed Ellipsoid	
			SR	SP1	SR	SP1	SR	SP1
$N_k = 2$ $L_k = 10$	$N = 10$	CMA-ES	0.72	2120.3	0.84	**3075.9**	0.00	–
		CMA-ES-SoP	**1.00**	**1567.2**	**1.00**	3652.8	**1.00**	**3605.6**
	$N = 20$	CMA-ES	0.28	11448.9	0.28	22646.9	0.00	–
		CMA-ES-SoP	**1.00**	**3632.6**	**1.00**	**10402.5**	**1.00**	**14764.8**
	$N = 30$	CMA-ES	0.04	120750.0	0.12	101188.8	0.00	–
		CMA-ES-SoP	**1.00**	**6444.4**	**1.00**	**25319.2**	**1.00**	**26569.7**
$N_k = 5$ $L_k = 40$	$N = 10$	CMA-ES	0.52	2871.3	0.40	**5455.0**	0.08	29687.5
		CMA-ES-SoP	**1.00**	**1594.0**	**0.92**	12787.3	**0.92**	**7545.3**
	$N = 20$	CMA-ES	0.04	19837.5	0.04	**44300.0**	0.00	–
		CMA-ES-SoP	**0.96**	**3835.4**	**0.76**	78968.1	**0.84**	**57559.8**
	$N = 30$	CMA-ES	0.04	119700.0	0.12	**238000.0**	0.00	–
		CMA-ES-SoP	**1.00**	**6890.8**	**0.48**	355264.5	**0.64**	**185078.9**

7 Conclusion

We have proposed an extension of the CMA-ES for discrete and mixed-variable optimization problems on sets of points. The proposed CMA-ES-SoP contains three additional steps: sample encoding, margin correction, and margin adaptation. In the sample encoding, the samples generated from multivariate Gaussian distribution are mapped to the closest points in the subspaces. In the margin correction, the updated covariance matrix $\boldsymbol{C}^{(t+1)}$ is modified to maintain the marginal generation probability $p_{k,b}^{(t+1)}$ above the margin $\alpha_k^{(t)}$. Finally, in the margin adaptation, the margin is adjusted so that the average of marginal probabilities is maintained close to the target value α_{target}. The numerical simulation showed the efficiency of CMA-ES-SoP in discrete and mixed-variable optimizations on sets of points.

In our experiment, we used the benchmark functions extended from the benchmarks for continuous optimization. As the benchmark functions for optimization on sets of points are not well-structured, developing suitable benchmark functions is one of our future works. Moreover, because the step-size adaptation in the CMA-ES assumes the optimization in continuous space, we will develop an efficient step-size adaptation for optimization on sets of points in the future.

A Ensuring Margin by Modification of Covariance

In the following, we omit the iterators for short, e.g. we denote $\xi_{k,b}^{(t+1)}$ as $\xi_{k,b}$. According to the Sherman-Morrison formula, the inverse matrix of modified

covariance matrix in Eq. (15) is given by

$$\boldsymbol{C}^{-1} = \bar{\boldsymbol{C}}^{-1} - \frac{\zeta \cdot \bar{\boldsymbol{C}}^{-1} \xi_{k,b} \xi_{k,b}^{\mathrm{T}} \bar{\boldsymbol{C}}^{-1}}{1 + \zeta \cdot \xi_{k,b}^{\mathrm{T}} \bar{\boldsymbol{C}}^{-1} \xi_{k,b}} \quad , \tag{17}$$

where $\bar{\boldsymbol{C}}$ is the covariance matrix before the margin correction, and $\zeta = (d_{k,b}^2 - \gamma_\alpha^2)/d_{k,b}^2/\gamma_\alpha^2$. Considering the relation $d_{k,b}^2 = \xi_{k,b}^{\mathrm{T}} \bar{\boldsymbol{C}}^{-1} \xi_{k,b}$, the squared Mahalanobis distance after the margin correction is given by

$$\xi_{k,b}^{\mathrm{T}} \boldsymbol{C}^{-1} \xi_{k,b} = \xi_{k,b}^{\mathrm{T}} \bar{\boldsymbol{C}}^{-1} \xi_{k,b} - \frac{\zeta \cdot \left(\xi_{k,b}^{\mathrm{T}} \bar{\boldsymbol{C}}^{-1} \xi_{k,b} \right)^2}{1 + \zeta \cdot \xi_{k,b}^{\mathrm{T}} \bar{\boldsymbol{C}}^{-1} \xi_{k,b}} = d_{k,b}^2 - \frac{\zeta \cdot d_{k,b}^4}{1 + \zeta \cdot d_{k,b}^2} \quad . \tag{18}$$

Then, substituting $\zeta = (d_{k,b}^2 - \gamma_\alpha^2)/d_{k,b}^2/\gamma_\alpha^2$ shows $\xi_{k,b}^{\mathrm{T}} \boldsymbol{C}^{-1} \xi_{k,b} = \gamma_\alpha^2$. Finally, remaining that Φ_{ppf} is the inverse function of Φ_{cdf} and $\gamma_\alpha = \Phi_{\mathrm{ppf}}(1 - \alpha_k) = -\Phi_{\mathrm{ppf}}(\alpha_k)$, we have $p_{k,b} = \alpha_k$ after the margin correction.

References

1. Akimoto, Y., Shirakawa, S., Yoshinari, N., Uchida, K., Saito, S., Nishida, K.: Adaptive stochastic natural gradient method for one-shot neural architecture search. In: Proceedings of the 36th International Conference on Machine Learning (ICML), vol. 97, pp. 171–180 (2019)
2. Andrew Ning, S., Damiani, R., Moriarty, P.J.: Objectives and constraints for wind turbine optimization. J. Solar Energy Eng. **136**(4), 041010 (2014). https://doi.org/10.1115/1.4027693
3. Auger, A., Hansen, N., Perez Zerpa, J.M., Ros, R., Schoenauer, M.: Experimental comparisons of derivative free optimization algorithms. In: Vahrenhold, J. (ed.) SEA 2009. LNCS, vol. 5526, pp. 3–15. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02011-7_3
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. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3583131.3590523
5. Berny, A.: Linear representation of categorical values. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 119–120. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3449726.3459513
6. Doerr, B., Doerr, C., Koetzing, T.: The right mutation strength for multi-valued decision variables. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, pp. 1115–1122. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2908812.2908891
7. Doerr, B., Doerr, C., Kötzing, T.: Static and self-adjusting mutation strengths for multi-valued decision variables. Algorithmica **80**(5), 1732–1768 (2018). https://doi.org/10.1007/s00453-017-0341-1
8. Drira, A., Pierreval, H., Hajri-Gabouj, S.: Facility layout problems: a survey. Annu. Rev. Control. **31**(2), 255–267 (2007). https://doi.org/10.1016/j.arcontrol.2007.04.001

9. Hamano, R., Saito, S., Nomura, M., Shirakawa, S.: CMA-ES with margin: lower-bounding marginal probability for mixed-integer black-box optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 639–647. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3512290.3528827
10. Hansen, N.: A CMA-ES for mixed-integer nonlinear optimization. Research Report RR-7751, INRIA (2011). https://inria.hal.science/inria-00629689
11. Hansen, N.: The CMA evolution strategy: a tutorial. CoRR abs/1604.00772 (2016)
12. Hansen, N., Auger, A.: Principled design of continuous stochastic search: from theory to practice. In: Borenstein, Y., Moraglio, A. (eds.) Theory and Principled Methods for the Design of Metaheuristics. NCS, pp. 145–180. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-33206-7_8
13. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 312–317 (1996). https://doi.org/10.1109/ICEC.1996.542381
14. Ikeda, K., Ono, I.: Natural evolution strategy for mixed-integer black-box optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 831–838. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3583131.3590518
15. Kohira, T., Kemmotsu, H., Akira, O., Tatsukawa, T.: Proposal of benchmark problem based on real-world car structure design optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 183–184. Association for Computing Machinery (2018). https://doi.org/10.1145/3205651.3205702
16. Laumanns, M., Laumanns, N.: Evolutionary multiobjective design in automotive development. Appl. Intell. **23**(1), 55–70 (2005). https://doi.org/10.1007/s10489-005-2372-6
17. Nomura, M., Shibata, M.: Cmaes: A Simple yet Practical Python Library for CMA-ES. arXiv preprint arXiv:2402.01373 (2024)
18. Prodhon, C., Prins, C.: A survey of recent research on location-routing problems. Eur. J. Oper. Res. **238**(1), 1–17 (2014). https://doi.org/10.1016/j.ejor.2014.01.005
19. Rothlauf, F.: Binary representations of integers and the performance of selectorecombinative genetic algorithms. In: Guervós, J.J.M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., Fernández-Villacañas, J.-L. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 99–108. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45712-7_10

Natural Gradient Interpretation of Rank-One Update in CMA-ES

Ryoki Hamano[1(✉)], Shinichi Shirakawa[2], and Masahiro Nomura[1]

[1] CyberAgent, Inc., Shibuya, Japan
{hamano_ryoki_xa,nomura_masahiro}@cyberagent.co.jp
[2] Yokohama National University, Yokohama, Japan
shirakawa-shinichi-bg@ynu.ac.jp

Abstract. The covariance matrix adaptation evolution strategy (CMA-ES) is a stochastic search algorithm using a multivariate normal distribution for continuous black-box optimization. In addition to strong empirical results, part of the CMA-ES can be described by a stochastic natural gradient method and can be derived from information geometric optimization (IGO) framework. However, there are some components of the CMA-ES, such as the rank-one update, for which the theoretical understanding is limited. While the rank-one update makes the covariance matrix to increase the likelihood of generating a solution in the direction of the evolution path, this idea has been difficult to formulate and interpret as a natural gradient method unlike the rank-μ update. In this work, we provide a new interpretation of the rank-one update in the CMA-ES from the perspective of the natural gradient with prior distribution. First, we propose maximum a posteriori IGO (MAP-IGO), which is the IGO framework extended to incorporate a prior distribution. Then, we derive the rank-one update from the MAP-IGO by setting the prior distribution based on the idea that the promising mean vector should exist in the direction of the evolution path. Moreover, the newly derived rank-one update is extensible, where an additional term appears in the update for the mean vector. We empirically investigate the properties of the additional term using various benchmark functions.

Keywords: Covariance Matrix Adaptation Evolution Strategy · Natural Gradient · Information Geometric Optimization

1 Introduction

The covariance matrix adaptation evolution strategy (CMA-ES) [10,13,16] is recognized as a state-of-the-art derivative-free stochastic algorithm for black-box continuous optimization problems. The CMA-ES proceeds the optimization by repeatedly sampling from the multivariate normal distribution and updating the distribution parameters such as the mean vector, the covariance matrix, and the step-size. Despite the small number of hyperparameters to be tuned, the CMA-ES shows high performance for non-linear, ill-conditioned, or multimodal problems [15,21].

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 252–267, 2024.
https://doi.org/10.1007/978-3-031-70068-2_16

In addition to its empirical success, the CMA-ES has attractive theoretical properties, such as its relationship to the natural gradient method. The update of the distribution parameters in the CMA-ES can be partially described by the stochastic natural gradient ascent [3]. In fact, the pure rank-μ update CMA-ES is an instance of information geometric optimization (IGO) [20], a unified framework for probabilistic model-based black-box optimization algorithms.

However, the theoretical understanding of the other components of the CMA-ES, such as the rank-one update, is limited. The rank-one update makes the covariance matrix to increase the likelihood of generating a solution in the direction of the evolution path, which accumulates the updating direction of the mean vector. Li and Zhang [18] discuss that the evolution path cancels opposite update directions of the mean vector and the rank-one update term with the evolution path serves as momentum term for the covariance matrix. However, unlike the rank-μ update, these ideas of the rank-one update have been difficult to formulate and interpret based on natural gradients.

In this work, we provide a new interpretation of the rank-one update in the CMA-ES from the perspective of the natural gradient with prior distribution. To this end, we first propose maximum a posterioiri IGO (MAP-IGO), which is the IGO framework extended to incorporate a prior distribution. The MAP estimation approach has provided new interpretations or extensions to existing methods, such as regularization in linear regression [8] and reinforcement learning [2,23]. Then, we derive the rank-one update from the MAP-IGO with the prior distribution set based on the idea that the promising mean vector should exist in the direction of the evolution path. Moreover, the newly derived rank-one update is extensible, where an additional term, we call *momentum update*, appears in the update for the mean vector depending on the setting of the prior distribution. We propose the CMA-ES with the momentum update as maximum a posteriori CMA-ES (MAP-CMA) and empirically investigate its properties using various benchmark functions.

The remainder of this paper is organized as follows. Section 2 shows background of this study, which is needed in the rest of the paper. In Sect. 3, we propose the MAP-IGO and derive the update rule when the multivariate normal distribution and the normal-inverse-Wishart distribution are applied to the MAP-IGO. In Sect. 4, we provide a new interpretation of the rank-one update in the CMA-ES and propose the MAP-CMA. Section 5 shows experimental results for the MAP-CMA on benchmark functions. Section 6 concludes the paper with future work.

2 Preliminaries

2.1 CMA-ES

The covariance matrix adaptation evolution strategy (CMA-ES) generates multiple solutions in each iteration from the multivariate normal distribution $\mathcal{N}(\boldsymbol{m}^{(t)}, (\sigma^{(t)})^2\boldsymbol{C}^{(t)})$. To generate candidate solutions from the distribution, the CMA-ES updates the distribution parameter, the mean vector $\boldsymbol{m}^{(t)} \in \mathbb{R}^N$, and the

covariance matrix $(\sigma^{(t)})^2 \boldsymbol{C}^{(t)} \in \mathbb{R}^{N \times N}$. This update partially corresponds to the natural gradient ascent in the parameter space, which is the steepest ascent with respect to the Fisher metric [4]. We will describe *information geometric optimization* [20], a framework that generalizes this natural gradient ascent step, in Sect. 2.2. This section introduces the well-known CMA-ES variant, $(\mu/\mu_{\mathrm{w}}, \lambda)$-CMA-ES, minimizing the objective function $f(\boldsymbol{x})$.

In the t-th iteration, the λ candidate solutions $\boldsymbol{x}_i$ $(i = 1, \ldots, \lambda)$ are generated as

$$\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(t)}) \ , \tag{1}$$

$$\boldsymbol{x}_i = \boldsymbol{m}^{(t)} + \sigma^{(t)} \boldsymbol{y}_i \ . \tag{2}$$

The mean vector $\boldsymbol{m}^{(t)}$ is updated as

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + c_m \sum_{i=1}^{\lambda} w_i (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}^{(t)}) \ , \tag{3}$$

where c_m is the learning rate for the mean vector, and w_i is the weight that holds $\sum_{i=1}^{\mu} w_i = 1, w_1 \geq w_2 \geq \cdots \geq w_\mu > 0$, $w_j = 0$ $(j = \mu + 1, \ldots, \lambda)$. The index of the i-th best sample is denoted as $i{:}\lambda$.

The CMA-ES employs the two evolution paths[1].

$$\boldsymbol{p}_\sigma^{(t+1)} = (1 - c_\sigma) \boldsymbol{p}_\sigma^{(t)} + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\mathrm{w}}} \boldsymbol{C}^{(t)^{-\frac{1}{2}}} \sum_{i=1}^{\lambda} w_i \boldsymbol{y}_{i:\lambda} \ , \tag{4}$$

$$\boldsymbol{p}_c^{(t+1)} = (1 - c_c) \boldsymbol{p}_c^{(t)} + \sqrt{c_c (2 - c_c) \mu_{\mathrm{w}}} \sum_{i=1}^{\lambda} w_i \boldsymbol{y}_{i:\lambda} \ , \tag{5}$$

where $\mu_{\mathrm{w}} = 1/\sum_{i=1}^{\mu} w_i^2$, c_σ and c_c are cumulative rates.

The covariance matrix $\boldsymbol{C}^{(t)}$ is updated as

$$\boldsymbol{C}^{(t+1)} \leftarrow \boldsymbol{C}^{(t)} + \underbrace{c_1 \left(\boldsymbol{p}_c^{(t+1)} \boldsymbol{p}_c^{(t+1)\top} - \boldsymbol{C}^{(t)} \right)}_{\text{rank-one update}} + \underbrace{c_\mu \sum_{i=1}^{\lambda} w_i \left(\boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\top} - \boldsymbol{C}^{(t)} \right)}_{\text{rank-}\mu\text{ update}} \ , \tag{6}$$

where c_1 and c_μ are the learning rates for the rank-one update and the rank-μ update, respectively.

The step-size $\sigma^{(t)}$ is updated as

$$\sigma^{(t+1)} \leftarrow \sigma^{(t)} \exp \left(\frac{c_\sigma}{d_\sigma} \left(\frac{\| \boldsymbol{p}_\sigma^{(t+1)} \|}{\mathbb{E}[\| \mathcal{N}(\boldsymbol{0}, \mathbf{I}) \|]} - 1 \right) \right) \ , \tag{7}$$

where d_σ is a damping factor. The expected norm $\mathbb{E}[\| \mathcal{N}(\boldsymbol{0}, \mathbf{I}) \|]$ is practically approximated by $\sqrt{N} \left(1 - \frac{1}{4N} + \frac{1}{21N^2}\right)$. The CMA-ES that only employs the mean vector update and the rank-μ update is called *pure rank-μ update CMA-ES* [16].

[1] The CMA-ES sometimes employs the indicator function h_σ to prevent evolution path $\boldsymbol{p}_c^{(t)}$ from rapidly lengthening.

2.2 Information Geometric Optimization

Information geometric optimization (IGO) [20] is a unified framework of probabilistic model-based black-box optimization algorithms. Given a family of probability distributions $\{P_{\boldsymbol{\theta}}\}$ on X parameterized by $\boldsymbol{\theta} \in \Theta$, the IGO transforms the original problem into the maximization of the expected value of $J_{\boldsymbol{\theta}^{(t)}} : \Theta \to \mathbb{R}$. The function $J_{\boldsymbol{\theta}^{(t)}}$ depends on the current distribution parameter $\boldsymbol{\theta}^{(t)}$ and is defined as the expectation of the utility function $W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x})$ over $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, i.e.,

$$J_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\theta}) = \int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \ . \tag{8}$$

The utility function $W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x})$ is defined based on the quantiles of f under the current distribution. This approach provides invariance under increasing transformations of the objective function. Let $q^{\leqslant}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \Pr_{\boldsymbol{x}' \sim p_{\boldsymbol{\theta}}}(f(\boldsymbol{x}') \leq f(\boldsymbol{x}))$ and $w : [0,1] \to \mathbb{R}$ be a non-increasing function. Assuming that the probability of having the same evaluation value for different samples is 0, the utility function $W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x})$ is defined as $w(q^{\leqslant}_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}))$. See [20] for the definition of $W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x})$ when this assumption is not satisfied.

The IGO maximizes Eq. (8) by natural gradient ascent. The natural gradient $\tilde{\nabla}_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\theta})$ is given by the product of the inverse of Fisher information matrix and the vanilla gradient, namely, $F^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\theta})$. Then, the natural gradient $\tilde{\nabla}_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\theta})$ is calculated as follows:

$$\tilde{\nabla}_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\theta}) = \int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) \tilde{\nabla}_{\boldsymbol{\theta}} (\ln p_{\boldsymbol{\theta}}(\boldsymbol{x})) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \ . \tag{9}$$

In the black-box setting, Eq. (9) cannot be computed analytically, hence the natural gradient of $J_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\theta})$ at $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$ is approximated by the Monte Carlo estimation using λ samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ generated from $P_{\boldsymbol{\theta}^{(t)}}$ as

$$\tilde{\nabla}_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \approx \frac{1}{\lambda} \sum_{i=1}^{\lambda} \hat{w}_i \ \tilde{\nabla}_{\boldsymbol{\theta}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_{i:\lambda})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \ , \tag{10}$$

where $\hat{w}_i$ is the ranking-based utility value that can be regarded as the estimation value of $W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}_i)$. Introducing the learning rate η, we obtain the update rule.

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \eta \sum_{i=1}^{\lambda} \frac{\hat{w}_i}{\lambda} \ \tilde{\nabla}_{\boldsymbol{\theta}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_{i:\lambda})\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \ . \tag{11}$$

The IGO framework recovers the pure rank-μ update CMA-ES when the family of normal distributions is applied, and the population-based incremental learning (PBIL) [7] when the family of Bernoulli distributions is applied.

3 Maximum a Posteriori IGO

In this section, we propose maximum a posteriori IGO (MAP-IGO), which is the IGO framework extended to incorporate a prior distribution in a principled way.

Furthermore, we apply the multivariate normal distribution and the normal-inverse-Wishart distribution, which is the conjugate prior distribution of the multivariate normal distribution, to the MAP-IGO.

3.1 Introducing Prior Information to IGO

First, we rewrite the existing IGO objective in the form of maximum likelihood (ML) estimation. Then, we further rewrite the ML estimation in the form of the maximum a posteriori (MAP) estimation.

Equivalence of IGO Objective to ML Estimation. To rewrite the existing IGO objective in the form of the ML estimation, we introduce a binary event $R_{\boldsymbol{\theta}^{(t)}} \in \{0, 1\}$ where it holds $p(R_{\boldsymbol{\theta}^{(t)}} = 1|\boldsymbol{x}) \propto W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x})$, inspired by [1,2,23]. It should be noted that $R_{\boldsymbol{\theta}^{(t)}} = 1$ depends on the current distribution parameter $\boldsymbol{\theta}^{(t)}$. Intuitively, the probability $p(R_{\boldsymbol{\theta}^{(t)}} = 1|\boldsymbol{x}) \propto W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x})$ is larger when $\boldsymbol{x}$ has a better objective function value than solutions sampled from $p_{\boldsymbol{\theta}^{(t)}}$. Here, we consider the marginal distribution of the event $R_{\boldsymbol{\theta}^{(t)}} = 1$ over $\boldsymbol{\theta}$.

$$p(R_{\boldsymbol{\theta}^{(t)}} = 1|\boldsymbol{\theta}) = \int p(R_{\boldsymbol{\theta}^{(t)}} = 1|\boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \ . \tag{12}$$

The important thing here is that the maximization of the IGO objective can be rewritten in the form of the ML estimation as follows:

$$\operatorname*{argmax}_{\boldsymbol{\theta} \in \Theta} J_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\theta} \in \Theta} p(R_{\boldsymbol{\theta}^{(t)}} = 1|\boldsymbol{\theta}) \ . \tag{13}$$

MAP Estimation Instead of ML Estimation. Equation (13) allows us to consider the optimization of the IGO objective as a kind of ML estimation. Then, we can introduce the prior distribution into the IGO by taking the approach of the MAP estimation [8]. To that end, we calculate the posterior distribution by using the Bayes' theorem as follows:

$$p(\boldsymbol{\theta}|R_{\boldsymbol{\theta}^{(t)}} = 1) \propto p(R_{\boldsymbol{\theta}^{(t)}} = 1|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \ , \tag{14}$$

where $p(\boldsymbol{\theta})$ is the prior distribution. We note that the prior distribution $p(\boldsymbol{\theta})$ can be set at each time depending on the current parameter $\boldsymbol{\theta}^{(t)}$. The resulting framework, MAP-IGO, estimates $\boldsymbol{\theta}$ as the mode of the posterior distribution:

$$\operatorname*{argmax}_{\boldsymbol{\theta} \in \Theta} \ln p(\boldsymbol{\theta} \mid R_{\boldsymbol{\theta}^{(t)}} = 1) = \operatorname*{argmax}_{\boldsymbol{\theta} \in \Theta} \ln p(R_{\boldsymbol{\theta}^{(t)}} = 1 \mid \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) \ . \tag{15}$$

3.2 Natural Gradient Update for MAP-IGO

To optimize Eq. (15) using the natural gradient ascent, we calculate the natural gradient of $\ln p(R_{\boldsymbol{\theta}^{(t)}} = 1 \mid \boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})$. We assume that $\int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} (=$

$\int_0^1 w(q)\mathrm{d}q \neq 0$ holds in this work[2]. First, the natural gradient $\tilde{\nabla}_{\boldsymbol{\theta}} \ln p(R_{\boldsymbol{\theta}^{(t)}} = 1 \mid \boldsymbol{\theta})$ can be calculated as follows:

$$\begin{aligned}
&\tilde{\nabla}_{\boldsymbol{\theta}} \ln p(R_{\boldsymbol{\theta}^{(t)}} = 1 \mid \boldsymbol{\theta}) \\
&\quad = \tilde{\nabla}_{\boldsymbol{\theta}} \ln \int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} && (16) \\
&\quad = \frac{1}{\int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}} \tilde{\nabla}_{\boldsymbol{\theta}} \int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}x && (17) \\
&\quad = \frac{1}{\int W^f_{\boldsymbol{\theta}^{(t)}}(x) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}} \int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) \tilde{\nabla}_{\boldsymbol{\theta}} (\ln p_{\boldsymbol{\theta}}(\boldsymbol{x})) p_{\boldsymbol{\theta}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \ . && (18)
\end{aligned}$$

According to [20], $\int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$ is constant and always equal to the average weight $\int_0^1 w(q)\mathrm{d}q$. Then, the natural gradient Eq. (18) at $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$ can be approximated by Monte Carlo estimation as follows, as with the original IGO framework:

$$\frac{1}{\frac{1}{\lambda}\sum_{i=1}^{\lambda} \hat{w}_i} \cdot \frac{1}{\lambda} \sum_{i=1}^{\lambda} \hat{w}_i \ \tilde{\nabla}_{\boldsymbol{\theta}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_{i:\lambda}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \ . \tag{19}$$

Note that $\int W^f_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \int_0^1 w(q)\mathrm{d}q \approx \frac{1}{\lambda}\sum_{i=1}^{\lambda} \hat{w}_i$. Finally, using the natural gradient of $\ln p(\boldsymbol{\theta})$ at $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$, the updated parameter $\theta^{(t+1)}$ is given as follows:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \eta \left(\sum_{i=1}^{\lambda} \frac{\hat{w}_i}{\sum_{j=1}^{\lambda} \hat{w}_j} \tilde{\nabla}_{\boldsymbol{\theta}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}_{i:\lambda}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} + \tilde{\nabla}_{\boldsymbol{\theta}} \ln p(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \right) \tag{20}$$

When we apply the multivariate normal distribution to the probability distribution and the normal-inverse-Wishart distribution, which is the conjugate prior of the multivariate normal distribution, to the prior distribution, we can derive the pure rank-μ update CMA-ES that can incorporate prior information. In addition, when we apply the family of Bernoulli distributions to the probability distribution and the family of beta distributions, where the beta distribution is the conjugate prior of the Bernoulli distribution, to the prior distribution, we can derive the PBIL that can incorporate prior distribution.

3.3 Natural Gradient for Normal-Inverse-Wishart Distribution

We apply the multivariate normal distribution to $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and the normal-inverse-Wishart distribution to $p(\boldsymbol{\theta})$. The normal-inverse-Wishart distribution is defined by

[2] It should be noted that while the assumption $\int_0^1 w(q)\mathrm{d}q \neq 0$ usually holds in the CMA-ES, some instances of IGO, such as compact genetic algorithm [17], do not satisfy this. We will not pursue this limitation in depth as our focus is on the CMA-ES.

$$p(\boldsymbol{\theta}) = \mathcal{N}\left(\boldsymbol{m} \mid \boldsymbol{\delta}, \frac{1}{\gamma}\boldsymbol{C}\right) \mathcal{W}^{-1}(\boldsymbol{C} \mid \boldsymbol{\Psi}, \nu) \ . \tag{21}$$

In the normal-inverse-Wishart distribution, the multivariate normal distribution is given as

$$\mathcal{N}\left(\boldsymbol{m} \mid \boldsymbol{\delta}, \frac{1}{\gamma}\boldsymbol{C}\right) = \left(\frac{\gamma}{2\pi}\right)^{\frac{N}{2}} |\boldsymbol{C}|^{-\frac{1}{2}} \exp\left(-\frac{\gamma}{2}(\boldsymbol{m}-\boldsymbol{\delta})^\top \boldsymbol{C}^{-1}(\boldsymbol{m}-\boldsymbol{\delta})\right) ,$$

where $\boldsymbol{\delta} \in \mathbb{R}^N$ and $\gamma > 0$. The inverse-Wishart distribution is given as

$$\mathcal{W}^{-1}(\boldsymbol{C} \mid \boldsymbol{\Psi}, \nu) = \frac{|\boldsymbol{\Psi}|^{\frac{\nu}{2}}}{2^{\frac{\nu N}{2}} \Gamma_N\left(\frac{\nu}{2}\right)} |\boldsymbol{C}|^{-\frac{\nu+N+1}{2}} \exp\left(-\frac{1}{2}\mathrm{Tr}\left(\boldsymbol{\Psi}\boldsymbol{C}^{-1}\right)\right) \ ,$$

where $\Gamma_N(\cdot)$ is the multivariate gamma function, $\nu > N-1$, and $\boldsymbol{\Psi} \in \mathbb{R}^{N\times N}$ is a positive define matrix.

We derive the natural gradient of the log-likelihood of the normal-inverse-Wishart distribution over the parameter space of $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. Let $\boldsymbol{\theta} = [\boldsymbol{m}^\top, \mathrm{vec}(\boldsymbol{C})^\top]^\top$, where $\mathrm{vec}(\boldsymbol{C})$ represents the matrix $\boldsymbol{C}$ rearranged into a column vector. The vanilla gradient of the log-likelihood of $p(\boldsymbol{\theta})$ is given as follows:

$$\nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{\theta}) = \begin{bmatrix} -\gamma\boldsymbol{C}^{-1}(\boldsymbol{m}-\boldsymbol{\delta}) \\ \frac{1}{2}\mathrm{vec}\left(\gamma\boldsymbol{C}^{-1}(\boldsymbol{m}-\boldsymbol{\delta})(\boldsymbol{m}-\boldsymbol{\delta})^\top\boldsymbol{C}^{-1} \right. \\ \left. -(\nu+N+2)\boldsymbol{C}^{-1} + \boldsymbol{C}^{-1}\boldsymbol{\Psi}\boldsymbol{C}^{-1}\right) \end{bmatrix} \tag{22}$$

According to [3], the Fisher information matrix $F(\boldsymbol{\theta})$ with respect to the multivariate normal distribution parameter and its inverse matrix are, respectively,

$$F(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{C}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{1}{2}\boldsymbol{C}^{-1} \otimes \boldsymbol{C}^{-1} \end{bmatrix} \text{ and } F^{-1}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{0} & 2\boldsymbol{C} \otimes \boldsymbol{C} \end{bmatrix},$$

where $\otimes$ is the Kronecker product. Hence, the natural gradient of the log-likelihood of $p(\boldsymbol{\theta})$ is calculated as follows:

$$\tilde{\nabla}_{\boldsymbol{\theta}} \ln p(\boldsymbol{\theta}) = \begin{bmatrix} -\gamma(\boldsymbol{m}-\boldsymbol{\delta}) \\ \mathrm{vec}\left(\gamma(\boldsymbol{m}-\boldsymbol{\delta})(\boldsymbol{m}-\boldsymbol{\delta})^\top + \boldsymbol{\Psi} - (\nu+N+2)\boldsymbol{C}\right) \end{bmatrix} \tag{23}$$

3.4 Update Rules for MAP-IGO with Multivariate Normal Distribution

From [3], the natural gradient of the log-likelihood of $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is calculated as

$$\tilde{\nabla}_{\boldsymbol{\theta}} \ln p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \begin{bmatrix} (\boldsymbol{x}-\boldsymbol{m}) \\ \mathrm{vec}\left((\boldsymbol{x}-\boldsymbol{m})(\boldsymbol{x}-\boldsymbol{m})^\top - \boldsymbol{C}\right) \end{bmatrix} \ . \tag{24}$$

Let $w_i = \hat{w}_i / (\sum_{j=1}^{\lambda} \hat{w}_j)$ for short, and we obtain the update rule from Eq. (20).

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + c_m \left(\sum_{i=1}^{\lambda} w_i (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}^{(t)}) \; -\gamma(\boldsymbol{m}^{(t)} - \boldsymbol{\delta}) \right) \tag{25}$$

$$\begin{aligned} \boldsymbol{C}^{(t+1)} = \boldsymbol{C}^{(t)} + c_\mu \Bigg(& \sum_{i=1}^{\lambda} w_i \left((\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}^{(t)})(x_{i:\lambda} - \boldsymbol{m}^{(t)})^\top - \boldsymbol{C}^{(t)} \right) \\ & +\gamma(\boldsymbol{m}^{(t)} - \boldsymbol{\delta})(\boldsymbol{m}^{(t)} - \boldsymbol{\delta})^\top + \boldsymbol{\Psi} - (\nu + N + 2)\boldsymbol{C}^{(t)} \Bigg) \end{aligned} \tag{26}$$

Shaded terms are differences from the original pure rank-μ update and represent terms corresponding to the natural gradient of the log-likelihood of the prior distribution.

4 Interpretation of the Rank-One Update with Prior Distribution

In the previous section, we introduced the MAP-IGO to incorporate prior distribution into the IGO framework, and derived the update rules of the MAP-IGO in which the multivariate normal distribution and the normal-inverse-Wishart distribution are applied. In this section, we demonstrate that the rank-one update can be derived from the MAP-IGO by setting the prior distribution based on the idea that the promising mean vector should exist in the direction of the evolution path. In addition, we discuss the interpretation to the parameter setting of the prior distribution used to derive the rank-one update.

4.1 Derivation of the Rank-One Update

To confirm that we can derive the rank-one update from the MAP-IGO and reproduce the update rule of the CMA-ES, we first introduce the step-size $\sigma^{(t)}$ into the update rule of the MAP-IGO. By replacing $\boldsymbol{C}^{(t)}$ with $(\sigma^{(t)})^2 \boldsymbol{C}^{(t)}$, Eq. (26) results in the following:

$$\begin{aligned} \boldsymbol{C}^{(t+1)} = \boldsymbol{C}^{(t)} + c_\mu \Bigg(& \sum_{i=1}^{\lambda} w_i \left(\left(\frac{\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}^{(t)}}{\sigma^{(t)}} \right) \left(\frac{\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}^{(t)}}{\sigma^{(t)}} \right)^\top - \boldsymbol{C}^{(t)} \right) \\ & +\gamma \left(\frac{\boldsymbol{m}^{(t)} - \boldsymbol{\delta}}{\sigma^{(t)}} \right) \left(\frac{\boldsymbol{m}^{(t)} - \boldsymbol{\delta}}{\sigma^{(t)}} \right)^\top + \frac{\boldsymbol{\Psi}}{(\sigma^{(t)})^2} - (\nu + N + 2)\boldsymbol{C}^{(t)} \Bigg) \end{aligned} \tag{27}$$

Next, we set the parameters $\boldsymbol{\delta}, \gamma, \boldsymbol{\Psi}$ of the normal-inverse-Wishart distribution. When the expected update direction of the mean vector is obtained as the evolution path $\boldsymbol{p}_c^{(t+1)}$, the promising mean vector can be considered to exist at $\boldsymbol{m}^{(t)} + r\sigma^{(t)} \boldsymbol{p}_c^{(t+1)}$ with $r > 0$. In this situation, we can set $\boldsymbol{\delta} = \boldsymbol{m}^{(t)} + r\sigma^{(t)} \boldsymbol{p}_c^{(t+1)}$,

Algorithm 1: Single update in the MAP-CMA

given $\boldsymbol{m}^{(t)} \in \mathbb{R}^N$, $\sigma^{(t)} \in \mathbb{R}_+$, $\boldsymbol{C}^{(t)} \in \mathbb{R}^{N\times N}$, $\boldsymbol{p}_\sigma^{(t)} \in \mathbb{R}^N$, $\boldsymbol{p}_c^{(t)} \in \mathbb{R}^N$
for $i = 1, \ldots, \lambda$ **do**
  $\boldsymbol{y}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(t)})$
  $\boldsymbol{x}_i \leftarrow \boldsymbol{m}^{(t)} + \sigma^{(t)}\boldsymbol{y}_i$
end for
$\boldsymbol{p}_\sigma^{(t+1)} \leftarrow (1-c_\sigma)\boldsymbol{p}_\sigma^{(t)} + \sqrt{c_\sigma(2-c_\sigma)\mu_\mathrm{w}}\boldsymbol{C}^{(t)^{-\frac{1}{2}}} \sum_{i=1}^{\lambda} w_i \boldsymbol{y}_{i:\lambda}$
$\boldsymbol{p}_c^{(t+1)} \leftarrow (1-c_c)\boldsymbol{p}_c^{(t)} + \sqrt{c_c(2-c_c)\mu_\mathrm{w}} \sum_{i=1}^{\lambda} w_i \boldsymbol{y}_{i:\lambda}$
$\boldsymbol{m}^{(t+1)} \leftarrow \boldsymbol{m}^{(t)} + c_m \left(\sum_{i=1}^{\lambda} w_i (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}^{(t)}) + \frac{c_1}{r c_\mu} \sigma^{(t)} \boldsymbol{p}_c^{(t+1)} \right)$
$\boldsymbol{C}^{(t+1)} \leftarrow \boldsymbol{C}^{(t)} + c_1 \left(\boldsymbol{p}_c^{(t+1)} \boldsymbol{p}_c^{(t+1)^\top} - \boldsymbol{C}^{(t)} \right) + c_\mu \sum_{i=1}^{\lambda} w_i \left(\boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^\top - \boldsymbol{C}^{(t)} \right)$
$\sigma^{(t+1)} \leftarrow \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|}{\mathbb{E}[\|\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\|]} - 1 \right) \right)$

which results from the fact that the expected value of the mean vector in the normal-inverse-Wishart distribution is given by $\boldsymbol{\delta}$. To match the coefficients in Eq. (6) and Eq. (27), we set

$$\gamma = \frac{c_1}{r^2 c_\mu} \qquad \text{and} \qquad \boldsymbol{\Psi} = \left(\nu + N + 2 - \frac{c_1}{c_\mu} \right) (\sigma^{(t)})^2 \boldsymbol{C}^{(t)} \ . \tag{28}$$

Therefore, we obtain the update rule of the mean vector and covariance matrix.

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + c_m \left(\sum_{i=1}^{\lambda} w_i (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}^{(t)}) + \frac{c_1}{r c_\mu} \sigma^{(t)} \boldsymbol{p}_c^{(t+1)} \right) \tag{29}$$

$$\boldsymbol{C}^{(t+1)} = \boldsymbol{C}^{(t)} + c_\mu \left(\sum_{i=1}^{\lambda} w_i \left(\boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^\top - \boldsymbol{C}^{(t)} \right) + \frac{c_1}{c_\mu} \left(\boldsymbol{p}_c^{(t+1)} \boldsymbol{p}_c^{(t+1)^\top} - \boldsymbol{C}^{(t)} \right) \right) \tag{30}$$

We note that $\boldsymbol{y}_{i:\lambda} = (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}^{(t)})/\sigma^{(t)}$ as defined in Eq. (2). While the rank-one update of the covariance matrix is reproduced, the term $\frac{c_m c_1}{r c_\mu} \sigma^{(t)} \boldsymbol{p}_c^{(t+1)}$ appears in the mean vector update rule. As r approaches infinity, Eq. (29) converges to the original update rule of CMA-ES shown in Eq. (3), whose interpretation is discussed in Sect. 4.2. When r takes a finite value, the term $\frac{c_m c_1}{r c_\mu} \sigma^{(t)} \boldsymbol{p}_c^{(t+1)}$ can be interpreted as a kind of *momentum*, which is used in gradient descent methods. We call this additional term *momentum update* in this study. In fact, Li and Zhang [18] show that the evolution path accumulates natural gradients with respect to the mean vector and acts as a momentum under stationary condition. They also show that the outer product of the evolution path serves as a rank-one momentum term for the covariance matrix. Our

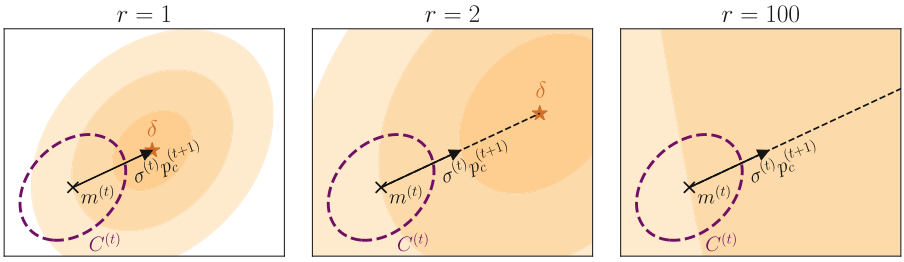


Fig. 1. The prior distribution with respect to the mean vector $\mathcal{N}\left(\boldsymbol{m} \mid \boldsymbol{\delta}, \frac{1}{\gamma}\boldsymbol{C}\right)$, where $\boldsymbol{\delta}$ and $\frac{1}{\gamma}\boldsymbol{C}$ are indicated by the orange star and ellipse, respectively. Since $\boldsymbol{\delta} = \boldsymbol{m}^{(t)} + r\sigma^{(t)}\boldsymbol{p}_c^{(t+1)}$ and $\frac{1}{\gamma} \propto r^2$, multiplying r by a constant $r' > 1$ corresponds to expanding $\mathcal{N}\left(\boldsymbol{m} \mid \boldsymbol{\delta}, \frac{1}{\gamma}\boldsymbol{C}\right)$ by r' times around $\boldsymbol{m}^{(t)}$.

interpretation suggests that the rank-one update with the momentum update is more rational than the original rank-one update in that it is derived in terms of natural gradients. In addition, it can be said that the momentum update is conducive to achieving the goal of the rank-one update, which is to generate a solution in the direction of the evolution path.

Algorithm 1 shows the single-update procedure of the CMA-ES with the momentum update, named maximum a posterior CMA-ES (MAP-CMA). In this study, the MAP-CMA employs the cumulative step-size adaptation, which differs from the CMA-ES only by the momentum update, the shaded term. The effect of the momentum update with finite r is investigated in Sect. 5.

4.2 Interpretation for the Setting of the Prior Distribution

In the normal-inverse-Wishart distribution, $\boldsymbol{\delta}$ and γ are the expected value and global variance of the mean vector, respectively. Multiplying r by a constant $r' > 1$ corresponds to the affine transformation of expanding the prior distribution with respect to the mean vector by r' times around $\boldsymbol{m}^{(t)}$ as shown in Fig. 1. Thus, as r increases, the prior distribution with respect to the mean vector $\mathcal{N}\left(\boldsymbol{m} \mid \boldsymbol{\delta}, \frac{1}{\gamma}\boldsymbol{C}\right)$ approaches a non-informative prior distribution. Then, as r increases, the effect of the prior distribution on the mean vector update decreases.

5 Experiments

In the previous section, we derived the rank-one update by setting the prior distribution. Moreover, we showed that it is extensible by setting r to a finite value and proposed MAP-CMA, a CMA-ES equipped with it. In this section, we investigate the behavior of the MAP-CMA varying r. The code of MAP-CMA will be made available at https://github.com/CyberAgentAILab/cmaes [19].

Table 1. Benchmark functions to be minimized.

Name	Definition
Sphere	$f(\boldsymbol{x}) = \sum_{i=1}^{N} x_i^2$
Ellipsoid	$f(\boldsymbol{x}) = \sum_{i=1}^{N} 10^{6\frac{i-1}{N-1}} x_i^2$
Cigar	$f(\boldsymbol{x}) = x_1^2 + \sum_{i=2}^{N} 10^6 x_i^2$
Rosenbrock	$f(\boldsymbol{x}) = \sum_{i=1}^{N-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$
Ackley	$f(\boldsymbol{x}) = 20 - 20\exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right) + e - \exp\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right)$
Rastrigin	$f(\boldsymbol{x}) = 10N + \sum_{i=1}^{N}\left(x_i^2 - 10\cos(2\pi x_i)\right)$

5.1 Experimental Setting

The benchmark functions are summarized in Table 1. The initial mean vector $\boldsymbol{m}^{(0)}$ was drawn uniform randomly from $[a, b]^N$ for each trial, and the initial step-size $\sigma^{(0)}$ was given by $(b-a)/2$, where (a, b) was given by $(1, 5)$ for Sphere, Ellipsoid, Cigar, and Rastrigin, $(-2, 2)$ for Rosenbrock, $(1, 30)$ for Ackley. The initial covariance matrix $\boldsymbol{C}^{(0)}$ was given by an identity matrix.

In the CMA-ES, c_m was set to 1. In the MAP-CMA, c_m was set to $1/(1 + c_1/(c_\mu r))$ ensuring $c_m + c_m c_1/(r c_\mu) = 1$. The population size λ of the CMA-ES and MAP-CMA was set to $4 + \lfloor 3 \ln N \rfloor$ for Sphere, Ellipsoid, Cigar, Rosenbrock, and Ackley, and for Rastrigin, it was set to $\{700, 1400, 2100, 2800\}$ for dimensions $\{10, 20, 40, 80\}$ so that the success rate for the CMA-ES was sufficiently high, referring to [15]. The other hyperparameters of the CMA-ES and MAP-CMA were set to those in [14, Table 2].

Each trial was terminated and regarded as a success if the best evaluation value reached less than 10^{-10}. Each trial was terminated and regarded as a failure if any of the following conditions were met: the number of evaluations reached more than $10^6 N$; the minimum eigenvalue of $(\sigma^{(t)})^2 \boldsymbol{C}^{(t)}$ became less than 10^{-30}. For each setting, the 100 independent trials were conducted.

5.2 Results and Discussion

Table 2 shows the success rate and SP1 [5] over 100 independent trials. We note that the SP1 index is defined as the average evaluation counts in successful trials divided by the success rate. The behavior of the MAP-CMA approaches that of the CMA-ES as r increases. When r is too small and the effect of the old samples accumulated in the evolution path is too large, the momentum update in the MAP-CMA tends to have a negative effect on optimization. When $r = N$, the MAP-CMA is superior to the CMA-ES for Rosenbrock and competitive for the other functions. Figure 2 shows the transitions of the best evaluation value on the 100 independent trials. While the difference in performance between the CMA-ES and the MAP-CMA is slight for Sphere, Ellipsoid, Cigar, and Rastrigin,

Table 2. Success rate (SR) and SP1 over 100 independent trials.

Function	N	CMA-ES		MAP-CMA					
				$r = 1$		$r = \sqrt{N}$		$r = N$	
		SR	SP1	SR	SP1	SR	SP1	SR	SP1
Sphere	10	1.00	**1787**	1.00	2399	1.00	1961	1.00	1824
	20	1.00	**3329**	1.00	5034	1.00	3617	1.00	3366
	40	1.00	**6276**	1.00	10719	1.00	6705	1.00	6322
	80	1.00	**11568**	1.00	24518	1.00	12246	1.00	11658
Ellipsoid	10	1.00	6078	1.00	6888	1.00	6215	1.00	**6050**
	20	1.00	18906	1.00	21486	1.00	18984	1.00	**18853**
	40	1.00	68401	1.00	74256	1.00	**67205**	1.00	68056
	80	1.00	265855	1.00	284614	1.00	**259434**	1.00	264651
Cigar	10	1.00	**4423**	1.00	5623	1.00	4768	1.00	4506
	20	1.00	**8693**	1.00	12314	1.00	9256	1.00	8787
	40	1.00	**16976**	1.00	26003	1.00	17738	1.00	17068
	80	1.00	**31951**	1.00	58000	1.00	33227	1.00	32098
Rosenbrock	10	0.93	6972	0.91	7039	0.91	6825	0.94	**6802**
	20	0.90	24015	0.80	24152	0.92	**20599**	0.91	22824
	40	0.93	87489	0.76	80442	0.89	**77235**	0.94	82818
	80	0.91	354686	0.80	**258692**	0.92	287489	0.91	344089
Ackley	10	0.97	**3719**	0.73	6734	0.94	4248	0.96	3964
	20	0.95	**6946**	0.60	17122	0.96	7555	0.96	6991
	40	1.00	**12143**	0.31	69112	0.98	13318	1.00	12316
	80	1.00	**21974**	0.00	-	0.91	25775	0.99	22314
Rastrigin	10	0.99	50781	1.00	**50113**	0.98	50722	0.97	51602
	20	1.00	**167412**	1.00	168168	1.00	167748	1.00	168196
	40	1.00	**606858**	0.99	620722	1.00	610302	1.00	608874
	80	1.00	2110892	1.00	2127692	1.00	**2096612**	1.00	2106244

the MAP-CMA improves the best evaluation values more efficiently than the CMA-ES for Rosenbrock.

Figure 3 shows the transition of the mean vector when optimizing Rosenbrock with $N = 20$. The initial mean vector and step-size were given as $\boldsymbol{m}^{(0)} = (1, \ldots, 1)$ and $\sigma^{(0)} = 1$, respectively. Once the mean vector reaches near the origin, it moves to $(1, \ldots, 1)$. At this stage, the momentum update in the mean vector update allows the MAP-CMA to move the mean vector faster than the CMA-ES. This result suggests that the MAP-CMA can be more efficient than the CMA-ES when the required mean vector moves are large.

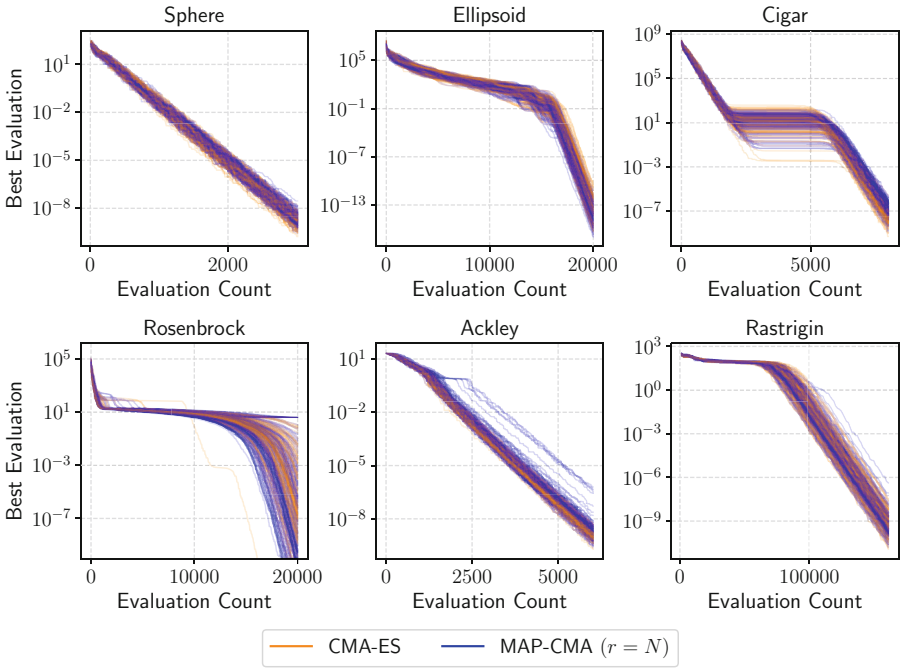


Fig. 2. Transitions of best evaluation value for $N = 20$ over 100 independent trials.

When optimizing Ackley with the MAP-CMA, the optimization stalls in some trials. This stagnation may be due to the effect of past samples [22], which is an issue to be investigated in the future.

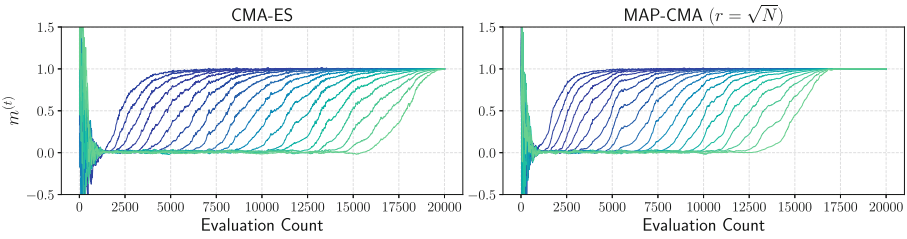


Fig. 3. Transition of mean vector in one typical trial of optimizing Rosenbrock with $N = 20$.

6 Conclusion

We provided a new interpretation of the rank-one update in the CMA-ES from the perspective of the natural gradient with prior distribution. We firstly proposed maximum a posteriori IGO (MAP-IGO), which is the IGO framework extended to incorporate a prior distribution, and derived the update rules of the MAP-IGO in which the multivariate normal distribution and the normal-inverse-Wishart distribution are applied. Then, we derived the rank-one update from the MAP-IGO by setting the prior distribution based on the idea that the promising mean vector should exist in the direction of the evolution path. Furthermore, the newly derived rank-one update is extensible, and we proposed maximum a posterior CMA-ES (MAP-CMA), a CMA-ES equipped with it. The mean update rule in the MAP-CMA has an additional term containing the evolution path, which can be interpreted as a momentum term as discussed in [18] and we call this additional term *momentum update* in this study. Experimental results showed that the MAP-CMA is better than the CMA-ES in optimizing functions that require a lot of mean vector moves, such as Rosenbrock.

In this study, the coefficients of the prior distribution were set so that the derived rank-one update matched that of the original CMA-ES. However, there is room for adjustment of these coefficients and the current hyperparameters of the MAP-CMA because some trials stagnate in the multimodal function. The detailed investigation of these settings is future work. In addition, it is important to research the effectiveness of momentum update in the CMA-ES with restart strategies [6,11] or integer handling [9,12]. Furthermore, the MAP-IGO can set the parameter of the prior distribution other than the idea of the rank-one update, and it is also possible to derive the PBIL that can handle the prior distribution by applying the Bernoulli and Beta distributions. The discovery of new knowledge and the improvement of algorithms based on these techniques are also important future works.

References

1. Abdolmaleki, A., Price, B., Lau, N., Reis, L.P., Neumann, G.: Deriving and improving CMA-ES with information geometric trust regions. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 657-664. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3071178.3071252
2. Abdolmaleki, A., Springenberg, J.T., Tassa, Y., Munos, R., Heess, N., Riedmiller, M.: Maximum a posteriori policy optimisation. In: International Conference on Learning Representations (2018)
3. Akimoto, Y., Nagata, Y., Ono, I., Kobayashi, S.: Bidirectional relation between CMA evolution strategies and natural evolution strategies. In: Parallel Problem Solving from Nature, PPSN XI, pp. 154–163 (2010)
4. Amari, S.I., Nagaoka, H.: Methods of Information Geometry, vol. 191 (2000)
5. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1769–1776. IEEE (2005)

6. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size, vol. 2, pp. 1769–1776 (2005). https://doi.org/10.1109/CEC.2005.1554902
7. Baluja, S.: Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning (1994). https://api.semanticscholar.org/CorpusID:14799233
8. Bishop, C.M., Nasrabadi, N.M.: Pattern Recognition and Machine Learning, vol. 4. Springer, Heidelberg (2006)
9. Hamano, R., Saito, S., Nomura, M., Shirakawa, S.: CMA-ES with margin: lower-bounding marginal probability for mixed-integer black-box optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2022, pp. 639-647. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3512290.3528827
10. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 312–317 (1996). https://doi.org/10.1109/ICEC.1996.542381
11. Hansen, N.: Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, GECCO 2009, pp. 2389-2396. Association for Computing Machinery, New York (2009). https://doi.org/10.1145/1570256.1570333
12. Hansen, N.: A CMA-ES for mixed-integer nonlinear optimization (2011)
13. Hansen, N.: The CMA evolution strategy: a tutorial. arXiv preprint arXiv:1604.00772 (2016)
14. Hansen, N., Auger, A.: Principled design of continuous stochastic search: from theory to practice, pp. 145–180 (2014). https://doi.org/10.1007/978-3-642-33206-7_8
15. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 282–291. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_29
16. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol. Comput. **11**(1), 1–18 (2003). https://doi.org/10.1162/106365603321828970
17. Harik, G., Lobo, F., Goldberg, D.: The compact genetic algorithm. IEEE Trans. Evol. Comput. **3**(4), 287–297 (1999). https://doi.org/10.1109/4235.797971
18. Li, Z., Zhang, Q.: What does the evolution path learn in CMA-ES? In: Parallel Problem Solving from Nature – PPSN XIV, pp. 751–760 (2016)
19. Nomura, M., Shibata, M.: cmaes: a simple yet practical python library for CMA-ES. arXiv preprint arXiv:2402.01373 (2024)
20. Ollivier, Y., Arnold, L., Auger, A., Hansen, N.: Information-geometric optimization algorithms: a unifying picture via invariance principles. J. Mach. Learn. Res. **18**(1), 564–628 (2017)
21. Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. J. Global Optim. **56**, 1247–1293 (2013)

22. Shirakawa, S., Akimoto, Y., Ouchi, K., Ohara, K.: Sample reuse in the covariance matrix adaptation evolution strategy based on importance sampling. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 305–312 (2015)
23. Song, H.F., et al.: V-mpo: on-policy maximum a posteriori policy optimization for discrete and continuous control. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=SylOlp4FvH

Avoiding Redundant Restarts in Multimodal Global Optimization

Jacob de Nobel[1], Diederick Vermetten[1(✉)], Anna V. Kononova[1], Ofer M. Shir[2], and Thomas Bäck[1]

[1] LIACS, Leiden University, Leiden, The Netherlands
d.l.vermetten@liacs.leidenuniv.nl
[2] Tel-Hai College and Migal Institute, Upper Galilee, Israel

Abstract. Naïve restarts of global optimization solvers when operating on multimodal search landscapes may resemble the Coupon's Collector Problem, with a potential to waste significant function evaluations budget on revisiting the same basins of attractions. In this paper, we assess the degree to which such "duplicate restarts" occur on standard multimodal benchmark functions, which defines the *redundancy potential* of each particular landscape. We then propose a repelling mechanism to avoid such wasted restarts with the CMA-ES and investigate its efficacy on test cases with high redundancy potential compared to the standard restart mechanism.

Keywords: numerical optimization · multimodal landscapes · CMA-ES

1 Introduction

Finding an effective balance between exploring the domain and exploiting promising regions is one of the primary challenges when designing any iterative optimization heuristic, being subject to an underlying hard conflict. This design choice is especially important in multimodal search landscapes, where premature convergence leads to poor overall performance.

Because of its challenging nature, a large number of algorithms have been developed for multimodal optimization [22], yet consisting of two main branches with different goals—(i) niching methods [24], whose target is adjusted from locating the globally optimal solution to finding a wider set of high-quality optima, and (ii) "upgraded global solvers", which still target a single global optimum, but are better equipped to handle multimodality (see, e.g., [3,9]).

While niching methods offer multiple approaches to treating such multimodal landscapes, primarily in promoting population diversity, these practices are usually not transferred to global optimization for various reasons (for instance, since focusing on solution diversity can hamper performance by slowing down convergence). Instead, global solvers often rely on rebalancing the search from global to local over time, e.g., by adjusting step sizes in evolutionary algorithms. To

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 268–283, 2024.
https://doi.org/10.1007/978-3-031-70068-2_17

prevent convergence to sub-optimal solutions in these methods, restart mechanisms can be utilized to reset the algorithm's state, essentially starting a new optimization process by using the remaining function evaluation budget. This process is usually independent of the previous trajectory but can update the algorithm's strategy parameters, such as the population size within CMA-ES.

However promising a restart scheme can be, it still carries the potential to encounter the equivalent of the **Coupon Collector's Problem**, which is rooted in the retrials' naïvity. Given q unique coupons, the expected number of trials needed to collect them all, with replacement, is $q \cdot H_q$ (with $H_q := \sum_{t=1}^{q} \frac{1}{t}$ being the q^{th}-harmonic number). Similarly, in the context of multimodal optimization, a straightforward approach of *iteration* can be used to locate sequentially multiple peaks in the landscape by means of an *iterative local search* (ILS) [14]. If the procedure is blind to any information accumulated throughout previous runs, and it sequentially restarts stochastic search processes, the ambition to hit a different peak in every run resembles the collector's hope to obtain all the coupons in only q trials. Overall, it is likely to encounter *redundancy*, and the number of expected iterations is then increased by a factor. A **redundancy factor** can be derived if the peaks are of equal height (an *equi-fitness landscape*, i.e., the probability to converge into any of the q peaks is uniform and equal to $1/q$), simply by normalizing the expected number of trials with respect to q - that is an overall redundancy factor of H_q [24]. Importantly, when dropping the equi-fitness assumption, this factor is expected to increase.

Mind should be given to this analogy and to the actual target of the "collector": the careful reader must note that a *global* multimodal solver is not necessarily concerned with "collecting" all the optima (as in niching), since it is targeted at attaining only the best (i.e., picking only the "top coupon"). At the same time, when operating in a black-box fashion, the global quality of such optima is often *undecidable*. Therefore, the "collector" has no choice but to start a campaign of iterative restarts.

The idea of "avoiding duplicates" is a fundamental concept in heuristic search (e.g., the classical Tabu Search [5]), and even in statistical sampling (e.g., the conditioned Latin Hypercube Sampling that produces sampling designs while avoiding redundancy by accounting for external information [16]). To investigate this idea in the context of multimodal optimization, we propose an alternative restart strategy in the context of the renowned CMA-ES algorithm.

We provide the necessary background, especially concerning basins of attraction of local optima and an illustrative motivation, in Sect. 2, and identify the potential evaluation redundancy problem caused by restarts in global optimization. Then, we analyze the extent of this redundancy phenomenon on standard continuous landscapes, which motivates our proposal for an alternative restart strategy (Sect. 3). We propose and benchmark a restart strategy that integrates ideas from multimodal optimization, specifically *repelling subpopulations in niching*, with the existing restart practices of CMA-ES. This new algorithm, the *CMA-ES with repelling restarts* (RR-CMA-ES), is introduced in Sect. 4. This method is benchmarked on a wide set of problems, which shows that while

wasted evaluations from "duplicate" restarts can be avoided, the precise methods to achieve this potential must be carefully calibrated to prevent deteriorating performance on some types of landscapes (Sect. 5). Conclusions and future work are discussed in Sect. 6.

2 Preliminaries and Problem Formulation

2.1 Basins of Attraction in Global Optimization

We consider the *global optimization* challenge of a single-objective, continuous minimization problem; the aim is to identify a *single* solution $\boldsymbol{x}^* \in \mathcal{X} \subseteq \mathbb{R}^d$ from the feasible region $\mathcal{X}$, which minimizes a given objective function $f(\boldsymbol{x}) : \mathcal{X} \rightarrow \mathbb{R}$:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) \tag{1}$$

When the convexity property does not hold for f [4], it is often the case that other candidate solutions $\boldsymbol{x}_\ell \neq \boldsymbol{x}^*$ minimize certain neighborhoods of radii ϵ, and thus each forms a *local minimum*:

$$\exists \epsilon > 0 \quad \forall \boldsymbol{x} \in \mathcal{X} \ : \ \|\boldsymbol{x} - \boldsymbol{x}_\ell\| < \epsilon \Rightarrow f(\boldsymbol{x}_\ell) \leq f(\boldsymbol{x})$$

$\boldsymbol{x}^*$ is called the *global optimizer*. Accordingly, the process of global optimization [27] is considered successful when it concludes with locating it while escaping *local optimizers* $\boldsymbol{x}_\ell$ (also referred to as *traps*). Next, we would like to define the attraction basin of an optimizer (denoted as $\hat{\boldsymbol{x}}$, being either local or global), following [27] (alternative definitions exist - see, e.g., [2]). Given the standard Gradient Descent Algorithm [4], which is defined by its variation step from iteration i to $i+1$ (with $\sigma(i)$ being the step-size),

$$\boldsymbol{x}(i+1) := \boldsymbol{x}(i) - \sigma(i) \cdot \nabla f\left(\boldsymbol{x}(i)\right), \tag{2}$$

we denote its initial search-point $\boldsymbol{x}(0) := \boldsymbol{x}_0$. The following set of points is defined whenever the limit $\lim_{i \rightarrow \infty} \boldsymbol{x}(i)$ exists:

$$\Omega = \left\{ \boldsymbol{x} \in \mathbb{R}^d \,\middle|\, \boldsymbol{x}(0) = \boldsymbol{x} \wedge \boldsymbol{x}(i)|_{i \geq 0} \text{ satisfies (2)} \wedge \lim_{i \rightarrow \infty} \boldsymbol{x}(i) \text{ exists} \right\} \tag{3}$$

Then, given an optimizer $\hat{\boldsymbol{x}}$, we define its region of attraction using Ω:

$$\mathcal{A}(\hat{\boldsymbol{x}}) = \left\{ \boldsymbol{x} \in \Omega \,\middle|\, \boldsymbol{x}(0) = \boldsymbol{x} \wedge \boldsymbol{x}(i)|_{i \geq 0} \text{ satisfies (2)} \wedge \lim_{i \rightarrow \infty} \boldsymbol{x}(i) = \hat{\boldsymbol{x}} \right\} \tag{4}$$

Finally, and most importantly, the *basin* of $\hat{\boldsymbol{x}}$ is the **maximal level set** that is fully contained in $\mathcal{A}(\hat{\boldsymbol{x}})$.

We are particularly interested in addressing this global optimization challenge within black-box optimization, where the analytical form of the objective function f is unknown to the solver and is thus treated as a black box, which, upon receiving input, yields a (continuous) output.

2.2 CMA-ES

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10] is a state-of-the-art method for single objective black-box optimization. The CMA-ES is a stochastic method that evolves a population of candidate solutions $\mathbf{x}$ to an optimization problem $f : \mathbb{R}^d \to \mathbb{R}$. To guide the search, it uses a parameterized multivariate normal distribution $\mathcal{N}(\mathbf{m}, \sigma\mathbf{C})$. The defining feature of the CMA-ES is that it can adapt the parameters of its mutation distribution to an arbitrary shape during optimization to guide the search. This makes it invariant to linear transformations of the search space and underpins its effectiveness on non-separable and ill-conditioned problems.

2.3 Restart Mechanisms

Restarts are commonly employed to prevent premature convergence in optimization algorithms. A whole class of algorithms relies on restart mechanisms to adapt local search methods to global optimization problems, dubbed as *Iterative Local Search* (ILS) [14]. While ILS is mostly prevalent in discrete domains, techniques for continuous domains such as Multi-Level Single Linkage (MLSL) [23] have shown promising performance on commonly used benchmark problems [21]. While restart methods are common in combination with local search, they can also be applied to more global optimizers, such as evolutionary algorithms. In its simplest form, for the CMA-ES, restarting involves resetting its mutation distribution to a standard multivariate normal distribution and setting the center of mass $\mathbf{m}$ to a new location u.a.r. whenever *local* restart criteria have been met. Additionally, the algorithm's parameters can be altered to change its behavior following a restart, of which the most well-known strategies are IPOP [3] and BIPOP [6]. The IPOP-CMA-ES increases its population size by a factor of 2 on every consecutive restart. The BIPOP-CMA-ES alternates between two regimes; the first uses an increased population size and a larger value for the initial step size σ^0, while the second uses a smaller population size and σ^0. While other restart strategies have been proposed [13,17], IPOP and BIPOP remain among the most commonly used methods.

2.4 Motivation

While the aforementioned restart mechanisms have been shown to be effective on multimodal problems [6], no inherent method is implemented to ensure a subsequent restart does not converge to the same basin of attraction. By restarting u.a.r. it could very well be possible that several runs will be drawn to the same attractor. When this happens, the restart effectively spends more of the evaluation budget to potentially find an already identified solution. If this could be avoided by ensuring each consecutive restart only explores a previously unvisited region of the search space, we could potentially save a portion of the evaluation budget.

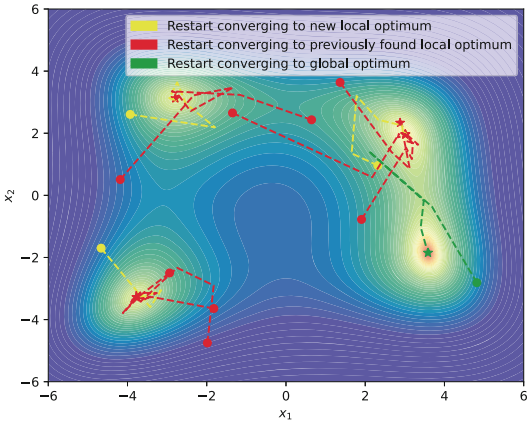


Fig. 1. A single run of the CMA-ES with a simple u.a.r. restart strategy ($\sigma^0 = 2$), optimizing a modified version of the Himmelblau function (f_{mh}, see Eq. 5). Every restart is visualized as a line, which shows the trace followed by the CMA-ES, where the circle shows the start of the trace and the star shows the finally obtained solution. The restart that converges to the global optimum is shown in green, and restarts converging to a new local optimum are shown in yellow. The traces in red show restarts that converge to local optima, which have already been found during previous restarts (yellow). (Color figure online)

Consider the example shown in Fig. 1, which shows a single run of the CMA-ES with a simple restart strategy solving a modified version of the 2-dimensional Himmelblau function:

$$f_{mh}(\mathbf{x}) = (x_1^2 + x_2 - 11) + (x_1 + x_2^2 - 7)^2 + \min\{0.01, ||\mathbf{x} - \mathbf{x}^*||_2\} \qquad (5)$$

Here, $\mathbf{x}^*$ is the location of the global optimum, and $||\cdot||_2$ denotes the Euclidean norm. From the figure, we can observe that the CMA-ES restarts several times during the optimization process. Only during one of the restarts does the CMA-ES converge to the global optimum (green trace), while the other three local optima are found during other restarts. Notably, it can be observed that a considerable number of restarts are converging to local optima that have already been identified (red traces). In fact, this is the case for eight out of twelve restarts for this specific run. If we had been able to avoid revisiting previously found optima, considerable savings to the evaluation budget could have been realized. While one might argue that this is just a bad run or that a change to the initial σ could avoid this type of behavior, we aim to demonstrate in the next section that this is not the case.

2.5 Repelling Subpopulations

The CMSA with *Repelling Subpopulations* (RS-CMSA) [1] has been proposed as a niching strategy that uses a number of parallel subpopulations to find multiple

local optima in multimodal optimization. Our approach is inspired by the RS-CMSA, but it focuses on avoiding repeatedly sampling points in the basin of attraction of a local optimum that the algorithm has already identified. The overall goal is to make the CMA-ES more sample-efficient for the goal of finding a single best solution (i.e., the best local optimum approximation, given a budget B of function evaluations), rather than multiple solutions representing different local optima (i.e., multiple niches). The underlying methods, such as the usage of the Hill-Valley heuristic, restarts, and tabu regions, are inspired by the RS-CMSA.

2.6 Hill-Valley Function as a Boolean Heuristic

The Hill-Valley function $\texttt{HV}(\mathbf{x}^i, \mathbf{x}^j, f) \in \{0, 1\}$, listed as Algorithm 1, introduced as a part of the multinational EA [28], can be used as a heuristic to determine whether two points $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$ belong to the same basin of attraction [15]. It calculates the objective function value for a maximum of $N_t = 10$ points on a line drawn between $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$. If any of these points are of a higher objective function value, assuming minimization, a hill is assumed to be present between $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$, which then do not share a basin of attraction.

Algorithm 1. Hill-Valley test $\texttt{HV}(\mathbf{x}^i, \mathbf{x}^j, f)$

Require: $\boldsymbol{x}^i \in \mathbb{R}^d, \boldsymbol{x}^j \in \mathbb{R}^d, N_t \in \mathbb{N}, f : \mathbb{R}^d \to \mathbb{R}$
  for $k \in 1, \ldots, N_t$ **do**
    $\boldsymbol{x}^{test} = \boldsymbol{x}^i + \frac{k}{N_t+1}(\boldsymbol{x}^j - \boldsymbol{x}^i)$
    if $\max\{f(\boldsymbol{x}^i), f(\boldsymbol{x}^j)\} \leq f(\boldsymbol{x}^{test})$ **then**
      return 0
    end if
  end for
  return 1

3 The Potential Gain of Avoiding Redundant Restarts

3.1 Defining the Redundancy Measure

Motivated by the *redundancy factor* mentioned earlier in the context of the Coupon's Collector Problem, we would like to define a measure that quantifies the gain potential for having effective restarts. The trials of the Coupon's Collector must be adapted to the notion of restarts and to account for the global optimum, which constitutes the "top priority coupon" when following the analogy. To this end, we define the *restarts' redundancy factor* as the proportion of function evaluations spent by *duplicate restarts* (different restarts that converged to a basin of attraction that was previously visited). This calculation excludes function evaluations spent within the global basin (regardless of being duplicates),

denoted as $\mathbf{x}^*$ – and it accumulates such function evaluations and normalizes them by the total budget. With r denoting the current number of restarts, having the current restart converging to a point $\mathbf{x}^{(r)}$, we use $\text{red}(\mathbf{x}^{(r)}) \in \{0, 1\}$ to determine whether the restart is redundant, relying on an oracle that answers the boolean query `sameBasin()` indicative of two points sharing a basin of the function f:

$$\text{red}(\mathbf{x}^{(r)}) := \left[\neg \texttt{sameBasin}(\mathbf{x}^*, \mathbf{x}^{(r)}, f))\right] \wedge \left[\bigvee_{1 \leq k < r} \texttt{sameBasin}(\mathbf{x}^{(k)}, \mathbf{x}^{(r)}, f)\right] \quad (6)$$

Importantly, the previously defined Hill-Valley heuristic may approximate the boolean query `sameBasin()`, and will play this role in our implementation.

We then define the *restarts' redundancy factor* as the normalized accumulated redundancy of a given set of restarts:

$$\text{RRF}\left(\left\{\mathbf{x}^{(r)}\right\}_{r=1}^{R}\right) := \frac{\sum_{r=1}^{R} \text{red}(\mathbf{x}^{(r)}) b^{(r)}}{B}, \quad (7)$$

where $b^{(r)}$ is the number of function evaluations spent by a restart, R is the total number of completed restarts, and B is the total budget of function evaluations spent. Importantly, this factor accounts for a given set of restarts rather than measuring a search algorithm/mechanism (i.e., an implicit algorithmic measure).

3.2 Numerical Assessment of Expected Redundancy Factors

We analyze the CMA-ES with three different restart strategies for several benchmark functions to identify the potential performance gains from avoiding duplicated restarts in global optimization. We compare a naive restart strategy (labeled 'RESTART') alongside the popular IPOP and BIPOP restart strategies. Each strategy places the center of mass $\boldsymbol{m}$ uniformly at random in the domain on each restart and uses saturation for bound correction. The remaining settings of CMA-ES are left as default in the modCMA package [19]. For each benchmark function, we perform 100 independent runs. In addition to logging the performance trajectories with IOHexperimenter [20], we log the center of mass and the number of evaluations used each time a restart is triggered. Based on this information, we can use Eq. 7 to calculate the RRF for a given experiment.

BBOB. We benchmark on the noiseless, single-objective BBOB suite [8]. Originally proposed as part of the COCO benchmarking platform [7], this set of 24 continuous optimization problems has been one of the last decade's most commonly used benchmarks for iterative optimization heuristics. For completeness, we perform our redundancy analysis on all 24 BBOB functions, even though a relatively large fraction consists of unimodal problems or problems with strong global structures where we don't expect to see any redundant restart. We run each problem for dimensionality $d \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 20\}$ for 10 instances.

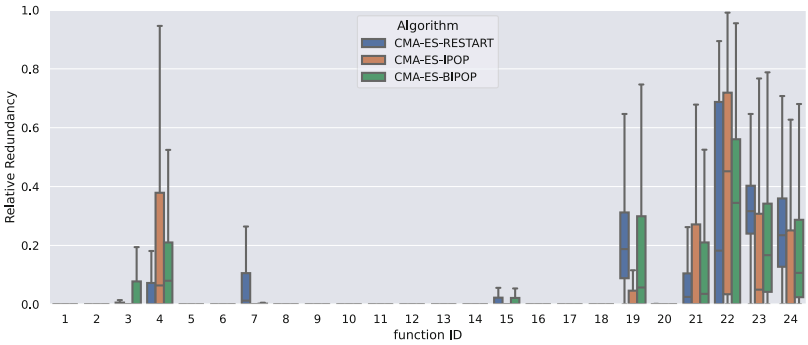


Fig. 2. Boxplot showing the fraction of the total budget spent by restarts converging to a previously visited basin of attraction, the RRF (Eq. 7). All 24 objective functions from the BBOB benchmark are aggregated over all dimensions and instances for each tested restart strategy.

Figure 2 shows the RRF for each function, aggregated over all dimensions, runs, and instances for each of the three restart strategies. From the figure, as expected, most redundant restarts occur in functions f_{21}–f_{24}, multimodal functions with weak global structure. We expect this is related to the weak structure since the chance we go to any given optima over one of the others should be related to the 'direction' of the function's global structure. Since there is no global structure and several local optima of equal height, we would be equally likely to end up in any of them randomly.

The other functions showing some redundancy in Fig. 2 are f_3, f_4, f_{15} and f_{19}, which also exhibit some multimodality. Notably, f_7 is an interesting outlier, which, due to its plateauing landscape, causes several restarts for the 'RESTART' strategy to be classified as redundant. Figure 3 provides a closer look at the functions with any redundant restarts, specifically showing the relation between dimensions for the 'RESTART' strategy. Here we can observe that for some functions, such as f_{19} and f_{22}, the RRF seems to increase with dimension; the opposite is true for functions f_3 and f_4.

CEC 2013. While the overall potential to be gained from avoiding duplicated restarts is limited on the BBOB suite, we still see relatively large values in some multimodal problems. As such, we extend our setup to the CEC'13 suite [11], which is designed specifically for multimodal optimization. We multiply each function with -1 to enable minimization. To transform the functions to global optimization problems, we select one of the existing global optima to turn into the only global one by adding a norm of distance as follows: $f'(\boldsymbol{x}) = -f(\boldsymbol{x}) + \min(0.01, \|\boldsymbol{x} - \boldsymbol{x}^*\|^2)$. Given this "globalization" procedure, we create 10 instances of each of the 16 CEC problems (where each problem has a predefined dimensionality). The remainder of the setup is equivalent to the one used for the BBOB problems. The resulting redundancy factors are

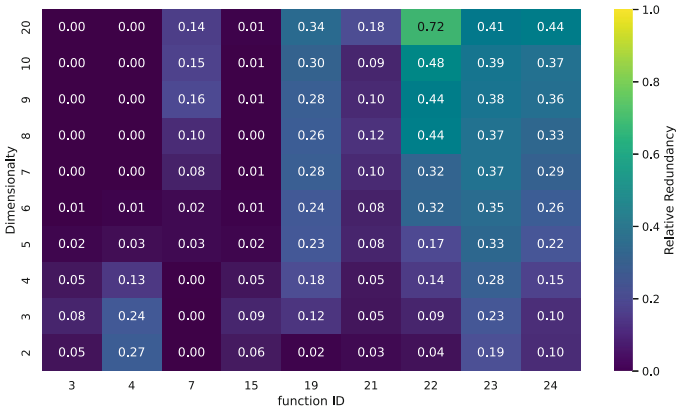


Fig. 3. The average Relative Redundancy Factor over all instances for the CMA-ES using the 'RESTART' strategy for the BBOB functions with any redundant restarts. The grid shows the RRF (Eq. 7) per dimension and function individually.

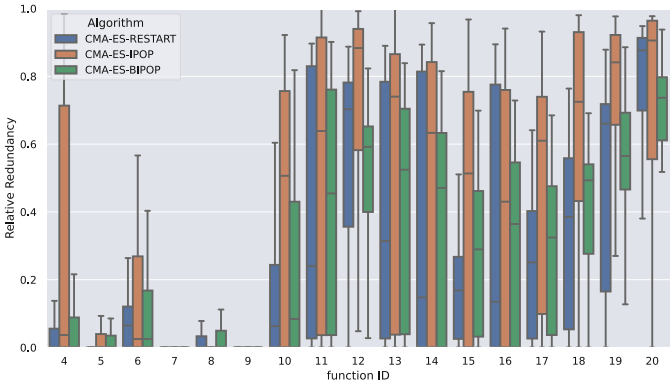


Fig. 4. Fraction of budget which could be saved by avoiding convergence to redundant regions of the search space in subsequent restarts (as defined in Eq. 6), for 16 functions from the CEC 2013 benchmark, aggregated over all instances for runs with the CMA-ES using different restart strategies.

plotted in Fig. 4, from which we observe that most problems defined for multi-modal optimization show significant potential for saving redundant evaluations. We also observe that the distribution for IPOP is much wider and has a higher mean than the other methods. This happens because the population sizes are increased, leading to individual redundant runs containing a larger fraction of the total budget. This also occurs to a lesser extent in BIPOP since the larger population sizes are interleaved with low population size restarts.

Algorithm 2. RR-CMA-ES

Require: $\sigma_0 \in \mathbb{R}, \gamma \in \mathbb{R}, f : \mathbb{R}^d \to \mathbb{R}$
1: $\mathcal{T} \leftarrow \emptyset$ ▷ Tabu point archive
2: $r \leftarrow 0$
3: **while** not happy **do**
4: $\sigma \leftarrow \sigma_0,\ \mathbf{C} \leftarrow \mathbf{I},\ \boldsymbol{m} \sim \mathcal{U}(lb, ub)$
5: **while** not any restart conditions **do**
6: $\mathcal{X} \leftarrow \emptyset,\ n_{\text{rej}} \leftarrow 0$
7: **while** $|\mathcal{X}| < \lambda$ **do**
8: $\boldsymbol{x} \leftarrow \boldsymbol{m} + \mathcal{N}(0, \sigma\mathbf{C})$
9: **if** $(\forall \boldsymbol{T} \in \mathcal{T} : \Delta_{\text{rej}}(\boldsymbol{x}, \boldsymbol{T}, \gamma, n_{\text{rej}}) = 0)$ **then** ▷ Check if $\boldsymbol{x}$ can be accepted
10: $\mathcal{X} \leftarrow \mathcal{X} \cup \{\boldsymbol{x}\}$
11: **else**
12: $n_{\text{rej}} \leftarrow n_{\text{rej}} + 1$ ▷ Increase rejection count for $\boldsymbol{m}$
13: **end if**
14: **end while**
15: $\mathcal{F} \leftarrow \texttt{evaluate}(\mathcal{X})$ ▷ Evaluate all offspring
16: $\texttt{cmaUpdate}(\mathcal{X}, \mathcal{F}, \boldsymbol{m}, \sigma, \mathbf{C})$ ▷ Continue the regular CMA-ES procedure
17: **end while**
18: $r \leftarrow r + 1$
19: $\mathcal{T} \leftarrow \mathcal{T} \cup \{\boldsymbol{m}\}$ ▷ Add $\boldsymbol{m}$ to the tabu point archive
20: **end while**

4 Combating the Redundancy: Repelling CMA-ES

Based on the RS-CMSA [1], which uses the concept of repelling subpopulations in a niching context, we introduce a CMA-ES with repelling restarts (RR-CMA-ES). Building on the standard restart strategies, the algorithm's idea is to define regions in the space where the CMA-ES cannot sample, as they were already visited during previous restarts. It maintains a set of tabu points $\boldsymbol{T}$, see Sect. 4.1, which describe a region. To determine the shape and size of this rejection region, we use the ideas from the RS-CMSA [1] and use the current covariance matrix $\mathbf{C}$ and step size σ. The size of the region is unique for each tabu point and controlled by $\delta(\boldsymbol{T})$. Algorithm 2 gives a general overview of the method.

4.1 Tabu Points

The CMA-ES samples λ points at every generation g. In the RR-CMA-ES, each newly sampled point $\boldsymbol{x}$ is tested against the archive $\mathcal{T}$ of tabu points before being accepted for evaluation (line 9). A tabu point defines a location in the search space where the optimizer is not allowed to go. While initially imagined for combinatorial optimization, for the continuous spaces the CMA-ES deals with, it defines a hyper-ellipsoid centered around a point where the optimizer cannot sample. Specifically, a tabu point $\boldsymbol{T}$ consists of a triplet $(\boldsymbol{x}^{\boldsymbol{T}}, f(\boldsymbol{x}^{\boldsymbol{T}}), n^{\boldsymbol{T}})$, where $\boldsymbol{x}^{\boldsymbol{T}}$ is the location in the search space with $f(\boldsymbol{x}^{\boldsymbol{T}})$ its corresponding fitness and $n^{\boldsymbol{T}}$ being the number of times the CMA-ES has converged to $\boldsymbol{T}$ during

previous restarts. During sampling, a tabu point $\boldsymbol{T}$ rejects a newly sampled point $\boldsymbol{x}$ according to the following boolean query ($\Delta_{\text{rej}} \in \{0, 1\}$):

$$\Delta_{\text{rej}}(\boldsymbol{x}, \boldsymbol{T}, \gamma, n_{\text{rej}}) = \left(\frac{d_m(\boldsymbol{x}, \boldsymbol{x}^{\boldsymbol{T}}, \mathbf{C}^{-1})}{\sigma} < \gamma^{n_{\text{rej}}} \delta(\boldsymbol{T}) \right) \tag{8}$$

Here, d_m denotes the Mahanolobis distance metric, scaled by the current covariance matrix $\mathbf{C}$ and step size σ, and $\delta(\boldsymbol{T})$ denotes the rejection radius around the tabu point $\boldsymbol{T}$. To avoid stagnation, a shrinkage factor $0 < \gamma < 1$ is applied to $\delta(\boldsymbol{T})$ with n_{rej} denoting the number of times a point has been rejected in the current generation. Then, for every newly sampled point $\boldsymbol{x}$, it is accepted if and only if:

$$\forall \boldsymbol{T} \in \mathcal{T} : \Delta_{\text{rej}}(\boldsymbol{x}, \boldsymbol{T}, \gamma, n_{\text{rej}}) = 0 \tag{9}$$

4.2 Restarting

Upon every restart r, the archive $\mathcal{T}$ is updated with the converged center of mass $\boldsymbol{m}$. Using the Hill Valley `HV` routine, we check whether $\boldsymbol{m}$ has converged to a new basin of attraction. If the restart converged to a new basin of attraction, that is $\texttt{HV}(\boldsymbol{m}, \boldsymbol{x}^{\boldsymbol{T}}, f) = 0, \forall \boldsymbol{T} \in \mathcal{T}$, the new tabu point $(\boldsymbol{m}, f(\boldsymbol{m}), 1)$ gets added to the archive. If the restart converged to a point that is already present in $\mathcal{T}$, $n^{\boldsymbol{T}}$ gets increased by 1, and if $f(\boldsymbol{m}) < f(\boldsymbol{x}^{\boldsymbol{T}})$, $(\boldsymbol{m}, f(\boldsymbol{m}), n^{\boldsymbol{T}})$ replaces $(\boldsymbol{x}^{\boldsymbol{T}}, f(\boldsymbol{x}^{\boldsymbol{T}}), n^{\boldsymbol{T}})$.

4.3 Search Space Coverage

We define a *coverage factor* c, which controls the ratio of the total volume of the search space $S = \prod_{i=1}^{d} (ub_i - lb_i)$ that is covered by the repelling regions of all tabu points. Intuitively, $\frac{1}{c}$ denotes the maximal proportion of S, which is unavailable for the CMA-ES to sample. This is divided amongst all tabu points, where points with a higher $n^{\boldsymbol{T}}$ cover a larger part of this volume. Then, for each tabu point, the volume of the repelling region, normalized for σ_0 is:

$$V(\boldsymbol{T}) = n^{\boldsymbol{T}} \frac{S}{c \sigma_0 R} \tag{10}$$

where R denotes the total number of restarts. This is used to calculate the rejection radius:

$$\delta(\boldsymbol{T}) = V(\boldsymbol{T})^{\frac{1}{d}} \frac{\Gamma(\frac{d}{2} + 1)^{\frac{1}{d}}}{\sqrt{\pi}} \tag{11}$$

where d denotes the dimensionality and $\Gamma(\cdot)$ the gamma function.

5 Proof-of-Concept: Repelling CMA-ES in Action

To illustrate the workings of the RR-CMA-ES, we benchmark several versions with redundancy factors ranging from 2 to 1000 on both the BBOB and CEC'13 functions. We perform 50 independent runs on each of the 10 instances used in

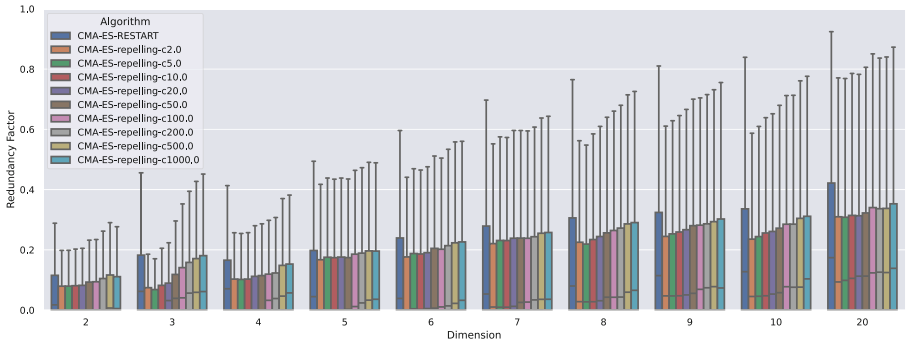


Fig. 5. Distribution of redundant function evaluation over the BBOB functions where redundant restarts were found in Sect. 3.2, i.e., f_3, f_3, f_{15}, f_{19}, f_{21}, f_{22}, f_{23} and f_{24}, separated by problem dimensionality. The CMA-ES with the 'RESTART' strategy is compared to the RR-CMA-ES, with different coverage factors c.

Sect. 3.2 and compare the results to the original CMA-ES. In this section, we use the versions with the default restart mechanism, but the IPOP and BIPOP results are available in our reproducibility repository [18]. To gauge whether the repelling strategy prevents duplicate restarts from occurring, we perform the same expected redundancy factor calculation from Eq. 7 on the runs from the RR-CMA-ES and visualize the results in Fig. 5. From this figure, we can see that on average, the RR-CMA-ES with the lowest coverage factor (the largest repelling regions) quite effectively prevents different restarts from converging to the same basins.

We can perform the same redundancy-based comparisons for the CEC functions, where in Fig. 6, we see how the potential changes when the repelling strategy is used. From this figure, we can see that the impact of the repelling regions is even larger than that of the BBOB problems, but with large variations between the different functions. However, the relation to the coverage factor seems consistent, with a lower factor leading to fewer redundant restarts.

Given the promising reduction in redundant evaluations, we look at the performance of the considered algorithm variants. In Fig. 7, we show the empirical cumulative distribution plot[1] for both the BBOB and CEC benchmarks. We use bounds 10^2 and 10^{-8} with log-scaling between them to be consistent with the common COCO setup [7]. This figure shows no difference between these methods in the early search stage (since no restarts have been triggered yet). Overall, repelling results in a slight performance drop for the BBOB benchmark, while performance increases for the CEC benchmark. Looking at individual functions for the BBOB benchmark (see Fig. ??), we can observe that while for some functions, the RR-CMA-ES shows lower performance than its standard counterpart, this is not always the case. An important aspect to note is that on functions with

[1] This ECDF is based on the empirical attainment function, equivalent to infinite targets for the standard ECDF [12].

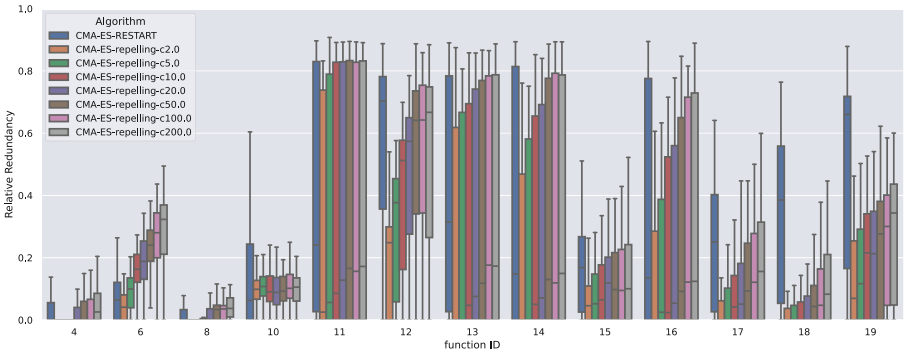


Fig. 6. Distribution of redundant function evaluation over the CEC'13 functions, only function with any redundant restarts for CMA-ES-RESTART are shown. The CMA-ES with the 'RESTART' strategy is compared to the RR-CMA-ES, with different coverage factors c.

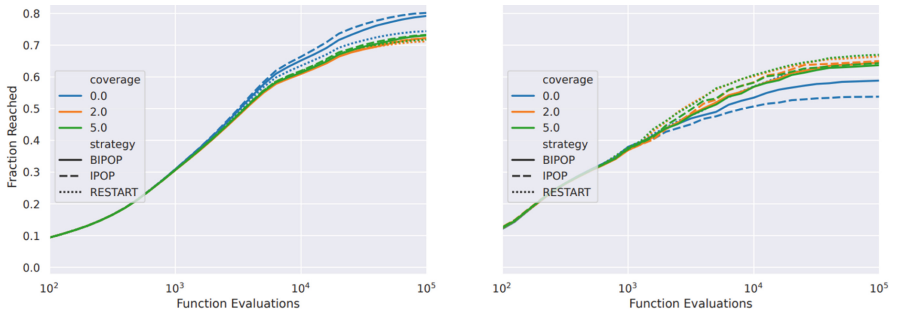
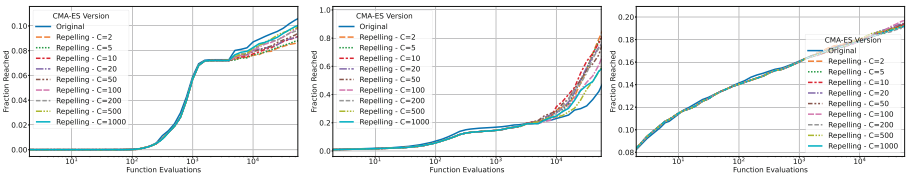


Fig. 7. Aggregated ECDF of the EAF for both the BBOB and CEC benchmarks, shown for the 'standard' ($c = 0$) CMA-ES using a RESTART, IPOP, and BIPOP strategy, and the RR-CMA-ES with coverage factors c of 2.0 and 5.0.



(a) F4 (b) F22 (c) F23

Fig. 8. ECDF of the different versions of RR-CMA-ES compared to the original CMA-ES, for selected BBOB functions (in dimensionality 10) where potential improvement was observed (see Fig. 2).

a clear global structure, such as F4, the repelling regions might adversely affect convergence, as shown in Fig. 8a. For other functions, such as F22, where the distribution of the local optima is more uniform, the repelling strategy has a clear benefit in performance. Thus, a balance between preventing redundant restarts and avoiding steering the search away from regions near the global optimum has to be found.

6 Conclusions and Future Work

In this paper, we reflected that global optimization of multimodal problems is an area where current optimization algorithms can still be improved in light of an evident redundancy of restart mechanisms. When independent restarts converge to the same local optima, these essentially waste function evaluations could be prevented by more effectively guiding the search in subsequent restarts.

Our proposed repelling-based restart method for CMA-ES shows a slight performance improvement over the default restart mechanisms for specific cases and reduces redundant convergence for the BBOB functions. For the multimodal CEC functions specifically, the method shows a large reduction of redundant restarts compared to a standard restart strategy.

Further research into the precise ways of avoiding these redundancies is still required. In particular, the rejection criteria of the tabu regions could be modified to incorporate information about the fitness values found in this region to handle global structure better.

The tabu regions' shapes could also be formed based on the converged covariance matrix at the point or slightly before the restart is triggered to potentially better capture the precise basin shape. Furthermore, extended practices such as *regularization* could also be exercised to remedy numerical issues and to obtain a more accurate structure (see, e.g., [25]), relying on the theoretical relation of the ESs' covariance matrix and the landscape local Hessian [26].

In addition to more effectively preventing a restart from converging to a known basin of attraction, one could also utilize the information from prior restarts to more intelligently select the new location and initial parameterization of new restarts. In comparison to, e.g., MLSL, starting points could be selected based on which regions of the domain have already been sampled, which would also cause the repelling mechanism to trigger less often during the initial phase of the restarted run.

In the context of global optimization, the explicit criteria used to trigger a restart mechanism also play a large role in the algorithm's performance. Depending on the optimization goal, restarting more frequently and exploiting the best-attained basins of attraction using a local search method with a small fraction of the total budget might be worthwhile.

References

1. Ahrari, A., Deb, K., Preuss, M.: Multimodal optimization by covariance matrix self-adaptation evolution strategy with repelling subpopulations. Evol. Comput. **25**(3), 439–471 (2017). https://doi.org/10.1162/evco_a_00182
2. Antonov, K., Botari, T., Tukker, T., Bäck, T., van Stein, N., Kononova, A.V.: New solutions to Cooke triplet problem via analysis of attraction basins. In: Kress, B.C., Czarske, J.W. (eds.) Digital Optical Technologies 2023, vol. 12624, p. 126240T. International Society for Optics and Photonics, SPIE (2023). https://doi.org/10.1117/12.2675836
3. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1769–1776 (2005). https://doi.org/10.1109/CEC.2005.1554902
4. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, New York (2004)
5. Glover, F., Laguna, M.: Tabu search. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, pp. 2093–2229. Springer, Boston (1998). https://doi.org/10.1007/978-1-4613-0303-9_33
6. Hansen, N.: Benchmarking a bi-population CMA-ES on the BBOB-2009 function testbed. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, pp. 2389–2396 (2009)
7. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. Optim. Methods Softw. **36**(1), 114–144 (2021)
8. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions. Technical report, RR-6829, INRIA (2009). https://hal.inria.fr/inria-00362633/document
9. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 282–291. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_29
10. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
11. Li, X., Engelbrecht, A., Epitropakis, M.G.: Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization. RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Technical report (2013)
12. López-Ibáñez, M., Vermetten, D., Dreo, J., Doerr, C.: Using the empirical attainment function for analyzing single-objective black-box optimization algorithms. arXiv preprint arXiv:2404.02031 (2024)
13. Loshchilov, I., Schoenauer, M., Sebag, M.: Alternative restart strategies for CMA-ES. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012. LNCS, vol. 7491, pp. 296–305. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32937-1_30
14. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: framework and applications. In: Handbook of Metaheuristics, pp. 129–168 (2019)
15. Maree, S.C., Alderliesten, T., Thierens, D., Bosman, P.A.N.: Real-valued evolutionary multi-modal optimization driven by hill-valley clustering. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 857–864. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3205455.3205477

16. Minasny, B., McBratney, A.B.: A conditioned latin hypercube method for sampling in the presence of ancillary information. Comput. Geosci. **32**(9), 1378–1388 (2006). https://doi.org/10.1016/j.cageo.2005.12.009
17. Nishida, K., Akimoto, Y.: Benchmarking the PSA-CMA-ES on the BBOB noiseless testbed. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1529–1536 (2018)
18. de Nobel, J., Shir, O., Vermetten, D., Kononova, A.V., Bäck, T.: Reproducibility files and additional figures (2024). https://doi.org/10.5281/zenodo.10997200
19. de Nobel, J., Vermetten, D., Wang, H., Doerr, C., Bäck, T.: Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1375–1384 (2021)
20. de Nobel, J., Ye, F., Vermetten, D., Wang, H., Doerr, C., Bäck, T.: Iohexperimenter: benchmarking platform for iterative optimization heuristics. Evol. Comput. 1–6 (2024)
21. Pál, L.: Benchmarking a hybrid multi level single linkagealgorithm on the BBOB noiseless testbed. In: Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 1145–1152 (2013)
22. Preuss, M.: Multimodal Optimization by Means of Evolutionary Algorithms. Natural Computing Series, Springer (2015). https://doi.org/10.1007/978-3-319-07407-8
23. Rinnooy Kan, A., Timmer, G.T.: Stochastic global optimization methods part i: clustering methods. Math. Program. **39**, 27–56 (1987)
24. Shir, O.M.: Niching in evolutionary algorithms. In: Rozenberg, G., Baeck, T., Kok, J.N. (eds.) Handbook of Natural Computing, pp. 1035–1069. Springer, Heidelberg (2012). http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-92909-3. https://doi.org/10.1007/978-3-540-92910-9_32
25. Shir, O.M., Roslund, J., Whitley, D., Rabitz, H.: Efficient retrieval of landscape hessian: forced optimal covariance adaptive learning. Phys. Rev. E **89**, 063306 (2014). https://doi.org/10.1103/PhysRevE.89.063306
26. Shir, O.M., Yehudayoff, A.: On the covariance-hessian relation in evolution strategies. Theoret. Comput. Sci. **801**, 157–174 (2020). https://doi.org/10.1016/j.tcs.2019.09.002
27. Törn, A., Zilinskas, A.: Global Optimization. Lecture Notes in Computer Science, vol. 350. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-50871-6
28. Ursem, R.: Multinational evolutionary algorithms. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), vol. 3, pp. 1633–1640 (1999). https://doi.org/10.1109/CEC.1999.785470

LB+IC-CMA-ES: Two Simple Modifications of CMA-ES to Handle Mixed-Integer Problems

Tristan Marty[1,2], Nikolaus Hansen[2(✉)], Anne Auger[2], Yann Semet[1], and Sébastien Héron[1]

[1] Thales Research and Technology, Palaiseau, France
[2] Inria and CMAP, Ecole Polytechnique, IP Paris, Palaiseau, France
nikolaus.hansen@inria.fr

Abstract. We present LB+IC-CMA-ES, a variant of CMA-ES that handles mixed-integer problems. The algorithm uses two simple mechanisms to handle integer variables: (i) a lower bound (LB) on the variance of integer variables and (ii) integer centering (IC) of variables to their domain middle depending on their value. After presenting the algorithm, we evaluate the different variants ensuing from these modifications on the BBOB mixed-integer testbed and compare the performance with the recently introduced CMA-ES with margin.

Keywords: mixed-integer optimization · CMA-ES · Evolution Strategies

1 Introduction

Mixed-integer optimization problems appear commonly in applications. Applying a continuous optimizer, for example the CMA-ES algorithm [5], *as is* to mixed-integer problems often leads to premature convergence of the integer coordinates on the wrong plateau. For this reason, different variants of CMA-ES have been proposed in recent years to handle mixed-integer problems, including CMA-ES with margin for the $(\mu/\mu_w, \lambda)$-CMA-ES [3] and also for the $(1+1)$-CMA-ES [14]. In order to prevent stagnation, the CMA-ES with margin sets lower bounds on the marginal probabilities to mutate integer variables. Other Evolution Strategies to handle mixed-integer problems include DX-NES-ICI [10] which has been designed specifically for mixed-integer problems where continuous variables are more decisive than integer ones, or MIESs [11] that uses a different mutation operator for non-continuous variables.

In this context, we build on a version of CMA-ES that bounds the variance of integer coordinates from below [12]. First, we propose a theoretically motivated setting of the lower bound. Second, we introduce the centering of mutated integer variable values to the middle of the integer plateau. This latter mechanism usually introduces a bias on the sample population average for which we consequently correct.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 284–299, 2024.
https://doi.org/10.1007/978-3-031-70068-2_18

The paper is organized as follows. Section 2 presents CMA-ES and the two modifications introduced to improve the performance on mixed-integer problems. Section 3 illustrates the impact of both mechanisms by showing single runs on an ill-conditioned ellipsoid function with some integer variables. Section 4 assesses the performance of the different variants on the mixed-integer BBOB testbed as well as on functions with varying fraction of integer variables and Sect. 5 concludes the paper.

2 Two Simple Modifications of CMA-ES to Handle Mixed-Integer Problems

We describe the CMA-ES algorithm for numerical minimization of a function $f : \mathbb{R}^N \to \mathbb{R}$ and our two modifications to handle mixed-integer problems.

When given a mixed-integer function $f(\boldsymbol{x})$ where some coordinates of $\boldsymbol{x}$ belong to $\mathbb{Z}$ and other coordinates belong to $\mathbb{R}$, we denote by $\mathcal{S}_\mathbb{Z}$ the index set of integer coordinates and $\mathcal{S}_\mathbb{R}$ the index set of real coordinates. The cardinality of $\mathcal{S}_\mathbb{Z}$ equals $N_{\text{int}} = |\mathcal{S}_\mathbb{Z}|$ and we have $N = |\mathcal{S}_\mathbb{Z}| + |\mathcal{S}_\mathbb{R}|$. Then, $x_i \in \mathbb{Z}$ for $i \in \mathcal{S}_\mathbb{Z}$ and $x_i \in \mathbb{R}$ for $i \in \mathcal{S}_\mathbb{R}$. We define the function

$$\text{int}[.] : \mathbb{R} \to \mathbb{Z}, x \mapsto \lfloor x + 1/2 \rfloor \tag{1}$$

yielding the "integer value" of $x \in \mathbb{R}$. When we apply a continuous algorithm to a mixed-integer function, we apply int[.] to each integer coordinate before we evaluate f.

2.1 CMA-ES

The $(\mu/\mu_w, \lambda)$-CMA-ES samples λ solutions $\boldsymbol{x}$ at each iteration t distributed according to a multivariate normal distribution $\boldsymbol{x}_i^{(t)} \sim \mathcal{N}(\boldsymbol{m}^{(t)}, (\sigma^{(t)})^2 \mathbf{D}^{(t)} \mathbf{C}^{(t)} \mathbf{D}^{(t)})$ where $\boldsymbol{m}^{(t)} \in \mathbb{R}^N$ represents the incumbent mean solution. The covariance matrix of the sampling distribution is decomposed into three parts: (i) the overall step-size $\sigma^{(t)}$; (ii) the covariance matrix $\mathbf{C}^{(t)}$ containing information about the sensitivity in some principal axes; (iii) the diagonal matrix $\mathbf{D}^{(t)}$ introduced in [1] to scale the distribution in the given coordinate system.

For sampling λ solutions, we transform samples from an isotropic normal distribution, $\boldsymbol{z}_i^{(t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, using $\mathbf{D}^{(t)}$ and $\mathbf{C}^{(t)}$,

$$\boldsymbol{y}_i^{(t)} = \sqrt{\mathbf{C}^{(t)}} \boldsymbol{z}_i^{(t)} \ ; \ \boldsymbol{x}_i^{(t)} = \boldsymbol{m}^{(t)} + \sigma^{(t)} \mathbf{D}^{(t)} \boldsymbol{y}_i^{(t)} \quad \text{for } i = 1, \ldots, \lambda \ , \tag{2}$$

where $\sqrt{\mathbf{C}^{(t)}}$ is symmetric and positive definite. The candidate solutions are ranked according to their objective function f. The index $i{:}\lambda$, as defined by the next equation, refers to the i^{th} solution when ordered by their f-value,

$$f(\boldsymbol{x}_{1:\lambda}^{(t)}) \leq f(\boldsymbol{x}_{2:\lambda}^{(t)}) \leq \ldots \leq f(\boldsymbol{x}_{\lambda:\lambda}^{(t)}) \ . \tag{3}$$

The new mean is a weighted recombination of the sorted solutions

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + \sigma^{(t)}\mathbf{D}^{(t)} \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \tag{4}$$

with $\mu = \lfloor \lambda/2 \rfloor$ and $w_i \propto \log((\lambda+1)/2) - \log(i)$ such that $\sum_{i=1}^{\mu} w_i = 1$.

Step-Size. The parameter $\sigma^{(t)}$ is updated using the length of a so-called evolution path, $\boldsymbol{p}_\sigma^{(t+1)} \in \mathbb{R}^N$. The step size is increased if $\|\boldsymbol{p}_\sigma^{(t+1)}\|$ is larger than the expected length of a standard normally distributed random vector and decreased if it is smaller, specifically

$$\boldsymbol{p}_\sigma^{(t+1)} = (1-c_\sigma)\boldsymbol{p}_\sigma^{(t)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{(t)^{-\frac{1}{2}}} \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \tag{5}$$

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|}{\mathbb{E}\left(\|\mathcal{N}(\mathbf{0},\mathbf{I})\|\right)}\right)\right) \quad , \tag{6}$$

where c_σ determines the decay of $\boldsymbol{p}$ and, by default, $c_\sigma = \frac{\mu_{\text{eff}}+2}{N+\mu_{\text{eff}}+5}$ with $\mu_{\text{eff}} = 1/\sum_i w_i^2$ and the step-size damping $d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\text{eff}}-1}{N+1}} - 1\right) + c_\sigma$.

Covariance Matrix. Finally, the covariance matrix $\mathbf{C}^{(t)}$ is updated with both rank-one and rank-μ updates, the latter of which makes use of negative weights. Like for the step size update, the covariance matrix update relies on an evolution path updated as

$$\boldsymbol{p}_c^{(t+1)} = (1-c_c)\boldsymbol{p}_c^{(t)} + h_\sigma \sqrt{c_c(2-c_c)\mu_{\text{eff}}} \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \quad , \tag{7}$$

where h_σ prevents a rapid increase of variances in $\mathbf{C}^{(t)}$ when the step-size is already increasing, for example in the first iterations, and reads

$$h_\sigma = \begin{cases} 1 & \text{if } \frac{\|\boldsymbol{p}_\sigma^{(t+1)}\|^2}{1-(1-c_\sigma)^{2t}} < 2N\left(1 + \frac{2}{N+1}\right) \\ 0 & \text{otherwise} \end{cases} \quad . \tag{8}$$

Given the weights $w_1 \geq w_2 \geq \ldots \geq w_\mu > 0 \geq w_{\mu+1} \geq w_\lambda$ [6, Appendix A], we introduce

$$w_i^\circ = \begin{cases} w_i & \text{if } w_i \geq 0 \\ w_i \frac{N}{\|\sqrt{\mathbf{C}^{(t)^{-1}}}\boldsymbol{y}_{i:\lambda}\|^2} & \text{otherwise} \end{cases} \quad . \tag{9}$$

To guaranty positive definiteness, we further scale down the negative weights when necessary [1, Sec 3.2]. The covariance matrix update then reads:

$$\mathbf{C}^{(t+1)} = \left(1 + c_1(1-h_\sigma)c_c(2-c_c) - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i\right)\mathbf{C}^{(t)} + c_1 \boldsymbol{p}_c^{(t+1)}(\boldsymbol{p}_c^{(t+1)})^\top + c_\mu \sum_{i=1}^{\lambda} w_i^\circ \boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^\top \quad . \tag{10}$$

Learning rates and the cumulation parameter are set to $c_1 = \frac{2}{(N+1.3)^2+\mu_{\text{eff}}}$, $c_\mu = \min\left(1-c_1,\ 2\frac{\mu_{\text{eff}}+1/\mu_{\text{eff}}-1.75}{(N+2)^2+\mu_{\text{eff}}}\right)$ and $c_c = \frac{4+\mu_{\text{eff}}/N}{N+4+2\mu_{\text{eff}}/N}$.

The diagonal matrix $\mathbf{D}^{(t)}$ is not updated here but will be utilized to set a lower bound for integer variables.

2.2 Lower Bounding the Standard Deviation on Integer Coordinates

Applying CMA-ES as presented in the previous section to mixed-integer functions can lead to premature convergence of integer variables at a nonoptimal value when their standard deviation gets much smaller than 1/2. As the step-size converges to zero, also the probability to sample a solution with a different integer value converges to zero and the variable is trapped. This behavior can be observed in Figure 3a which shows the coordinates of the mean $\boldsymbol{m}^{(t)}$ during the optimization of a 30-dimensional mixed-integer ellipsoid function. We observe that four integer variables converge to a nonoptimal value.

To prevent the premature convergence of integer coordinates, we impose a lower bound, σ_{LB}, on their standard deviation by updating the diagonal matrix $\mathbf{D}^{(t)}$. Let $\cdot_{j,j}$ denote the j^{th} diagonal element of a matrix. We define

$$\sigma_{\text{std(j)}}^{(t)} = \sigma^{(t)}\mathbf{D}_{j,j}^{(t)}\sqrt{\mathbf{C}_{j,j}^{(t)}} \quad \text{for } j = 1,\ldots,N \ , \tag{11}$$

and update the diagonal matrix right before Eq. (2) as

$$\mathbf{D}_{j,j}^{(t)} \leftarrow \frac{\max\left(\sigma_{\text{LB}}, \sigma_{\text{std(j)}}^{(t)}\right)}{\sigma^{(t)}\sqrt{\mathbf{C}_{j,j}^{(t)}}} \quad \text{for } j \in \mathcal{S}_{\mathbb{Z}} \tag{12}$$

which changes $\mathbf{D}_{j,j}^{(t)}$ only when $\sigma_{\text{std(j)}}^{(t)} < \sigma_{\text{LB}}$ and ensures that $\sigma_{\text{std(j)}}^{(t)} \geq \sigma_{\text{LB}}$.

Setting the Lower-Bound. The choice of σ_{LB} indirectly controls the minimal mutation rate of integer variables. In order to allow the mutation rate to be small enough to not disrupt the optimization of continuous variables, we propose the lower bound

$$\sigma_{\text{LB}} = \min\left(\frac{\mu_{\text{eff}}}{N}, 0.2\right). \tag{13}$$

We estimate the effect of σ_{LB} on the *mutation rate* (the proportion of candidate solutions which is on a different integer plateau than the mean) based on the following simulation. We optimize the one-dimensional function $x \mapsto \lfloor x + 1/2 \rfloor^2$ with a fixed step-size starting at $x = 0$ and record at each iteration the fraction of solutions which are outside of $[-0.5, 0.5]$. The average is taken between iteration $3t_0$ and $\max(100\,t_0, 10^4/2)$ or at most 10^7, where t_0 is the first iteration with one mutation, i.e., with one candidate solution sampled outside $[-0.5, 0.5]$. The mutation rate, p_1, is shown in Fig. 1 for different step-sizes and different population sizes.

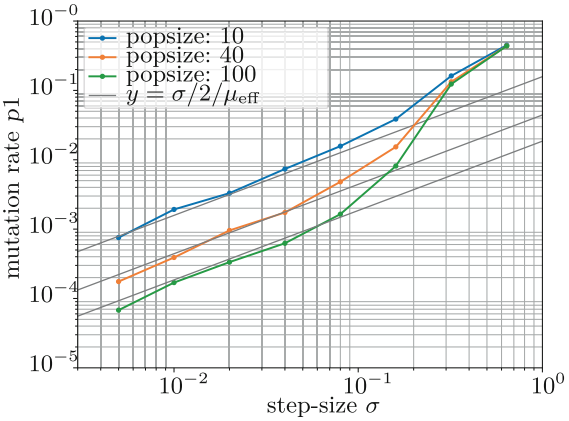


Fig. 1. Fraction of mutated individuals per iteration versus the step-size of the sample distribution for different population sizes.

The mutation rate can be well approximated by $p_1 \approx \sigma/\mu_{\mathrm{eff}}/2$ when $\sigma < 0.1$. This dependency on μ_{eff} is somewhat surprising, because the speed of the random walk is proportional to $1/\sqrt{\mu_{\mathrm{eff}}}$.

We now derive (13). Let p_N denote the probability that at least one integer coordinate of a solution is mutated (integer-different from the mean) and λ_{EC} denote the *effective continuous population size*, that is, the average number of solutions that are unaffected from integer mutations (that have in all integer coordinates the same integer value as the mean). Given that $\Delta f \to 0$ in the continuous subspace when $\sigma \to 0$, f-changes induced by an integer mutation dominate the f-values when σ is small. Therefore, to ensure progress in the continuous subspace, we want several solutions to be unaffected from integer mutations.

We have the following equations

$$p_1 \approx \frac{\sigma}{2\mu_{\text{eff}}} \qquad \text{if } \sigma < 0.1 \tag{14}$$

$$p_N \approx p_1 \times N_{\text{int}} \qquad \text{if } p_N \leq 1/2 \text{ or } \sigma \lesssim \mu_{\text{eff}}/N_{\text{int}} \tag{15}$$

$$\lambda_{\text{EC}} \approx \lambda(1 - p_N) \tag{16}$$

$$\lambda_{\text{EC}} \gtrsim \gamma \frac{N_{\text{int}}}{N} + \frac{N - N_{\text{int}}}{N}\lambda \qquad \text{if } N_{\text{int}} < N \tag{17}$$

Equation (14) follows from Fig. 1. Equation (15) approximates, when p_1 is small, $1 - p_N = (1 - p_1)^{N_{\text{int}}}$ assuming independence. Equation (16) approximates the unaffected population size, λ_{EC}. Equation (17) gives a heuristic lower bound for λ_{EC}, where γ represents the smallest reasonable population size for the continuous subspace (when N_{int} is close to N).

Combining the first two and the last two equations yields, respectively,

$$p_N \approx \frac{\sigma N_{\text{int}}}{2\mu_{\text{eff}}} \quad \text{and} \quad p_N \lesssim \frac{N_{\text{int}}}{N}\left(1 - \frac{\gamma}{\lambda}\right) \ , \tag{18}$$

and combining these gives an upper bound for the step-size

$$\sigma \lesssim \frac{2\mu_{\text{eff}}}{N}\left(1 - \frac{\gamma}{\lambda}\right) \quad \text{and} \quad \sigma \lesssim \frac{4\mu_{\text{eff}}}{3N} \quad \text{for } \gamma \lesssim \lambda/3 \ . \tag{19}$$

Equation (19) reveals the largest step-size that presumably allows for an effective search in the continuous subspace and is hence an upper bound for σ_{LB}.

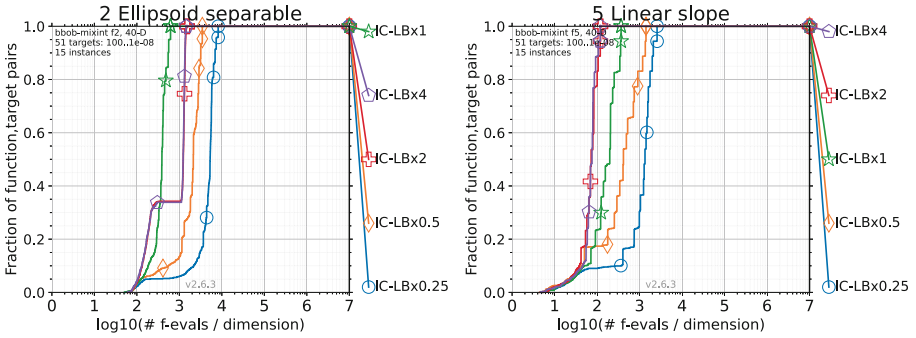


Fig. 2. Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of f-evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8..2]}$ in dimension 40. (Legend: algorithm IC-LBx0.25, IC-LBx0.5, IC-LBx1, IC-LBx2, IC-LBx4 are CMA-ES with integer centering and a lower bound defined $\sigma_{\text{LB}} = \min\left(\alpha\frac{\mu_{\text{eff}}}{N}, 0.2\right)$ where α takes respectively the values 0.25, 0.5, 1, 2, and 4)

To evaluate the validity of (13), we run LB-CMA-ES on the `bbob-mixint` testbed [13] with lower bound $\sigma_{\text{LB}} = \min\left(\alpha\frac{\mu_{\text{eff}}}{N}, 0.2\right)$ for $\alpha \in 0.25, 0.5, 1, 2, 4$.

Figure 2 shows some exemplary results on two functions in dimension 40. On the separable ellipsoid, more difficult target values are reached about twice as fast with $\alpha = 1$ as with the other tested values ($p \approx 10^{-4.5}$). We observe the same behavior on the 40D sphere function ($p \approx 10^{-3.5}$, not shown). On the linear slope in dimension 40, larger values for α are generally faster: with $\alpha = 1$, we reach the more difficult targets two times slower than with $\alpha \in 2, 4$ ($p \approx 10^{-3.5}$) but at least 3 times faster than with $\alpha \in 0.5, 0.25$ ($p \approx 0.02$). A lower bound which is two times larger than Eq. 13 but still clipped at 0.2 has mostly similar performance but is more at risk to disrupt convergence.

2.3 Integer Centering

When $\sigma^{(t)}_{\text{std(j)}}$ is small, a successful integer mutation will often have little effect because its value is likely to be close to the original integer value and because the impact of a single solution on the mean update diminishes with increasing population size. To mitigate these effects, we set successfully mutated values to the center of the integer interval. Furthermore, we aim to maintain the unweighted "unselected" average taken over the μ best sampled solutions.

Let $(.)_j$ denote the j-th coordinate of a vector, let $I_\mu = \{1{:}\lambda, \ldots, \mu{:}\lambda\}$ denote the index set of the μ best solutions from (3), and for $i \in I_\mu$ let $\boldsymbol{x}_i^{(t^-)} = \boldsymbol{x}_i^{(t)}$ from (2). Before applying Eq. (4), we set

$$(\boldsymbol{x}_i^{(t)})_j \leftarrow \text{int}\left[(\boldsymbol{x}_i^{(t^-)})_j\right] \quad \text{if} \quad \text{int}\left[(\boldsymbol{x}_i^{(t^-)})_j\right] \neq \text{int}\left[(\boldsymbol{m}^{(t)})_j\right] \ , \tag{20}$$

for $i \in I_\mu$ and $j = 1, \ldots, N$. That is, we apply integer centering to $(\boldsymbol{x}_i^{(t)})_j$ if it has a different integer value than the mean.

The centering of the candidate solutions introduces a bias,

$$b_j^{(t)} = \sum_{i \in I_\mu} \left((\boldsymbol{x}_i^{(t)})_j - (\boldsymbol{x}_i^{(t^-)})_j\right) \quad \text{for} j = 1, \ldots, N \ . \tag{21}$$

To compensate for this bias, we set for $i \in I_\mu$ and $j = 1, \ldots, N$

$$(\boldsymbol{x}_i^{(t)})_j \leftarrow (\boldsymbol{x}_i^{(t^-)})_j + \mathbb{1}(i,j)\alpha_j \Delta_{ij} \tag{22}$$

such that $\alpha_j \leq 1$ minimizes $b_j^{(t)}$, where $\Delta_{ij} = \text{int}\left[(\boldsymbol{x}_i^{(t^-)})_j\right] - (\boldsymbol{x}_i^{(t^-)})_j$ and

$$\mathbb{1}(i,j) = \begin{cases} 1 & \text{if int}\left[(\boldsymbol{x}_i^{(t)})_j\right] = \text{int}\left[(\boldsymbol{m}^{(t)})_j\right] \text{ and } b_j^{(t)}\Delta_{ij} < 0 \\ 0 & \text{otherwise} \end{cases} \ . \tag{23}$$

That is, when $(\boldsymbol{x}_i^{(t)})_j$ has not been centered yet, we move it by α_j towards its centered version, given this move reduces the bias (21).

These modifications do not change the f-value of the solutions, that is, $f(\boldsymbol{x}_i^{(t)}) = f(\boldsymbol{x}_i^{(t^-)})$, hence they do not change (3). The $\boldsymbol{y}_i^{(t)}$ from (2) are reassigned according to (20) and (22).

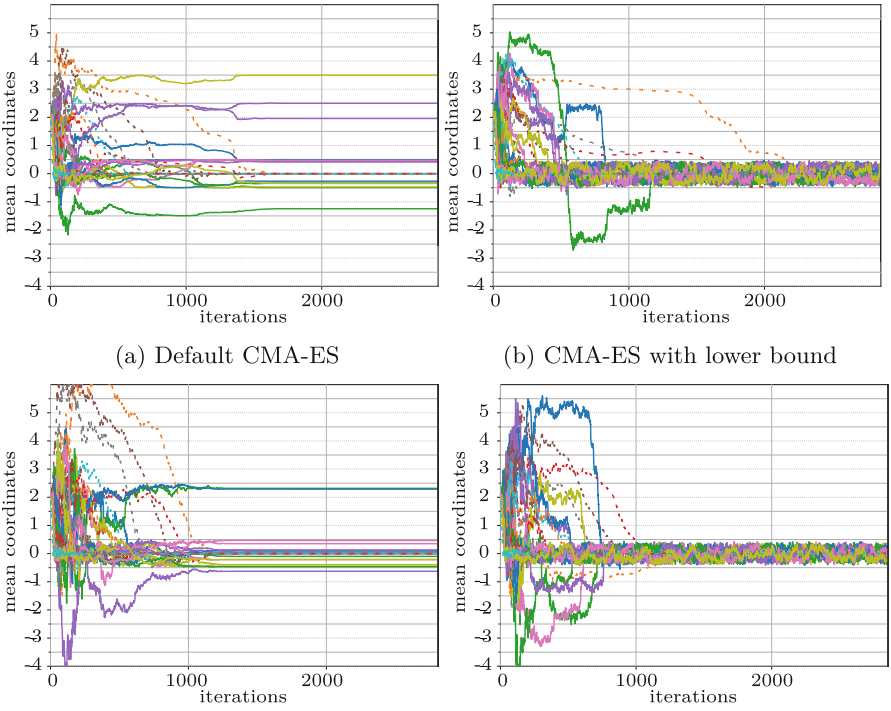
All our code is based on the cma Python package, commit 334abfc.

3 Single Runs of the Different Variants

To illustrate the effect of the above modifications, some optimization runs are shown in Fig. 3. All graphs result from the optimization of the 30-dimensional mixed-integer ellipsoid function (see caption). We desire the continuous variables to linearly converge to the optimum at zero and the integer variables to end up in $[-0.5, 0.5]$.

Without integer handling (Fig. 3a), integer variables get stuck after about 1400 iterations where four variables assume a nonoptimal value. Because the step size converges to zero, they will not change their final position anymore.

When the standard deviations are lower bounded according to Sect. 2.2, integer variables cannot get anymore stuck and end up in the optimal interval $[-0.5, 0.5]$ where their behavior resembles a bounded random walk (Fig. 3b). With integer centering only, variables appear to move more swiftly before iter-



(a) Default CMA-ES (b) CMA-ES with lower bound

(c) CMA-ES with integer centering (d) Lower bound and integer centering

Fig. 3. Evolution of the mean, $\boldsymbol{m}^{(t)}$, of CMA-ES with default population size $\lambda = 14$ on the 30-dimensional ellipsoid function $f_{\text{elli}}(x) = \sum_{i=1}^{N} 10^{6\frac{i-1}{N-1}} x_i^2$ where variables with even/odd index are continuous/integer and plotted in dashed/solid, respectively. Initial mean and step-size are $\boldsymbol{m}^{(0)} = (2, ..., 2)$ and $\sigma^{(0)} = 0.1$, respectively. Integer variable values in $[-0.5, 0.5]$ are optimal.

ation 1000, however, two integer variables still get stuck at a nonoptimal value (Fig. 3c). The effect of combining lower bound and integer centering appears to be additive (Fig. 3d): integer variables appear to move more swiftly, and all integer variables reach the optimal interval within about 800 iterations.

4 Performance Assessment

We assess the impact of lower bounding the standard deviations of integer variables and of integer centering on the 24 functions of the `bbob-mixint` testbed [13]. The functions are benchmarked in dimensions 5, 10, 20, 40 and 80 where 20% of these variables are continuous and the rest are integer, 20% with arity 2, 4, 8 and 16, respectively. The k-ary variables take values between 0 and $k-1$. For each function, 51 target f-values are defined as $f_{\text{opt}} + \Delta f$ where f_{opt} is the minimal function value and $\Delta f = 10^k$ with $k = 2, 1.8, \ldots, -8$. The number of function evaluations to reach each target is then recorded.

As the objective function remains constant for integer values smaller than 0 and larger than k, we bound their search domain to $[-1/2, k + 1/2]$. Continuous variables are not bounded. Boundaries are handled by adding a penalty term to the objective function [9] with the cma.BoundPenalty class. The initial coordinate-wise standard deviation is set to one fifth of the bounded range and to $10/5 = 2$ for continuous variables.

We benchmark all four combinations with and without LB and/or IC, referred to as base-CMA, LB-CMA, IC-CMA and LB+IC-CMA. All tested variants use IPOP-CMA-ES [2]: if a termination condition is met before the allowed budget is exhausted, we restart CMA-ES with doubled population size. The termination conditions `tolflatfitness` and `tolfunhist` are changed from their default values to 5 and 0, respectively.

We compare our variants with two previously benchmarked algorithms: CMA-ES-pycma, with some basic integer handling [13] and CMA-ES with margin (CMA-ESwM) which, similar to LB+IC-CMA, modifies coordinate-wise standard deviations and the mean vector to ensure a minimal marginal probability of exploring new integer variables [3,4].

Empirical runtime distributions [7] are shown by function groups in Fig. 4 and for each function in Figs. 5 and 6.[1] Mentioned p-values are computed within the COCO platform [8] and represent the result of a ranksum test.

First we investigate the effect of the lower bound. The LB-CMA algorithm takes at least 10 times less functions evaluations than base-CMA for solving the target $\Delta f = 10^{-7}$ on the separable ellipsoid $f2$ in dimension 10, 20, 40 and 80 ($p \approx 10^{-4.5}$). Also, LB-CMA reaches the target $\Delta f = 10^{-7}$ at least 5 times faster than base-CMA on the 80D functions `f1`, `f5`, `f12`, `f13` ($p \approx 0.02$ for `f12`, $p \approx 10^{-2.5}$ for `f13`, $p \approx 10^{-3.5}$ for `f5` and $p \approx 10^{-4.5}$ for `f1`). On the other hand base-CMA does not reach any target faster than LB-CMA. Introducing a lower

[1] All data and a code example can be found at https://trmarty.github.io/LB-IC-CMA-ES-data/.

bound on the standard deviations mostly improves the behavior of the algorithm on unimodal functions.

Next, we examine the effect of integer centering. The IC-CMA reaches the target $\Delta f = 10^{-7}$ at least twice as fast as base-CMA on the 80D functions `f2`, `f14`, `f18` ($p \approx 0.02$ for `f18`, $p \approx 10^{-2.5}$ for `f14` and $p \approx 10^{-4.5}$ for `f2`). However, base-CMA solves function `f15` eight times and function `f24` four times out of 15 whereas IC-CMA never succeeds.

With both lower bound *and* integer centering (LB+IC-CMA), the number of functions evaluations to reach the target 10^{-7} is reduced by a factor 4 compared to LB-CMA on the 80D functions `f14`, `f17` and `f18` ($p \approx 10^{-2.5}$ for `f18`, $p \approx 10^{-3.5}$ for `f17` and $p \approx 10^{-4.5}$ for `f14`). However, LB-CMA optimizes `f24` up to the final target while LB+IC-CMA does not even reach the easy target of $\Delta f = 10$. Overall, LB+IC-CMA has lower runtimes than the base-CMA in dimension 80 on functions `f1`, `f2`, `f12`, `f13`, `f14`, `f17` and `f19` at the expense of not solving the multimodal functions `f15` and `f24`.

Finally, the LB+IC-CMA-ES algorithm is compared with CMA-ES with margin. The target $\Delta f = 10^{-7}$ is reached faster with LB+IC-CMA than with CMA-ESwM on functions `f8` and `f24` in dimension 5, `f1`, `f2`, `f8`, `f12`, `f14`, `f17` in dimension 10, `f1`, `f2`, `f7`, `f12`, `f13`, `f15` and `f17` in dimension 20, `f1`, `f2`, `f7`, `f12` and `f13` in dimension 40 and on functions `f2`, `f12`, `f17` and `f18` in dimension 80 ($p < 0.05$ for all functions). On the other hand, CMA-ESwM is 2, 4 and 10 times faster on the linear slope `f5` in dimension 20, 40 and 80, respectively ($10^{-5} < p < 10^{-3}$) which has its optimum on the boundary of the domain. Removing the boundary handling speeds up LB+IC-CMA by a factor of three to hit the last target $\Delta f = 10^{-8}$ ($p \approx 10^{-5}$).

In all other cases, both algorithms have similar performance. In summary, LB+IC-CMA performs better than CMA-ESwM on 24 of 120 functions and worse on three.

Table 1. Final population size in dimension 40 and 80 for functions with at least one successful trial and one restart.

function id	min	med	max
f1, 80D	17	34	34
f2, 40D	15	15	30
f2, 80D	34	34	34
f5, 40D	15	15	30
f5, 80D	34	34	68
f7, 40D	120	240	960

function id	min	med	max
f12, 40D	15	15	30
f12, 80D	17	17	34
f13, 40D	15	15	30
f13, 80D	17	17	34
f14, 40D	15	15	60
f14, 80D	17	17	34

function id	min	med	max
f15, 40D	60	120	240
f17, 40D	15	30	30
f17, 80D	34	68	136
f18, 40D	15	30	120
f18, 80D	68	136	272
f21, 40D	15	30	480
f21, 80D	34	34	136

Table 1 shows final population sizes for functions with at least one successful run and *at least one restart* in dimensions 40 and 80 (11 functions). The default

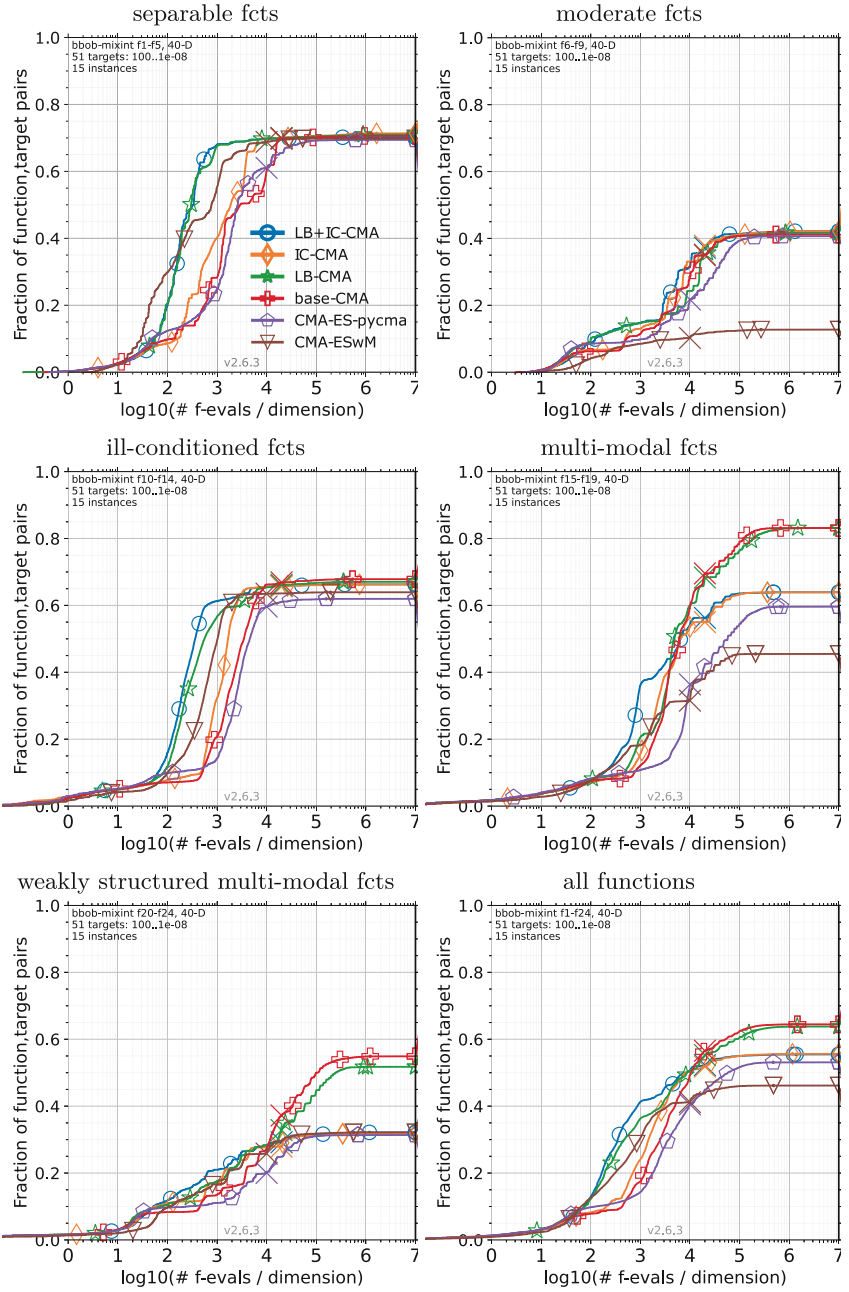


Fig. 4. Bootstrapped empirical cumulative distributions of the number of f-evaluations divided by dimension for 51 targets with target precision in $10^{[-8..2]}$ for all functions and subgroups in 40-D. The performance of CMA-ESwM on the moderate function group is negatively affected by missing data on `f9`.

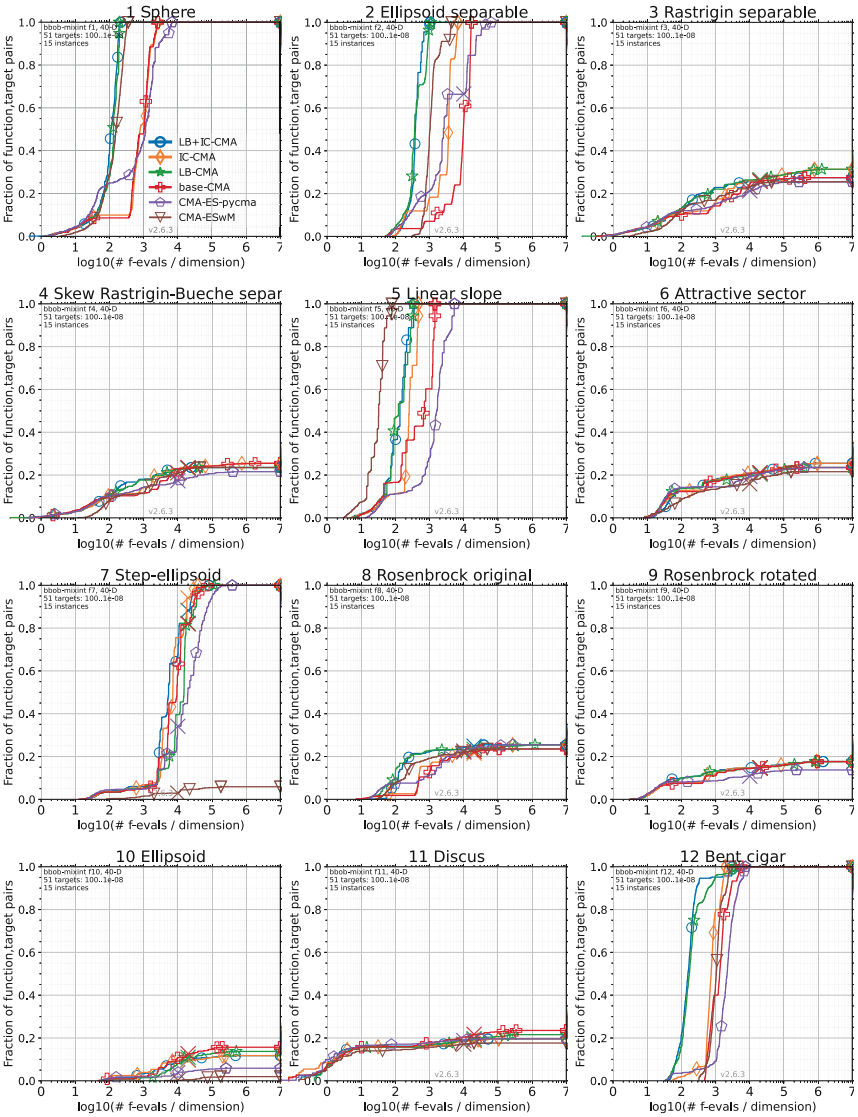


Fig. 5. Empirical cumulative distributions of simulated (bootstrapped) runtimes, measured in number of f-evaluations divided by dimension, for the 51 targets $10^{[-8..2]}$ in dimension 40.

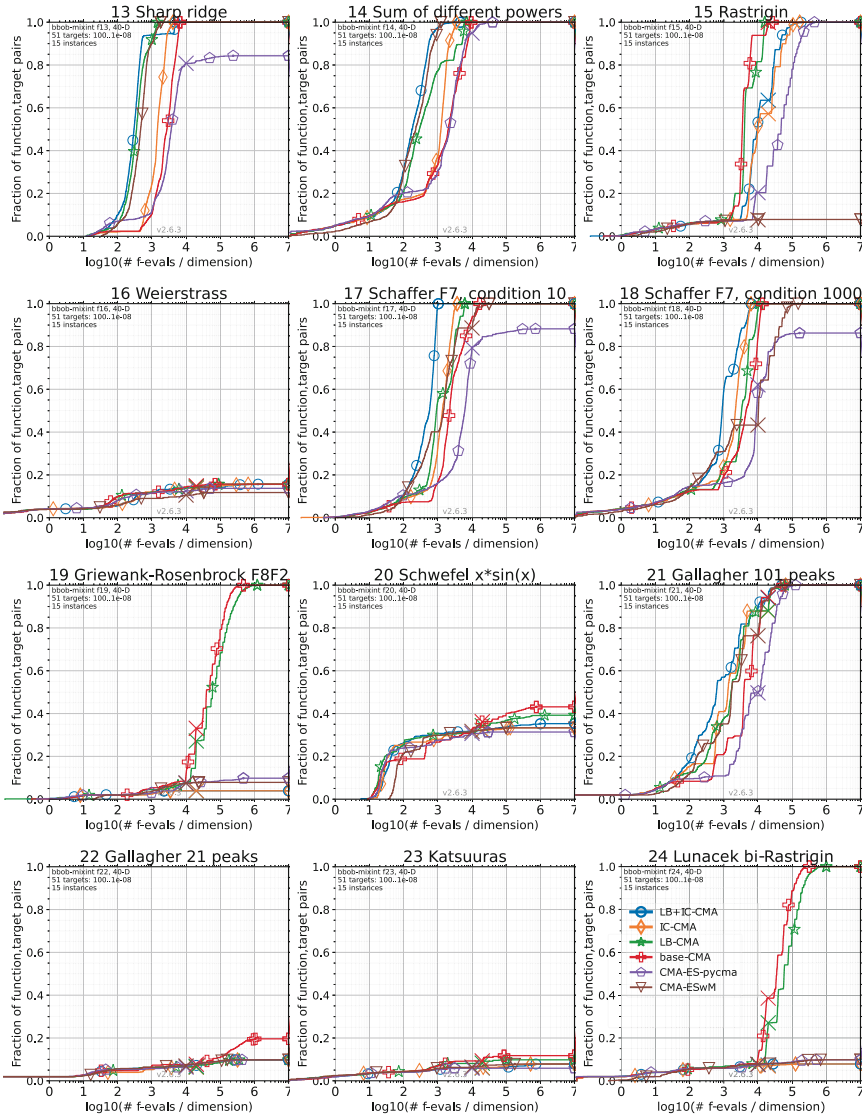
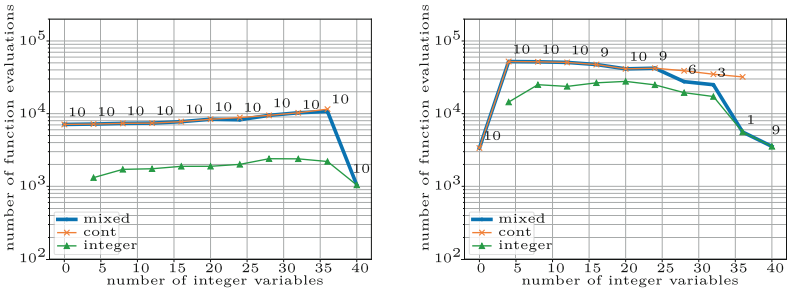


Fig. 6. Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of f-evaluations divided by dimension, for the 51 targets $10^{[-8..2]}$ in dimension 40.

population size is 15 and 17 in dimension 40 and 80, respectively, and with IPOP each restart doubles the population. Hence, the final population size also implies the number of conducted restarts. The largest population sizes can be observed on the step ellipsoid function `f7` (up to 960). In dimension 80, LB+IC-CMA often restarts even on the sphere function `f1`. Apparently, the `tolfun` termination condition triggers too early in larger dimension.



(a) Sphere (b) Ellipsoid

Fig. 7. Average number of function evaluations to solve the 40D sphere and ellipsoid function up to $f = 10^{-10}$ with LB+IC-CMA, plotted versus the number of integer variables. Results are for the mean $\boldsymbol{m}^{(t)}$ in the continuous (orange) and the integer (green) subspace and for the best solution on the overall mixed-integer problem (blue). The number of successful runs out of 10 is given too. (Color figure online)

Finally, Fig. 7 shows average runtimes versus the percentage of integer variables on the sphere function and the ellipsoid function (see Fig. 3) with a conditioning of 10^6 in dimension 40. A run is successful if the best solution reaches the target value of 10^{-10}. On the sphere integer function (Fig. 7a), problems with at least one continuous variable are solved within 7000 to 11 000 evaluations. Adding only few integer variables to this continuous problem does not increase the evaluation time by much ($\times 1.1$ between 0 and 4 integer variables), however adding a few continuous variables to an integer-only problem has a much larger impact ($\times 10$ between 40 and 36 integer variables). The ellipsoid function (Fig. 7b) is harder to solve. With increasing number of integer variables, the success rate falls below 100% and with 36 integer variables, the success rate drops to 10%. The integer-only and continuous-only problems takes around 3500 evaluations to find the optimum which is approximately 10 times faster than all other mixed problems that are solved with a probability larger than 50%. With 20% continuous variables, the BBOB benchmark represents comparatively difficult problems in terms of fraction of integer variables.

5 Summary and Conclusion

We propose two comparatively simple modifications of CMA-ES to improve the performance of the algorithm on problems with (some) integer variables. First,

we derived a lower bound for the standard deviation of integer coordinates which depends on the dimension of the problem and the population size. The bound aims to be low enough to preserve a large enough effective population size in the continuous subspace. Second, we move successfully mutated integer variables to the center of their integer interval to adopt good mutations of integer variables more swiftly.

The lower bound greatly reduces the runtime of CMA-ES on functions `f1`, `f2`, `f5`, `f12` and `f13` of the `bbob-mixint` suite [13]. Additional integer centering leads to faster convergence on functions `f14`, `f17` and `f18`, however also to a performance degradation on the multimodal functions `f19` and `f24` and, in higher dimension, on `f15`. The cause of this degradation remains unclear. We suspect that restarts on `f1` and `f7` lead to unnecessary performance impediments too. The LB+IC-CMA-ES compares overall favorably to the CMA-ES with margin [4], especially on functions `f2` and `f12` in dimension 10–80 and on eight other functions depending on the dimension.

Many BBOB mixed-integer multimodal functions remain unsolved by the current algorithms. The proposed variant, LB+IC-CMA-ES, however improves over previous versions and is a comparatively simple new baseline to compare with.

References

1. Akimoto, Y., Hansen, N.: Diagonal acceleration for covariance matrix adaptation evolution strategies. Evol. Comput. **28**(3), 405–435 (2020)
2. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1769–1776. IEEE (2005)
3. Hamano, R., Saito, S., Nomura, M., Shirakawa, S.: Benchmarking CMA-ES with margin on the bbob-mixint testbed. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1708–1716 (2022)
4. Hamano, R., Saito, S., Nomura, M., Shirakawa, S.: CMA-ES with margin: lower-bounding marginal probability for mixed-integer black-box optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 639–647 (2022)
5. Hansen, N.: A CMA-ES for mixed-integer nonlinear optimization. INRIA research report, RR-7751 (2011)
6. Hansen, N.: The CMA evolution strategy: a tutorial. arXiv preprint arXiv:1604.00772 (2016)
7. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. Optim. Methods Softw. **36**(1), 114–144 (2021)
8. Hansen, N., et al.: COmparing Continuous Optimizers: numbbo/COCO on Github (2019). https://doi.org/10.5281/zenodo.2594848
9. Hansen, N., Niederberger, A.S., Guzzella, L., Koumoutsakos, P.: A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. IEEE Trans. Evol. Comput. **13**(1), 180–197 (2008)

10. Ikeda, K., Ono, I.: Natural evolution strategy for mixed-integer black-box optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 831–838 (2023)
11. Li, R., et al.: Mixed integer evolution strategies for parameter optimization. Evol. Comput. **21**(1), 29–64 (2013)
12. Marty, T., Semet, Y., Auger, A., Héron, S., Hansen, N.: Benchmarking CMA-ES with basic integer handling on a mixed-integer test problem suite. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, pp. 1628–1635 (2023)
13. Tušar, T., Brockhoff, D., Hansen, N.: Mixed-integer benchmark problems for single- and bi-objective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 718–726 (2019)
14. Watanabe, Y., Uchida, K., Hamano, R., Saito, S., Nomura, M., Shirakawa, S.: (1+1)-CMA-ES with margin for discrete and mixed-integer problems. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 882–890 (2023)

Bayesian- and Surrogate-Assisted Optimization

Performance Comparison of Surrogate-Assisted Evolutionary Algorithms on Computational Fluid Dynamics Problems

Jakub Kůdela(✉) and Ladislav Dobrovský

Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Brno, Czech Republic
jakub.kudela@vutbr.cz, dobrovsky@fme.vutbr.cz

Abstract. Surrogate-assisted evolutionary algorithms (SAEAs) are recently among the most widely studied methods for their capability to solve expensive real-world optimization problems. However, the development of new methods and benchmarking with other techniques still relies almost exclusively on artificially created problems. In this paper, we use two real-world computational fluid dynamics problems to compare the performance of eleven state-of-the-art single-objective SAEAs. We analyze the performance by investigating the quality and robustness of the obtained solutions and the convergence properties of the selected methods. Our findings suggest that the more recently published methods, as well as the techniques that utilize differential evolution as one of their optimization mechanisms, perform significantly better than the other considered methods.

Keywords: Expensive optimization · evolutionary algorithm · surrogate model · computational fluid dynamics · benchmarking

1 Introduction

The field of evolutionary computation (EC) has produced a multitude of pivotal evolutionary (or metaheuristic) algorithms (EAs) [30], such as genetic algorithm (GA) [45], evolutionary strategy (ES) [4], differential evolution (DE) [55], or particle swarm optimization (PSO) [27], that were used to solve a wide range of different optimization problems [7,18]. However, these standard EAs usually require a large number of objective/constraint function evaluations to find satisfactory solutions, which severely limits their utility for solving expensive real-world problems [22]. In applications such as aerodynamic design, computational fluid dynamics (CFD), or finite element method analysis, a single function evaluation requires running expensive computations that may take several minutes to several hours [34]. The need for such high-fidelity simulations imposes a practical limit on the number of designs that may be considered during optimization.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 303–321, 2024.
https://doi.org/10.1007/978-3-031-70068-2_19

The interest in computationally expensive optimization has increased substantially in the past few years [34]. Perhaps the first widely known example of such problems was presented in [24] where the two applications were in integrated circuit design and in the automotive industry. Since then, there have been numerous applications of optimization for such expensive problems, especially in the field of CFD. A multi-objective shape optimization of aerofoils (accounting for low-drag and high-lift) was investigated in [46]. A CFD-based shape optimization problem that aimed to minimize the mass of beams under structural constraints was presented in [38]. In a heat exchanger design, a multi-objective formulation (that maximized heat flux and minimized pressure drop) was proposed in [16]. More recent applications are concerned with CFD-based geometry optimization problems, such as the minimization of pressure differences in pipes [12] and ducts [11], and in ship hydrodynamic optimization [54], or in multiobjective ship hull design [8].

One of the approaches for solving these computationally expensive problems is to construct surrogate models (also called meta-models) to assist the EAs. Such surrogate models are generally constructed using historical data to approximate and predict the landscape of the objective function, with a negligible computational cost. These types of algorithms are called data-driven EAs [22] or surrogate-assisted EAs (SAEAs) [42]. Depending on whether real function evaluations can be performed during the optimization process, SAEAs can be further divided into two categories: offline SAEAs and online SAEAs [22]. The offline SAEAs prioritize building the most suitable surrogate model based on available historical data to predict the position of the optimal solution [59]. In contrast to the offline SAEAs, the online SAEAs work by effectively sampling candidate solutions that will be evaluated on the real expensive function, and then update the corresponding surrogate models and populations in the process of optimization [22].

The majority of SAEAs are constructed on the basis of standard EAs and are organically combined with surrogate models for predicting expensive real function evaluations [65]. Various EAs, such as GA [68], DE [42], or PSO [63] have been successfully used in SAEAs and, in recent years, there has been a multitude of SAEAs proposed in the literature. However, to develop these methods and compare them with other SAEAs, most authors rely almost exclusively on artificially created problems.

As EC methods in general (as SAEAs in particular) are difficult to analyze analytically, the majority of the reasoning about their utility is done by benchmarking [19]. Throughout the years, many different benchmark sets and functions were introduced in journal articles [17,33], but the most widely-used ones were developed for competitions (and special sessions) on black-box optimization at two high-profile conferences, the Genetic and Evolutionary Computation Conference (GECCO), and the IEEE Congress on Evolutionary Computation (CEC). However, the use of these benchmark sets is not without critique, with some authors voicing their concern about the artificial nature of these problems [47] and advocating for benchmarking EAs on real-world problems instead [58].

Notable exceptions are the recent works on comparing low-budget methods on the OpenAI Gym [49] and on the EXPObench library [5].

Although there are numerous applications of SAEAs, the authors of these applications generally develop custom codes for the problems. As many of these applications are proprietary, the authors are often reluctant to release the source codes [14]. This means that despite having numerous published successful applications of SAEAs, it is generally difficult to use the simulators for benchmarking different methods. The SAEA community is now actively looking for real-world computationally expensive problems that can serve as benchmarks. The vast majority of published SAEAs algorithms that are not tied to a specific application are benchmarked on a handful of problems (such as the Ellipsoid, Rosenbrock, Ackley, and Griewank functions) from the CEC competitions [10,40]. However, these are only pseudo-expensive problems, i.e. problems that are inexpensive in nature and use artificial delays or restrictions on the number of function evaluations to mimic the expensive problems [14].

This lack in the availability of real-world-based benchmark sets of expensive problems for SAEAs spurred the creation of the CFD suite presented first at the PPSN XV conference [14]. The problems in this suite focus on designing different mechanisms, using CFD to evaluate the performances of the candidate geometries in a fluid environment. Other places to find computationally expensive problems that are well-suited for benchmarking are the GECCO Industrial Challenge Competitions, with one of them focusing on CFD problems [51], while others aimed at a hospital planning application [3,50]. These industrial problems were used to benchmark parallel SAEA methods in [52], but a thorough computational comparison of the state-of-the-art SAEAs on such problems was still missing.

In this paper, we aim to fill this research gap by benchmarking state-of-the-art single-objective SAEAs on the CFD problems from the suite [14]. As the representative methods, we selected eleven SAEAs that were recently published either in high-profile journals or conferences and were developed on similar test problems (in terms of dimensions and available function evaluations). We analyze the performance of the selected SAEAs by investigating the quality and robustness of the best-found solutions and their convergence properties. In the analysis, we follow the recently published guidelines for comparing EAs [37]. For the sake of replicability of the findings [2,28], the codes for all the methods used in the computational comparison, and the code used to run the CFD experiments are made publicly available in the Zenodo repository [31].

The rest of this paper is organized as follows. Section 2 introduces the framework of surrogate-assisted optimization. In Sect. 3 we describe the two CFD problems used for the computational comparisons. Section 4 contains a brief discussion of the selected state-of-the-art SAEAs. Section 5 reports on the results of the computational comparisons. Section 6 concludes the paper and discusses implications and future work.

Algorithm 1. Prototypical structure of a surrogated-assisted optimization method [26].

Require: budget B, surrogate model M, acquisition function A
  Initialize $x^{(1)}$ randomly and an empty set H
  for $m = 1 : B$ **do**
    $y^{(m)} \leftarrow f(x^{(m)})$
    $H \leftarrow H \cup \{(x^{(m)}, y^{(m)})\}$
    $M \leftarrow$ fit surrogate model using H
    $x^{(m+1)} \leftarrow \text{argmax}_x A(M, x)$
  end for
  return best found $(x^*, y^*) \in H$

2 Surrogate-Assisted Optimization

In this paper, we consider the class of so-called black-box optimization problems, where the objective function $f : \mathbb{R}^D \to \mathbb{R}$ has no closed-form expression and only obtainable information about f come from observing its output when evaluating $f(x)$ given some input (decision variable) $x \in \mathbb{R}^D$. The prototypical box-constrained black-box optimization problem has the following form

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathbb{R}^D \\ & l_i \leq x_i \leq u_i, \quad i = 1, \ldots, D, \end{aligned} \tag{1}$$

where l_i, u_i are the lower and upper bounds on the decision variable x_i, respectively. The evaluation of $y = f(x)$ is assumed to be computationally expensive (i.e., requiring running time-consuming CFD simulations). This implies that we are mostly interested in finding good (and not necessarily optimal) solutions to (1) in a reasonable amount of time. We approach this by setting a limited budget B for the number of available calls of f.

One of the possible approaches for this class of problems is to utilize a so-called surrogate model, which can be thought of as an auxiliary function M that we use to approximate the objective function f. The surrogate model should be cheaper to evaluate than the original black-box function f and is usually constructed using the evaluation history (m already evaluated points) $H = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$. Kriging (also known as Gaussian Process) models and Radial Basis Functions (RBFs) are among the most widely used approaches for generating surrogate models [34]. Other techniques that are also used with SAEAs are polynomial response surface methods [29] and support vector machines [43], with various different techniques also considered [6,44,67]. It was found that the Kriging models outperform other surrogates in solving low-dimensional optimization problems, while RBFs are the more efficient surrogates for solving high-dimensional optimization problems [15].

The main purpose of the surrogate model is to predict the next promising points for evaluation, which is typically also guided by an acquisition function

$A(M, x)$. This function predicts how promising a new point x is, balancing the trade-off of exploration (the search in regions where the surrogate model displays high levels of uncertainty) and exploitation (the search near already evaluated points that resulted in low objective values). The prototypical structure of a surrogate-assisted optimization method is shown in Algorithm 1. Most SAEAs initialize the population by evaluating a fixed number of points selected by the Latin Hypercube Sampling method [20].

3 Selected Problems

As the performance of a given engineering design in a fluid environment usually cannot be evaluated analytically, we often resort to various numerical CFD approximations. These CFD computations require the solution of a set of Partial Differential Equations (PDEs) describing the physics of the flow of the fluid, which is typically approached using finite volume (and similar) methods. There are numerous software products available that can perform these calculations. One of the most used open-source options is the C++ code OpenFOAM [60]. Both of the selected problems, described in greater detail in [14] and [52], use OpenFOAM for the CFD simulation. The problem geometries generated from the optimization variables are transformed into sterolithography (STL) files and imported into this OpenFOAM framework [13]. Similar setup was recently used in [5], but with a completely different set of algorithms used for the comparison.

3.1 PitzDaily

The separation of flows, reattachment, and recirculation are all phenomena commonly observed in various engineering applications. Such features are usually undesirable [14]. The first CFD problem is based on an experimental study by Pitz and Daily [48] (hence the name), which features a so-called "backward-facing step", that is often used as a simplified prototype for studying the flow phenomena mentioned above. In such a geometry (which can be found in Fig. 3 under the "PitzDaily boundary" label), the flow separates at the edge of the step and creates a recirculation zone. Afterward, the flow reattaches at a distance beyond this step. This case study has been used as a benchmark case for the accuracy testing of different CFD methodologies.

One of the undesirable characteristics of a flow is head losses. We can describe this quantity (the mechanical energy loss factor ζ) as the energy that is transformed to a form which can no longer be used in the operation of an energy-producing, conducting, or consuming system (for instance because of the frictional losses, or dissipation due to turbulence) [14]. The energy loss factor ζ of a given design can be defined as the total pressure difference between the inflow and outflow of the mechanism (taken relative to the kinetic energy at the inflow)

$$\zeta = \frac{2}{\rho U_{\text{in}}^2} \left[\frac{1}{A_{\text{in}}} \int_{\text{in}} P_{t,\text{in}} (u \cdot n) dA_{\text{in}} - \frac{1}{A_{\text{out}}} \int_{\text{out}} P_{t,\text{out}} (u \cdot n) dA_{\text{out}} \right]$$

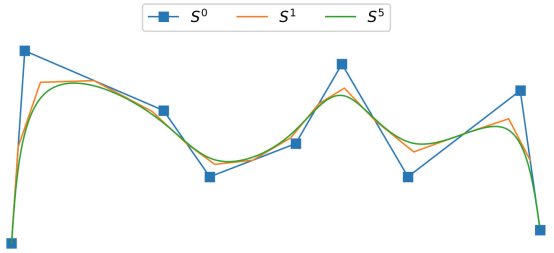


Fig. 1. Example of the Catmull-Clark subdivision curve. S^0 are the original control points, S^1 is the approximation of $\mathcal{C}$ after one iteration, S^5 is the visually smooth curve after five iterations.

where ρ is the density of the fluid, U_{in} is the inflow velocity, A is the cross-sectional area, P_t is the total pressure, $u \cdot n$ describes the velocity component normal to the boundary, and subscripts "in" and "out" indicate the inflow and outflow boundaries. The primary objective of this problem is to minimize the energy loss ζ by changing the geometry of the design.

The standard procedure to create such geometry would be to use some Computer-Aided Design (CAD) software. The problem is that automatically altering designs using CAD is very challenging. Therefore, we generally resort to different parametric representations of the parts of the original CAD geometry. We can then generate new geometries by changing the parameters of such representation.

For the PitzDaily problem, we use the Catmull-Clark subdivision curves [9] to alter the boundary wall. In this method, the curve $\mathcal{C}$ is parameterized by a sequence of n vertices (or control points) $S^0 = \langle v_1, \ldots, v_n \rangle$. In each iteration, a mid-point between adjacent vertices is inserted, and the positions of the vertices are adjusted, generating a larger sequence. From the practical perspective, only a few iterations of these subdivisions are necessary for a visually smooth curve (see Fig. 1) exportable in the STL format. In this paper, the iteration limit was set to five. We chose the positions of five control points of the Catmull-Clark curve in 2D as the decision vector (see Fig. 3), resulting in an optimization problem with 10 variables.

3.2 Electrostatic Precipitator

The Electrostatic Precipitator (ESP) is a real-world problem from industrial optimization that was first proposed in [52]. The ESP is one of the crucial components in gas cleaning systems used in combustion power plants (or similar industries) to remove solid particles from gas streams (pollution reduction). Figure 2 shows a schema of such a system.

To control the flow of the gas flow through the different separation zones (where the removal of particles from the exhaust gases occurs) a gas distribution system (GDS) is needed. If there were no GDS used (or the system was not well

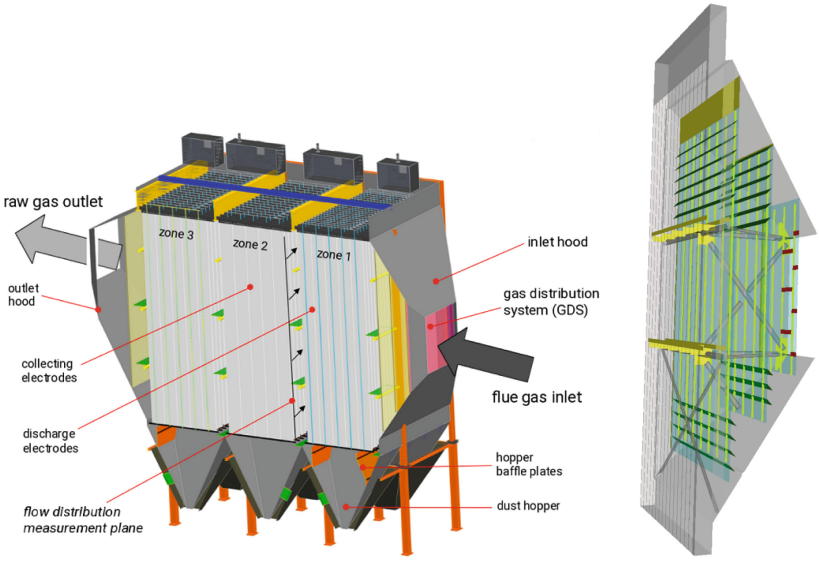


Fig. 2. ESP with 3 separation zones (left) and GDS mounted in the inlet hood of an ESP (right) [51].

configured), the fast inlet gas stream would run through the separation zones, resulting in low separation efficiencies. For the efficient operation of the EPS, the GDS must have a good configuration. In our case, the GDS has 49 slots, which can be configured with baffles (metal plates mounted at an angle to the flow of the gas), blocking plates (block the gas stream), or perforated plates (slow down and partially block the gas stream).

In total, there are 8 different options available for each of the 49 configurable slots (decision variables), resulting in a solution space of roughly 10^{44} possibilities. Even though this problem is discrete in nature, it has been shown in [26] that continuous SAEAs are well-suited to solve it. The ESP problem was also the focus of the GECCO 2020 Industrial Challenge [51].

4 Selected Methods and Experimental Setup

For the selection and comparison of representative SAEAs, we followed the guidelines published in [37]. The selected methods were all recently developed algorithms that were published either in high-profile journals (such as IEEE Transactions on Cybernetics, IEEE Transactions on Evolutionary Computation, Information Sciences, etc.) or conferences (such as GECCO). All of the methods were also trained on similar dimensions (between 10 and 100, which covers the range of our two CFD problems), using a similar computational budget.

Table 1 shows a brief overview of the SAEAs selected for the computational comparison in chronological order. We can find that the most used surrogate

Table 1. SAEAs selected for computational comparison. The "Real?" column shows if the method was evaluated on real-world problems.

Method name	Year	Ref.	Surrogate model	Optimization method	Real?
SHPSO	2018	[63]	RBF	PSO	N
SSL-A-PSO	2018	[57]	RBF	PSO	N
GORS-SSLPSO	2019	[62]	RBF	PSO	N
MS-MTO	2020	[41]	RBF	MTO	N
TL-SSLPSO	2020	[61]	RBF	PSO	Y
BiS-SAHA	2021	[53]	RBF	PSO, DE	Y
IKAEA	2021	[64]	Kriging	DE	N
SAMSO	2021	[39]	RBF	TLBO, PSO	N
TS-DDEO	2021	[66]	RBF	PSO, DE	N
ESA	2022	[65]	RBF	DE	N
LSADE	2022	[35]	RBF, Lipschitz	DE, SQP	N

model was RBF, and the most utilized EAs were PSO and DE. Only two of the selected methods were tested on real-world problems, with the rest relying purely on artificially created ones. However, these two real-world problems were not computationally expensive (both used analytical expressions in the formulations).

Another common feature of all of the selected methods was that they had their respective codes publicly available, making it possible for verification of the presented results as well as for conducting additional computational comparisons. The implementation of the selected SAEAs, as well as the important information about their parametrization, and the implementation of the CFD problems, can be found in a public Zenodo repository [31].

4.1 Experimental Setup

The selected SAEAs were implemented in MATLAB R2022b and the experiments were run on a workstation with 3.7 GHz AMD Ryzen 9 5900X 12-Core processor and 64 GB RAM. As the selected SAEAs are stochastic methods, each of them was run 24 times (efficiently using the 12-core machine) on the two CFD problems, in order to get statistically representative results and provide a solid basis for algorithmic comparison. To give context to the results, we also implemented a simple random search (RS) routine. The function evaluation budget was set to $B = 1{,}000$ in both cases. For both problems, the time needed for the evaluation of the objective function (i.e., running the CFD simulations) was approximately 30 s.

5 Results and Discussion

5.1 PitzDaily Results

The results of the computations (statistics of the 24 independent runs) on the PitzDaily problem (with $D = 10$) are summarized in Table 2. The Catmull-Clark curve of the best solution (i.e., the best design shape) found by any method (in this case, `ESA`) and a randomly generated solution as well as the flowfield (CFD) simulations for these designs are shown in Fig. 3 (these were generated using the ParaView software). On the best-found design, we can observe a much smoother transition of u from one boundary to the other, resulting in a lower value of the energy loss factor ζ.

Based on the statistics shown in Table 2, the best methods for the PitzDaily problem were `ESA`, `LSADE`, `IKAEA`, and `TS-DDEO`. A different insight can be found when looking at the convergence plots of average best-found solutions of the methods shown in Fig. 4. Here, we find that `LSADE`, `ESA`, and `TS-DDEO` were able to find good solutions faster than the other considered SAEAs. These three are among the most recently proposed methods (in Fig. 4, the convergence of methods published after 2021 is shown as a dashed line). As Fig. 4 shows only the average performance of the selected SAEAs, we also boxplots of the solutions found in the 24 runs with two different budgets $B = 500$ and $B = 1{,}000$, that are shown in Fig. 5. Here, we can find that `LSADE` had a very good performance on both budgets that was somewhat worsened by a single run for which the method was not able to find a good solution. Also, the `TS-DDEO` method showed only a marginal improvement upon `SHPSO` (its base method), which was among the best-performing methods for this problem, despite being the oldest one.

Table 2. Statistics of the 24 runs of the selected SAEAs on the PitzDaily model. The best three methods in each category are highlighted in bold.

Method	Min	Mean	Median	Max	Std	Rank
`RS`	8.59E-02	9.26E-02	9.24E-02	9.63E-02	3.19E-03	11.71
`SHPSO`	8.03E-02	8.31E-02	8.34E-02	8.61E-02	1.62E-03	5.50
`SSL-A-PSO`	8.06E-02	8.46E-02	8.31E-02	9.34E-02	3.21E-03	5.96
`GORS-SSLPSO`	8.08E-02	8.62E-02	8.52E-02	9.42E-02	3.26E-03	8.63
`MS-MTO`	8.31E-02	8.62E-02	8.52E-02	9.47E-02	3.04E-03	9.08
`TL-SSLPSO`	8.28E-02	8.45E-02	8.32E-02	9.35E-02	2.90E-03	6.04
`BiS-SAHA`	8.31E-02	8.52E-02	8.42E-02	9.14E-02	2.27E-03	8.50
`IKAEA`	**7.95E-02**	**8.28E-02**	**8.30E-02**	**8.88E-02**	2.31E-03	**4.17**
`SAMSO`	8.00E-02	8.38E-02	8.33E-02	8.99E-02	1.79E-03	6.17
`TS-DDEO`	7.98E-02	8.29E-02	**8.30E-02**	**8.79E-02**	1.67E-03	4.50
`ESA`	**7.94E-02**	**8.23E-02**	**8.27E-02**	**8.56E-02**	1.98E-03	**4.08**
`LSADE`	**7.95E-02**	**8.24E-02**	8.31E-02	9.13E-02	2.48E-03	**3.67**

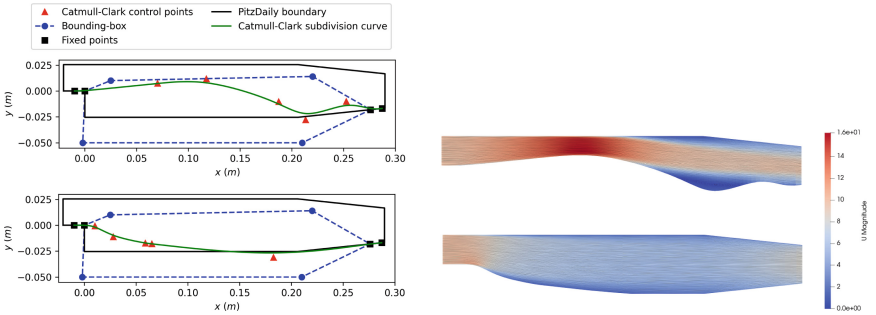


Fig. 3. Random design (top) and best-found design (bottom) for the PitzDaily problem (left), flowfield (CFD) simulations of these designs (right).

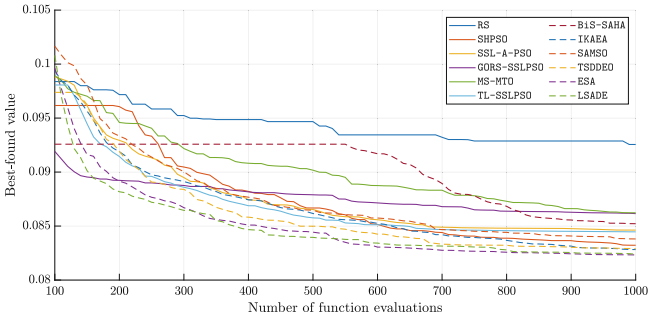


Fig. 4. Convergence of the average best-found value of the selected SAEAs on the PitzDaily problem.

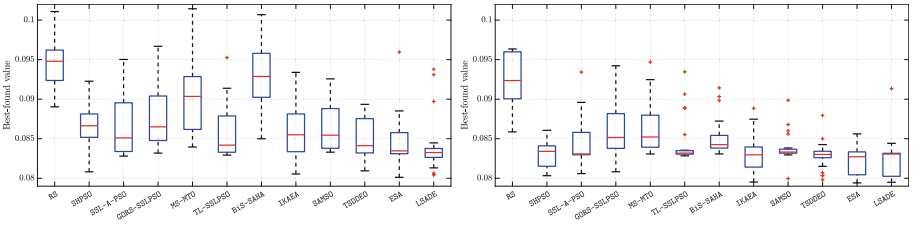


Fig. 5. Boxplots of the solutions of the SAEAs on the PitzDaily problem with $B = 500$ (left) and $B = 1000$ (right).

To perform a solid statistical comparison of the selected algorithms on the PitzDaily problem, we followed the guidelines published in [37]. First, we used the Friedman test to find if significant differences are present among all the algorithms on the $B = 500$ and $B = 1{,}000$ budgets (the Friedman ranks of the methods for $B = 1{,}000$ are shown in Table 2). The p-values for this test were 1.16E-18 (and 1.50E-12 when omitting `RS`) for $B = 500$ and 2.37E-21 (and 6.46E-13 when omitting `RS`) for $B = 1{,}000$. As all these p-values values

were much lower than the recommended confidence level $\alpha = 0.05$, we can state that there are statistically significant differences between the selected SAEAs. Furthermore, we utilized the Wilcoxon signed-rank test to find if there exist statistically significant differences between the best algorithm (in this case, the LSADE algorithm with the lowest Friedman rank) and the other SAEAs. Once the pairwise p-values were obtained, we applied the Holm-Bonferroni [21] correction method which counteracts the effect of multiple comparisons by controlling the family-wise error rate [1]. The results of the analysis are presented in Table 3. For the $B = 500$ budget, there were 6 methods that tied with LSADE as the best-performing ones. For the $B = 1{,}000$ budget, the list of the methods that tied with LSADE changed - TL-SSLPSO and SAMSO dropped, while SHPSO entered. Forming a union of these two lists, we find that four methods (SSL-A-PSO, IKAEA, TS-DDEO, and ESA) were found to be as good as LSADE on the PitzDaily problem. Four of these five methods were proposed after 2021, and also four of the five utilize DE in some capacity (in both cases, the exception was SSL-A-PSO).

Table 3. Statistical analysis of the comparison of the selected SAEAs on the PitzDaily problem.

$B = 500$			$B = 1{,}000$		
LSADE vs.	p	p^*	LSADE vs.	p	p^*
SHPSO	2.46E-03	1.72E-02	SHPSO	8.65E-02	3.46E-01✗
SSL-A-PSO	1.91E-02	9.57E-02✗	SSL-A-PSO	1.10E-02	5.50E-02✗
GORS-SSLPSO	8.05E-05	6.44E-04	GORS-SSLPSO	8.05E-05	7.25E-04
MS-MTO	2.67E-05	2.67E-04	MS-MTO	2.35E-05	2.35E-04
TL-SSLPSO	2.78E-02	1.11E-01✗	TL-SSLPSO	6.64E-03	3.99E-02
BiS-SAHA	5.61E-05	5.05E-04	BiS-SAHA	3.96E-04	3.17E-03
IKAEA	4.55E-02	1.37E-01✗	IKAEA	3.46E-01	8.71E-01✗
SAMSO	1.10E-02	6.60E-02✗	SAMSO	2.46E-03	1.72E-02
TS-DDEO	1.23E-01	2.32E-01✗	TS-DDEO	2.90E-01	8.71E-01✗
ESA	1.16E-01	2.32E-01✗	ESA	8.19E-01	8.19E-01✗

p: p-value computed by the Wilcoxon text
p^*: p-value corrected with the Holm-Bonferroni procedure
✗: statistical differences do not exist with significance level $\alpha = 0.05$

5.2 ESP Results

The results of the computations on the ESP problem (with $D = 49$) are summarized in Table 4. Unfortunately, we do not have an expressive way of visualizing the best-found solution to the ESP problem (this time, found by IKAEA), in the same way that we had for the PitzDaily problem in the form of Fig. 3. Based on the statistics shown in Table 4, the best methods for the ESP problem were TL-SSLPSO, IKAEA, SAMSO, and BiS-SAHA.

The convergence plot, shown in Fig. 6, displays some interesting patterns. There were four methods (BiS-SAHA, GORS-SSLPSO, TL-SSLPSO, and LSADE) that got relatively good solutions in a very short amount of function calls (approximately at $B = 200$). However, apart from TL-SSLPSOwhich was the best method overall, the three other methods stagnated and got overtaken by IKAEA and SAMSO roughly at $B = 500$. Another thing to note is that the SSL-A-PSO method, which performed relatively well on the PitzDaily problem, was the worst of the selected SAEAs on the ESP by a large margin. Similarly to the PitzDaily case, we can find that the more recent methods dominated the older ones (apart from TL-SSLPSO).

The boxplots of the solutions found in the 24 runs with the two budgets ($B = 500$ and $B = 1{,}000$) are shown in Fig. 7. We can observe that for the ESP problem many methods, such as LSADE, IKAEA, MS-MTO, GORS-SSLPSO, and SSL-A-PSO, had a problem of premature convergence (not being able to improve upon a bad solution found at the $B = 500$ budget). This may be explained by the difficulty these continuous SAEAs face on the discrete problems. Interestingly, the two methods with the most robust performance at the $B = 1{,}000$ budget, TL-SSLPSO and SAMSO, were not the ones that were able to find the overall best solutions (these were IKAEA, LSADE, and BiS-SAHA). Also, in this case, the improvement of TS-DDEO over its base method (SHPSO) was substantial.

Table 4. Statistics of the 24 runs of the selected SAEAs on the ESP model. The best three methods in each category are highlighted in bold.

Method	Min	Mean	Median	Max	Std	Rank
RS	1.05E+00	1.16E+00	1.16E+00	1.28E+00	6.46E-02	10.92
SHPSO	9.19E-01	1.02E+00	1.01E+00	1.18E+00	6.02E-02	6.92
SSL-A-PSO	9.56E-01	1.12E+00	1.11E+00	1.28E+00	9.25E-02	9.92
GORS-SSLPSO	9.21E-01	1.04E+00	1.00E+00	1.19E+00	8.41E-02	7.50
MS-MTO	8.89E-01	1.00E+00	9.88E-01	1.16E+00	8.15E-02	6.04
TL-SSLPSO	8.92E-01	**9.59E-01**	**9.37E-01**	**1.12E+00**	6.27E-02	**3.83**
BiS-SAHA	**8.84E-01**	9.79E-01	**9.63E-01**	1.14E+00	6.54E-02	4.96
IKAEA	**8.63E-01**	**9.74E-01**	9.78E-01	**1.11E+00**	6.17E-02	**4.75**
SAMSO	8.89E-01	**9.66E-01**	**9.65E-01**	1.14E+00	5.04E-02	**4.63**
TS-DDEO	9.14E-01	9.82E-01	9.72E-01	**1.11E+00**	4.85E-02	5.58
ESA	9.13E-01	1.01E+00	1.01E+00	1.12E+00	7.38E-02	6.46
LSADE	**8.78E-01**	1.01E+00	1.01E+00	1.17E+00	6.63E-02	6.50

As with the PitzDaily problem, we also used the Friedman test to find if significant differences are present among all the algorithms on the $B = 500$ and $B = 1{,}000$ budgets. The p-values for this test were 7.51E-15 (and 6.06E-09 when omitting RS) for$B = 500$ and 9.13E-15 (and 3.75-08 when omitting RS) for

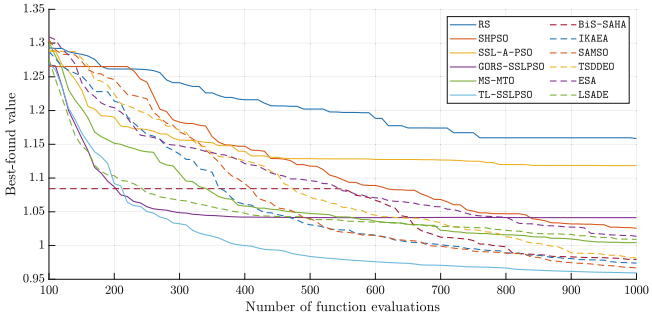


Fig. 6. Convergence of the average best-found value of the selected SAEAs on the ESP problem.

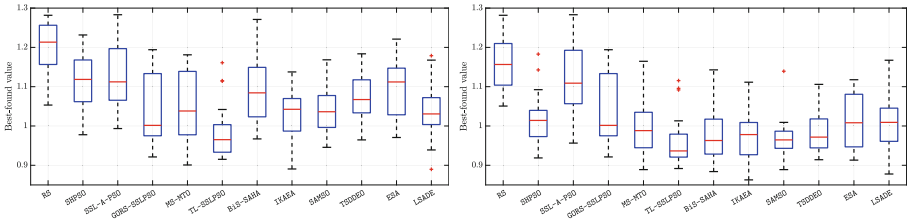


Fig. 7. Boxplots of the solutions of the SAEAs on the ESP problem with $B = 500$ (left) and $B = 1000$ (right).

Table 5. Statistical analysis of the comparison of the selected SAEAs on the ESP problem.

$B = 500$			$B = 1{,}000$		
TL-SSLPSO vs.	p	p^*	TL-SSLPSO vs.	p	p^*
SHPSO	4.97E-05	4.47E-04	SHPSO	1.37E-03	1.24E-02
SSL-A-PSO	3.88E-05	3.88E-04	SSL-A-PSO	3.88E-05	3.88E-04
GORS-SSLPSO	1.29E-02	2.80E-02	GORS-SSLPSO	1.84E-03	1.48E-02
MS-MTO	9.32E-03	2.80E-02	MS-MTO	7.65E-02	3.82E-01✗
BiS-SAHA	1.45E-04	1.01E-03	BiS-SAHA	1.99E-01	5.96E-01✗
IKAEA	1.29E-02	2.59E-02	IKAEA	4.58E-01	6.35E-01✗
SAMSO	4.68E-03	2.34E-02	SAMSO	3.17E-01	6.35E-01✗
TS-DDEO	9.19E-04	5.51E-03	TS-DDEO	8.14E-02	3.82E-01✗
ESA	9.07E-05	7.25E-04	ESA	2.78E-02	1.67E-01✗
LSADE	6.64E-03	2.66E-02	LSADE	1.64E-02	1.15E-01✗

p: p-value computed by the Wilcoxon text
p^*: p-value corrected with the Holm-Bonferroni procedure
✗: statistical differences do not exist with significance level $\alpha = 0.05$

$B = 1{,}000$. All these p-values were much lower than the confidence level $\alpha = 0.05$, so we can state that there are statistically significant differences between the selected SAEAs on the ESP problem. The results of the Wilcoxon signed-rank test between the method with the lowest rank (in this case `TL-SSLPSO`) and the remaining methods are presented in Table 5. These results show the dominance of `TL-SSLPSO` on the $B = 500$ budget. However, there was no statistically significant difference found between `TL-SSLPSO` and seven other methods on the $B = 1{,}000$ budget.

5.3 Aggregate Results

Combining the results of the PitzDaily and ESP problems, we can find the following patterns in the performance of the selected SAEAs. On very low budgets (approximately $B = 200$), the `LSADE` and `GORS-SSLPSO` algorithms were among the best methods in both cases. However, after this budget, `GORS-SSLPSO` seemed to stagnate, while `LSADE` was able to find further improvements. On the $B = 500$ budget, the situation was much more nuanced. Here, `LSADE`, `ESA`, `TS-DDEO`, and `TL-SSLPSO` were the best methods for the PitzDaily problem, while `TL-SSLPSO` dominated the ESP problem. Moreover, both `ESA` and `TS-DDEO` had relatively poor performance on the ESP for the $B = 500$ budget. On the $B = 1{,}000$ budget, the performance of many of the selected SAEAs relatively equalized, and their convergence seemed to have mostly plateaued. For both PitzDaily and ESP problems, `LSADE`, `ESA`, `TS-DDEO`, and `IKAEA` were tied for the spot of the best-performing method (based on the Wilcoxon test). The common features of these methods are their age (as they were all published after 2021), and the fact that they use DE as one of the mechanisms for optimization.

6 Conclusions

Comparing SAEAs on real-world problems instead of artificially created ones is still extremely rare. In this paper, we performed a computational comparison of eleven recently published state-of-the-art single-objective SAEAs on two CFD problems. The aggregate results of the computational comparisons showed that the overall best-performing methods (considering the quality and robustness of the obtained solution as well as their convergence properties) were `LSADE`, `ESA`, `TS-DDEO`, and `IKAEA`. All four of these methods were among the most recent ones (published after 2021), used DE as one of the mechanisms for optimization, and were published in the most high-profile journals (IEEE Transactions on Cybernetics, IEEE Transactions on Evolutionary Computation, and Information Sciences). We hope that these findings will help researchers faced with solving expensive optimization problems in the selection of appropriate methods. We also hope that the authors developing new SAEAs will use the results presented in this paper (and data and code available in the Zenodo repository [31]) for benchmarking.

In further research, we will focus on comparisons of SAEAs with the DIRECT-type approaches [23,56], which are also popular for solving computationally expensive problems. Another line of research lies in finding other good real-world applications, such as in hospital planning [3], inverse heat transfer [36] or robotics [25,32], that can be used for benchmarking SAEAs. Lastly, benchmarking multiobjective SAEAs (and finding appropriate real-world benchmark sets) is also one of the possible research directions.

Acknowledgments. This work was supported by the project IGA BUT No. FSI-S-23-8394 "Artificial intelligence methods in engineering tasks" and by the project (no. 24-12474S) "Benchmarking derivative-free global optimization methods" funded by the Czech Science Foundation.

Disclosure of interest statement. The authors declare no conflicts of interest.

References

1. Aickin, M., Gensler, H.: Adjusting for multiple testing when reporting research results: the bonferroni vs holm methods. Am. J. Public Health **86**(5), 726–728 (1996)
2. Bäck, T.H., et al.: Evolutionary algorithms for parameter optimization-thirty years later. Evol. Comput. **31**(2), 81–122 (2023)
3. Bartz-Beielstein, T., Rehbach, F., Mersmann, O., Bartz, E.: Hospital capacity planning using discrete event simulation under special consideration of the covid-19 pandemic. arXiv preprint arXiv:2012.07188 (2020)
4. Beyer, H.G., Schwefel, H.P.: Evolution strategies-a comprehensive introduction. Nat. Comput. **1**, 3–52 (2002)
5. Bliek, L., Guijt, A., Karlsson, R., Verwer, S., de Weerdt, M.: Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. Appl. Soft Comput. **147**, 110744 (2023)
6. Bliek, L., Verwer, S., de Weerdt, M.: Black-box combinatorial optimization using models with integer-valued minima. Ann. Math. Artif. Intell. **89**, 639–653 (2021)
7. Bujok, P., Lacko, M., Kolenovsky, P.: Differential evolution and engineering problems. Mendel **29**(1), 45–54 (2023)
8. Campana, E.F., et al.: A multi-objective direct algorithm for ship hull optimization. Comput. Optim. Appl. **71**, 53–72 (2018)
9. Catmull, E., Clark, J.: Recursively generated b-spline surfaces on arbitrary topological meshes. In: Seminal Graphics: Pioneering Efforts that Shaped the Field, pp. 183–188 (1998)
10. Chen, Q., Liu, B., Zhang, Q., Liang, J., Suganthan, P., Qu, B.: Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization. Technical report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University (2014)
11. Daniels, S., Rahat, A., Fieldsend, J., Everson, R., et al.: Automatic shape optimisation of the turbine-99 draft tube. In: 12th OpenFOAM Workshop, Exeter. OpenFOAM (2017)

12. Daniels, S., Rahat, A., Tabor, G., Fieldsend, J., Everson, R.: Shape optimisation using computational fluid dynamics and evolutionary algorithms. In: 11th OpenFOAM Workshop, Portugal. OpenDOAM (2016)
13. Daniels, S., Rahat, A., Tabor, G., Fieldsend, J., Everson, R.: A review of shape distortion methods available in the OpenFOAM® framework for automated design optimisation. In: Nóbrega, J.M., Jasak, H. (eds.) OpenFOAM®, pp. 389–399. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-60846-4_28
14. Daniels, S.J., Rahat, A.A.M., Everson, R.M., Tabor, G.R., Fieldsend, J.E.: A suite of computationally expensive shape optimisation problems using computational fluid dynamics. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 296–307. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99259-4_24
15. Díaz-Manríquez, A., Toscano, G., Coello Coello, C.A.: Comparison of metamodeling techniques in evolutionary algorithms. Soft. Comput. **21**, 5647–5663 (2017)
16. Foli, K., Okabe, T., Olhofer, M., Jin, Y., Sendhoff, B.: Optimization of micro heat exchanger: CFD, analytical approach and multi-objective evolutionary algorithms. Int. J. Heat Mass Transf. **49**(5–6), 1090–1099 (2006)
17. García-Martínez, C., Gutiérrez, P.D., Molina, D., Lozano, M., Herrera, F.: Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. Soft. Comput. **21**, 5573–5583 (2017)
18. Gong, W., Liao, Z., Mi, X., Wang, L., Guo, Y.: Nonlinear equations solving with intelligent optimization algorithms: a survey. Complex Syst. Model. Simul. **1**(1), 15–32 (2021)
19. Hellwig, M., Beyer, H.G.: Benchmarking evolutionary algorithms for single objective real-valued constrained optimization-a critical review. Swarm Evol. Comput. **44**, 927–944 (2019)
20. Helton, J.C., Davis, F.J.: Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliab. Eng. Syst. Saf. **81**(1), 23–69 (2003)
21. Holm, S.: A simple sequentially rejective multiple test procedure. Scand. J. Stat. 65–70 (1979)
22. Jin, Y., Wang, H., Chugh, T., Guo, D., Miettinen, K.: Data-driven evolutionary optimization: an overview and case studies. IEEE Trans. Evol. Comput. **23**(3), 442–458 (2018)
23. Jones, D.R., Martins, J.R.: The direct algorithm: 25 years later. J. Global Optim. **79**(3), 521–566 (2021)
24. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optim. **13**, 455–492 (1998)
25. Juricek, M., Parak, R., Kudela, J.: Evolutionary computation techniques for path planning problems in industrial robotics: a state-of-the-art review. Computation **11**(12), 245 (2023)
26. Karlsson, R., Bliek, L., Verwer, S., de Weerdt, M.: Continuous surrogate-based optimization algorithms are well-suited for expensive discrete problems. In: Baratchi, M., Cao, L., Kosters, W.A., Lijffijt, J., van Rijn, J.N., Takes, F.W. (eds.) BNAIC/Benelearn 2020. CCIS, vol. 1398, pp. 48–63. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76640-5_4
27. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995-International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE (1995)
28. Kononova, A.V., Vermetten, D., Caraffini, F., Mitran, M.A., Zaharie, D.: The importance of being constrained: dealing with infeasible solutions in differential evolution and beyond. Evol. Comput. **32**(1), 3–48 (2024)

29. Krithikaa, M., Mallipeddi, R.: Differential evolution with an ensemble of low-quality surrogates for expensive optimization problems. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 78–85. IEEE (2016)
30. Kudela, J.: A critical problem in benchmarking and analysis of evolutionary computation methods. Nat. Mach. Intell. **4**(12), 1238–1245 (2022)
31. Kudela, J.: Zenodo repository: performance comparison of surrogate-assisted evolutionary algorithms on computational fluid dynamics problems (2023). https://doi.org/10.5281/zenodo.8201977
32. Kudela, J., Juříček, M., Parák, R.: A collection of robotics problems for benchmarking evolutionary computation methods. In: Correia, J., Smith, S., Qaddoura, R. (eds.) EvoApplications 2023. LNCS, vol. 13989, pp. 364–379. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30229-9_24
33. Kudela, J., Matousek, R.: New benchmark functions for single-objective optimization based on a zigzag pattern. IEEE Access **10**, 8262–8278 (2022)
34. Kudela, J., Matousek, R.: Recent advances and applications of surrogate models for finite element method computations: a review. Soft. Comput. **26**(24), 13709–13733 (2022)
35. Kudela, J., Matoušek, R.: Combining Lipschitz and RBF surrogate models for high-dimensional computationally expensive problems. Inf. Sci. **619**, 457–477 (2023)
36. Kudela, J., Zalesak, M., Charvat, P., Klimes, L., Mauder, T.: Assessment of the performance of metaheuristic methods used for the inverse identification of effective heat capacity of phase change materials. Expert Syst. Appl. **238**, 122373 (2024)
37. LaTorre, A., Molina, D., Osaba, E., Poyatos, J., Del Ser, J., Herrera, F.: A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. Swarm Evol. Comput. **67**, 100973 (2021)
38. Leary, S.J., Bhaskar, A., Keane, A.J.: A derivative based surrogate model for approximating and optimizing the output of an expensive computer simulation. J. Global Optim. **30**, 39–58 (2004)
39. Li, F., Cai, X., Gao, L., Shen, W.: A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems. IEEE Trans. Cybern. **51**(3), 1390–1402 (2020)
40. Liang, J.J., Qu, B.Y., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, vol. 635, no. 2 (2013)
41. Liao, P., Sun, C., Zhang, G., Jin, Y.: Multi-surrogate multi-tasking optimization of expensive problems. Knowl.-Based Syst. **205**, 106262 (2020)
42. Liu, B., Zhang, Q., Gielen, G.G.: A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. IEEE Trans. Evol. Comput. **18**(2), 180–192 (2013)
43. Loshchilov, I., Schoenauer, M., Sebag, M.: Comparison-based optimizers need comparison-based surrogates. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 364–373. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_37
44. Luo, J., Chen, L., Li, X., Zhang, Q.: Novel multitask conditional neural-network surrogate models for expensive optimization. IEEE Trans. Cybern. **52**(5), 3984–3997 (2020)
45. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1998)

46. Naujoks, B., Willmes, L., Bäck, T., Haase, W.: Evaluating multi-criteria evolutionary algorithms for airfoil optimisation. In: Guervós, J.J.M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., Fernández-Villacañas, J.-L. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 841–850. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45712-7_81
47. Piotrowski, A.P.: Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions. Inf. Sci. **297**, 191–201 (2015)
48. Pitz, R., Daily, J.: An experimental study of combustion the turbulent structure of a reacting shear layer formed at a rearward-facing step. Technical report, NASA Contractor Report 165427, University of California, Berkeley, California, USA (1981)
49. Raponi, E., Rakotonirina, N.C., Rapin, J., Doerr, C., Teytaud, O.: Optimizing with low budgets: a comparison on the black-box optimization benchmarking suite and openai gym. IEEE Trans. Evol. Comput. (2023)
50. Rehbach, F., Rebolledo, M., , Bartz-Beielstein, S.C.T.: GECCO 2021 industrial challenge: optimization of a simulation model for a capacity and resource planning task for hospitals under special consideration of the covid-19 pandemic (2021). https://www.th-koeln.de/informatik-und-ingenieurwissenschaften/gecco-2021-industrial-challenge-call-for-participation_82086.php. Accessed 20 Mar 2023
51. Rehbach, F., Rebolledo, M., Bartz-Beielstein, T.: GECCO 2020 industrial challenge: optimizing expensive computational fluid dynamics simulations (2020). https://www.th-koeln.de/informatik-und-ingenieurwissenschaften/gecco-challenge-2020_72989.php. Accessed 20 Mar 2023
52. Rehbach, F., Zaefferer, M., Stork, J., Bartz-Beielstein, T.: Comparison of parallel surrogate-assisted optimization approaches. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1348–1355 (2018)
53. Ren, Z., Sun, C., Tan, Y., Zhang, G., Qin, S.: A bi-stage surrogate-assisted hybrid algorithm for expensive optimization problems. Complex Intell. Syst. **7**, 1391–1405 (2021)
54. Serani, A., et al.: Ship hydrodynamic optimization by local hybridization of deterministic derivative-free global algorithms. Appl. Ocean Res. **59**, 115–128 (2016)
55. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997)
56. Stripinis, L., Kůdela, J., Paulavičius, R.: Benchmarking derivative-free global optimization algorithms under limited dimensions and large evaluation budgets. IEEE Trans. Evol. Comput. (2024)
57. Sun, C., Jin, Y., Tan, Y.: Semi-supervised learning assisted particle swarm optimization of computationally expensive problems. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 45–52 (2018)
58. Tzanetos, A., Dounias, G.: Nature inspired optimization algorithms or simply variations of metaheuristics? Artif. Intell. Rev. **54**, 1841–1862 (2021)
59. Wang, H., Jin, Y., Sun, C., Doherty, J.: Offline data-driven evolutionary optimization using selective surrogate ensembles. IEEE Trans. Evol. Comput. **23**(2), 203–216 (2018)
60. Weller, H.G., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput. Phys. **12**(6), 620–631 (1998)
61. Yu, H., Kang, L., Tan, Y., Sun, C., Zeng, J.: Truncation-learning-driven surrogate assisted social learning particle swarm optimization for computationally expensive problem. Appl. Soft Comput. **97**, 106812 (2020)

62. Yu, H., Tan, Y., Sun, C., Zeng, J.: A generation-based optimal restart strategy for surrogate-assisted social learning particle swarm optimization. Knowl.-Based Syst. **163**, 14–25 (2019)
63. Yu, H., Tan, Y., Zeng, J., Sun, C., Jin, Y.: Surrogate-assisted hierarchical particle swarm optimization. Inf. Sci. **454**, 59–72 (2018)
64. Zhan, D., Xing, H.: A fast kriging-assisted evolutionary algorithm based on incremental learning. IEEE Trans. Evol. Comput. **25**(5), 941–955 (2021)
65. Zhen, H., Gong, W., Wang, L.: Evolutionary sampling agent for expensive problems. IEEE Trans. Evol. Comput. **27**(3), 716–727 (2022)
66. Zhen, H., Gong, W., Wang, L., Ming, F., Liao, Z.: Two-stage data-driven evolutionary optimization for high-dimensional expensive problems. IEEE Trans. Cybern. **53**(4), 2368–2379 (2021)
67. Zhou, Y., He, X., Chen, Z., Jiang, S.: A neighborhood regression optimization algorithm for computationally expensive optimization problems. IEEE Trans. Cybern. **52**(5), 3018–3031 (2020)
68. Zhou, Z., Ong, Y.S., Nair, P.B., Keane, A.J., Lum, K.Y.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **37**(1), 66–76 (2006)

Balancing Between Time Budgets and Costs in Surrogate-Assisted Evolutionary Algorithms

Cedric J. Rodriguez[1(✉)], Peter A. N. Bosman[2], and Tanja Alderliesten[1]

[1] Leiden University Medical Center, 2300 RC Leiden, The Netherlands
{c.j.rodriguez,t.alderliesten}@lumc.nl
[2] Centrum Wiskunde and Informatica, 1090 GB Amsterdam, The Netherlands
peter.bosman@cwi.nl

Abstract. For many real-world multi-objective optimisation problems, function evaluations are computationally expensive, resulting in a limited budget of function evaluations that can be performed in practice. To tackle such expensive problems, multi-objective surrogate-assisted evolutionary algorithms (SAEAs) have been introduced. Often, the performance of these EAs is measured after a fixed number of function evaluations (typically several hundreds) and complex surrogate models are found to be the best to use. However, when selecting an SAEA for a real-world problem, the surrogate building time, surrogate evaluation time, function evaluation time, and available optimisation time budget should be considered simultaneously. To gain insight into the performance of various surrogate models under different conditions, we evaluate an EA with and without four surrogate models (both complex and simple) for a range of optimisation time budgets and function evaluation times while considering the surrogate building and surrogate evaluation times. We use 55 BBOB-BIOBJ benchmark problems as well as a real-world problem where the fitness function involves a biomechanical simulation. Our results, on both types of problems, indicate that a larger hypervolume can be obtained with SAEAs when a function evaluation takes longer than 0.384 s (on the hardware we used). While we confirm that state-of-the-art complex surrogate models are mostly the best choice if up to several hundred function evaluations can be performed, we also observe that simple surrogate models can still outperform non-surrogate-assisted EAs if several thousand function evaluations can be performed.

Keywords: Expensive optimisation · Surrogate-Assisted Evolutionary Algorithms · Real-world problems · Biomechanical simulations

1 Introduction

Function evaluations of real-world multi-objective (MO) optimisation problems are often computationally expensive, costing multiple seconds to several hours.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 322–339, 2024.
https://doi.org/10.1007/978-3-031-70068-2_20

While evolutionary algorithms (EAs) are known to be effective at optimizing MO problems, they typically require well over a hundred thousand function evaluations to get (near-)optimal solutions (for non-trivial problems). This is not feasible for many expensive real-world problems. For this reason, MO surrogate-assisted EAs (SAEAs) have been introduced [20] in which many of the expensive-to-evaluate objective functions are replaced with quick-to-evaluate models (surrogates). The surrogate is typically built on all expensive function evaluations performed so far and is continuously updated during optimisation.

MO SAEAs, such as K-RVEA [10], AB-SAEA [35], CSEA [25], MOEA/D-EGO [39], or ADSAPSO [24], are often evaluated on DTLZ [13] benchmark problems with a fixed function evaluation budget (typically a few hundred). The efficacy of these SAEAs in scenarios that permit a "medium" budget of thousands of function evaluations has not been thoroughly investigated. Moreover, the time required for building and evaluating surrogate models is often not considered. This places non-surrogate-assisted EAs increasingly at a disadvantage as the expensiveness of function evaluations decreases, making it hard to assess the value of a surrogate model. For real-world problems, where optimisation time is frequently a limiting factor, it is crucial to assess surrogate models across various total optimisation time budgets instead of solely based on function evaluations.

SAEAs have been used for various real-world problems [19], e.g., energy-efficient design [6,8,14,23,33,36,38], motor manufacturing [29,31], ship design [1,17], automobile design [2,30,34], satellite design [28], antenna design [21,37,41], and energy and power [16]. In these works, total optimisation time is often disregarded as comparisons are performed at a fixed number of function evaluations or generations, possibly overestimating the *general* practical usability of SAEAs compared to non-surrogate-assisted EAs. Some works do provide the total optimisation time of different algorithms [23,28,36] and others do consider different optimisation time budgets [37]. No previous work has, however, varied *both* the optimisation time budget and the function evaluation time to systematically evaluate the performance of SAEAs for real-world problems.

We therefore emphasize comparing surrogate models under different total optimisation time budgets, following the strategy in [42], where surrogates based on Gaussian processes and polynomial regression were compared across varied time budgets. This however concerned single-objective optimisation, whereas we focus on MO optimisation which changes the balance between optimisation time and (surrogate) evaluation time budgets. By varying both the total optimisation time and the function evaluation time, we aim to cover the full spectrum of scenarios encountered in real-world applications, offering a more comprehensive evaluation of MO SAEAs so as to ultimately consider the following question: Given a spectrum of function evaluation times and optimisation time budgets, should an MO SAEA be considered and if yes, what type of surrogate model should be used?

The remainder of this paper is organized as follows: the MO SAEA and the surrogate models considered are described in Sect. 2. Our methodology and experimental setup are described in Sects. 3 and 4, respectively. In Sect. 5, the

results on the benchmark problems are presented. In the Sects. 6 and 7, we apply the methodology to a real-world application. Finally, we present our conclusions in Sect. 8.

2 Background

2.1 MO SAEA

The MO SAEA that we consider is outlined in Algorithm 1. This algorithm is similar to [27] where each objective is modelled by a separate surrogate. In the initialization phase, a set of solutions is sampled using Latin Hypercube Sampling (LHS). These solutions are then evaluated using the expensive (true) function, and are subsequently used to build an initial surrogate model. The generational process can be decomposed into an inner cycle and an outer cycle. In the inner cycle, a population is initialized that contains all solutions previously evaluated with the true function. Selection and variation operators are applied, and offspring solutions are evaluated using the surrogate model instead of the true function. When the maximum number of surrogate evaluations for an inner cycle is reached, the inner cycle is terminated. All surrogate-evaluated solutions are then considered in the outer cycle as potential candidate solutions for evaluation with the true function. The most promising solutions are then selected as described in Sect. 4.2 and evaluated using the true function. Subsequently, a surrogate model is rebuilt based on all true-evaluated solutions so far, and the next iteration of the inner cycle commences.

Algorithm 1. MO SAEA

Input: N = Number of initial solutions; FE_{max}^{true} = maximum number of true function evaluations; $FE_{max}^{surrogate}$ = maximum number of surrogate evaluations in each inner cycle; μ = number of new selected solutions in the outer cycle

Output: A = Archive of all true evaluated solutions

1: Initialize population P of size N using random sampling
2: Evaluate all solutions in P using the true function
3: Initialize archive A of all true evaluated solutions with all solutions in P
4: Set number of true function evaluations FE^{true} to N
5: **while** $FE^{true} < FE_{max}^{true}$ **do** ▷ *Outer Cycle*
6: Build surrogate model M based on A
7: Continue to optimize P using surrogate model M in EA with $FE_{max}^{surrogate}$ ▷ *Inner Cycle*
8: Select μ solutions from all surrogate evaluated solutions
9: Evaluate μ selected solutions with true function
10: $FE^{true} = FE^{true} + \mu$
11: Add true evaluated solutions to archive A
12: **end while**
13: Return A

2.2 Surrogates

We consider two commonly adopted surrogates of varying complexity as well as two surrogates that are not considered often, but are of very low complexity.

Gaussian Processes. The most popular surrogate, and often considered to be state-of-the-art for expensive optimisation, is Gaussian process regression (e.g., Kriging, also much used in Bayesian optimisation) [35]. It is defined as follows:

$$\hat{y}^{kriging}(x) = \beta + r^T(x)R^{-1}(y - F(\beta, x)\beta) \tag{1}$$

where β are the coefficients, R is the correlation matrix between all the training samples, $r^T(x)$ represents the correlation vector between the solution x and the training samples, y represents the fitness values of the training set of solutions, and $F(\beta, x)$ is the regression model. [10] This surrogate model requires maximizing a likelihood function for the hyperparameters which is computationally expensive.

Radial Basis Functions. This surrogate is a weighted aggregation of basis functions $\Psi(\cdot)$:

$$\hat{y}^{rbf}(x) = \sum_{i=1}^{n} w_i \Psi_{i,j} \tag{2}$$

In this paper, we utilize Gaussian radial basis functions, where the centre of each Gaussian corresponds to each solution that the surrogate model is built upon. The influence of each basis function is calculated using the squared Euclidean distance from the point being evaluated, i.e., $\Psi_{i,j} = e^{-\gamma d_{i,j}^2}$ where $d_{i,j}$ represents the Euclidean distance between the i-th and j-th solutions, and γ is a scalar that determines the spread of the basis functions. The weights are determined through generalized linear regression, which involves computing the pseudo-inverse of the Ψ matrix. [7] This is much faster than what is needed to build the Kriging model.

K-Nearest Neighbours Linear Regression. We also consider a simpler surrogate that leverages linear regression. The local structure of the problem landscape is however captured by using only the k-nearest neighbours (knn) of the solution of interest.

$$\hat{y}^{lr\text{-}knn}(x) = \text{mode}\left\{y_i \mid (x_i, y_i) \in \text{Nearest}_k(x, \mathcal{D})\right\} \tag{3}$$

where $\mathcal{D}$ is the training dataset consisting of pairs (x_i, y_i), where x_i are the decision vectors and y_i are the fitness values. The function $\text{Nearest}_k(x, \mathcal{D})$ returns the decision vectors and the fitness values of k-nearest neighbours using Euclidean distance in the decision space of x in $\mathcal{D}$. The mode function determines the desired interpolation function between these neighbours. This model, we use linear regression as the interpolation function defined as:

$$\hat{y}^{lr\text{-}knn}(x) = \beta_0 + \sum_{q=1}^{D} \beta_q x_q \tag{4}$$

where β_0 is the intercept, β_q are the coefficients (slopes) in each problem dimension, D is the number of problem dimensions, and x_q are the decision variables.

Nearest Neighbour. Finally, we also consider one of the simplest surrogates - solely considering the nearest neighbour. This surrogate model is defined as:

$$\hat{y}^{nn}(x) = \text{Nearest}(x, \mathcal{D}) \tag{5}$$

where $\mathcal{D}$ is the training dataset and the function $\text{Nearest}(x, \mathcal{D})$ returns the fitness of the nearest neighbour of x in $\mathcal{D}$ in terms of Euclidean distance in decision space.

2.3 Reference Vector Guided EA

The baseline MOEA that we use in this work, is a Reference Vector Guided Evolutionary Algorithm (RVEA) [9]. In RVEA, offspring solutions are generated using simulated binary crossover [11] and polynomial mutation, comparable to what is used in other MOEAs like NSGA-III [12]. The selection of the parent population for the subsequent generation of offspring, is performed using a technique known as angle penalized distance, which is guided by reference vectors. RVEA is often used within a surrogate-assisted approach, including some state-of-the-art expensive optimisation approaches for MO optimisation [10,35], which is why we consider this EA here as well.

2.4 MAMaLGaM

For the real-world application, we additionally consider an estimation-of-distribution algorithm (EDA), in particular iMAMaLGaM-X with a multivariate joint Gaussian distribution [4]. In part, this is because an EDA can be considered a different type of model-based EA, but also, this algorithm was previously used to optimize a simpler version of the real-world problem [26] that we also consider in this work. In iMAMaLGaM-X, solutions are selected from the parent population and elitist archive, and are organized into a user-defined number of clusters (K). A Gaussian distribution is estimated and adaptively scaled for each cluster, from which new offspring solutions are sampled. For more details, see [4].

3 Methodology

To get insights into the impact of the computational expensiveness of the true fitness function as well as the computational expensiveness associated with the

surrogate model, we run EAs and SAEAs using benchmark problems. For benchmark problems, the true function evaluation time is in the order of milliseconds and is thus negligible. We use this fact to subsequently estimate the time an algorithm would have taken if the true fitness function had been more expensive.

To this end, for each combination of problem instance and algorithm, we run an optimisation process. After each true function evaluation FE^{true}, the measured execution time t^m_{FE-1} is recorded. Because the time taken to evaluate a benchmark problem is negligible, the execution time t^m_{FE-1} represents time taken by the algorithm in between true function evaluations. For EAs, this is the time required for variation and selection and for SAEAs, this is the time for surrogate building and optimisation using the surrogate model.

We can now estimate the total optimisation time $\hat{t}_{FE}$ of an algorithm using

$$\hat{t}_{FE} = t^m_{FE-1} + \delta FE \tag{6}$$

where the execution time t^m_{FE-1} is the surrogate building time and surrogate evaluation time, FE is the number of true function evaluations and δ is the true function evaluation time.

In this paper, we consider $\delta = 0.0001 \cdot 2^i$ minutes for $i \in \{0, 1, \ldots, 20\}$ and a total optimisation time budget of $16 \cdot 2^j$ minutes for $j \in \{0, 1, \ldots, 11\}$.

4 Experimental Setup

In this section, we describe the considered benchmark problems, algorithm settings used, and details regarding how we analysed the results.

4.1 Benchmark Problems

We consider the bi-objective BBOB functions (BBOB-BIOBJ) [5] with 20 continuous problem variables from the COCO framework [18]. We only consider the first instance of the 55 problems, since we are not exploring rotation and translational invariances of algorithms. Each BBOB-BIOBJ problem is composed of a pair-wise combination of 10 single objective BBOB functions. The algorithms are initialized in the default COCO ranges: -100 to 100. The termination criteria are set to $10,000$ true function evaluations or a maximum runtime of 3 days, whichever comes first. We execute 15 repeats per optimisation. A high-performance computing cluster is used, composed of AMD EPYC 7H12 nodes, each containing 2 GB of memory and 64 cores with a 2.6GHz clock speed per node. Octave version 7.3.0 and the COCO framework [18][1] are used to perform all experiments.

[1] https://github.com/numbbo/coco.

4.2 Algorithm Settings

In this work, we evaluate four SAEAs where two SAEAs utilise state-of-the-art surrogates, namely *Kriging* from [10] and *rbf* from [22] as implemented in [32]. We also introduce two quick-to-compute surrogates namely *nn* and *lrknn* as described in Sect. 2.2. As a baseline, we also consider *rvea* without any surrogate assistance. This leads to five different algorithms in total: *kriging-rvea*, *rbf-rvea*, *nn-rvea*, *lrknn-rvea*, and *rvea*.

All algorithms start with an initial population of 32 solutions, sampled using LHS. Each repeat is completely independent, using a different seed number. This is the first multiple of 16 (available parallel cores for the real-world application) larger than the problem's dimension of 20. After a surrogate optimisation cycle of 2000 surrogate evaluations using *rvea*, for all SAEA variants except *kriging-rvea*, the potential solutions for true evaluation are determined by pre-selecting the non-dominated solutions in terms of the surrogate fitness values. For *Kriging*, we utilise the acquisition function for the pre-selection as was described in [35] and was implemented for Bayesian optimisation in the PlatEMO framework [32]. This acquisition function prioritizes solutions with uncertain fitness estimation at the beginning of the optimisation and solutions with good fitness estimations at the end of the optimisation. This is reported in [35] to provide higher hypervolume on the DTLZ [13] and UF [40] bi-objective benchmark problems compared to [10]. As suggested in [35], we also use an archive management method for the *Kriging* surrogate, limiting the archive size to 320 solutions.

For all SAEA variants, we then randomly select μ solutions from this pre-selection. In this work, we have used $\mu = 1$ so that the model is updated as frequently as possible (every new true evaluation). Finally, *rvea* is used as implemented by [9] in PlatEMO [32]. In this implementation, the parameter α prioritizing convergence is set to $100,000$ and the population size is set to 32.

For the *lrknn* surrogate, $k = 32$ is used to fit a linear slope through the training samples at the beginning of the optimisation and to capture the local structure of the problem landscape at the end of the optimisation. Finally, *nn-rvea* has no additional parameters to set.

4.3 Evaluation of Results

For each (estimated) optimisation time budget and expensive function evaluation time, the two algorithms are selected that exhibit the highest and second-highest median hypervolume over the 15 repeats on a BBOB-BIOBJ problem. The reference point is set to $[1, 1]$, as suggested by [18]. We also evaluate whether there is a statistically significant difference between the hypervolume of the best and second-best performing algorithms using a Wilcoxon test. When considering 55 BBOB-BIOBJ problems, 12 optimisation time budgets, and 21 expensive function evaluation times, this leads to 13,860 statistical tests. We consider statistical significance to be $p \leq 0.05$ and correct this using the Bonferroni correction. To get an aggregated view of an algorithm's performance over all BBOB-BIOBJ problems, we estimate, for each combination of optimisation time budget and

expensive function evaluation time, the frequency that a particular algorithm has been selected as the best-performing algorithm.

5 Results on Benchmark Problems

The best performing algorithms on the BBOB-BIOBJ problems 1 to 19 are visualised using a heatmap, see Fig. 1. The BBOB-BIOBJ problems 20 to 55 can be found in the Supplementary Material[2]. Because the first 32 function evaluations correspond to the initial randomly sampled solutions in the population, we shaded the regions where the total optimisation time is smaller than 32 times the function evaluation time. The regions with high optimisation times and low function evaluation times are also shaded because due to the termination criteria of 10,000 evaluations, no data was gathered for this region.

It can be observed in Fig. 1 and Fig. 2 that *rvea* typically has the highest median hypervolume for function evaluation times below 0.0064 min (0.384 s). Furthermore, *rvea* only has a higher median hypervolume if the total optimisation time is below 32 min. This is mostly due to the limit of 10,000 evaluations. If this budget was set larger, *rvea* is expected to also outperform all other algorithms for the longer total optimisation times.

As expected, the current state-of-the-art surrogate models such as *kriging* and *rbf* lead to higher median hypervolumes obtained with the SAEA for smaller budgets of function evaluations, which corresponds to the left-upper envelope of the graphs, see Fig. 2. In particular, for each row on the diagonal of the left-upper envelope, the first three columns represent up to 80 function evaluations. Interestingly, for some BBOB-BIOBJ problems (e.g., 7, 8, 13, 15) in Fig. 1, *rvea* still obtained a higher median hypervolume in these regions. A likely reason for this is that the considered surrogates are not effective at modelling the problem with such a limited number of function evaluations. Finally, and interestingly, on almost all problems the use of "simple" surrogate models in our SAEA, i.e., *nn-rvea* and *lrknn-rvea*, obtains a higher median hypervolume for "medium"-range expensive settings in which up to 10,000 function evaluations can be performed, with 10,000 evaluations corresponding to the bottom-right envelope in Fig. 2. Only in a few cases does *rvea* become the dominating algorithm again when 10,000 evaluations are used.

6 Real-World Application: A Biomechanical Simulation

While the BBOB-BIOBJ problems are certainly of importance and value, they may still differ from a real-world application in which a true expensive, and often complex, optimisation problem needs to be solved.

We therefore also consider a real-world optimisation problem in this work that is computationally demanding: biomechanical simulation optimisation for the purpose of deformable medical image registration in radiation treatment of

[2] https://zenodo.org/records/10992139.

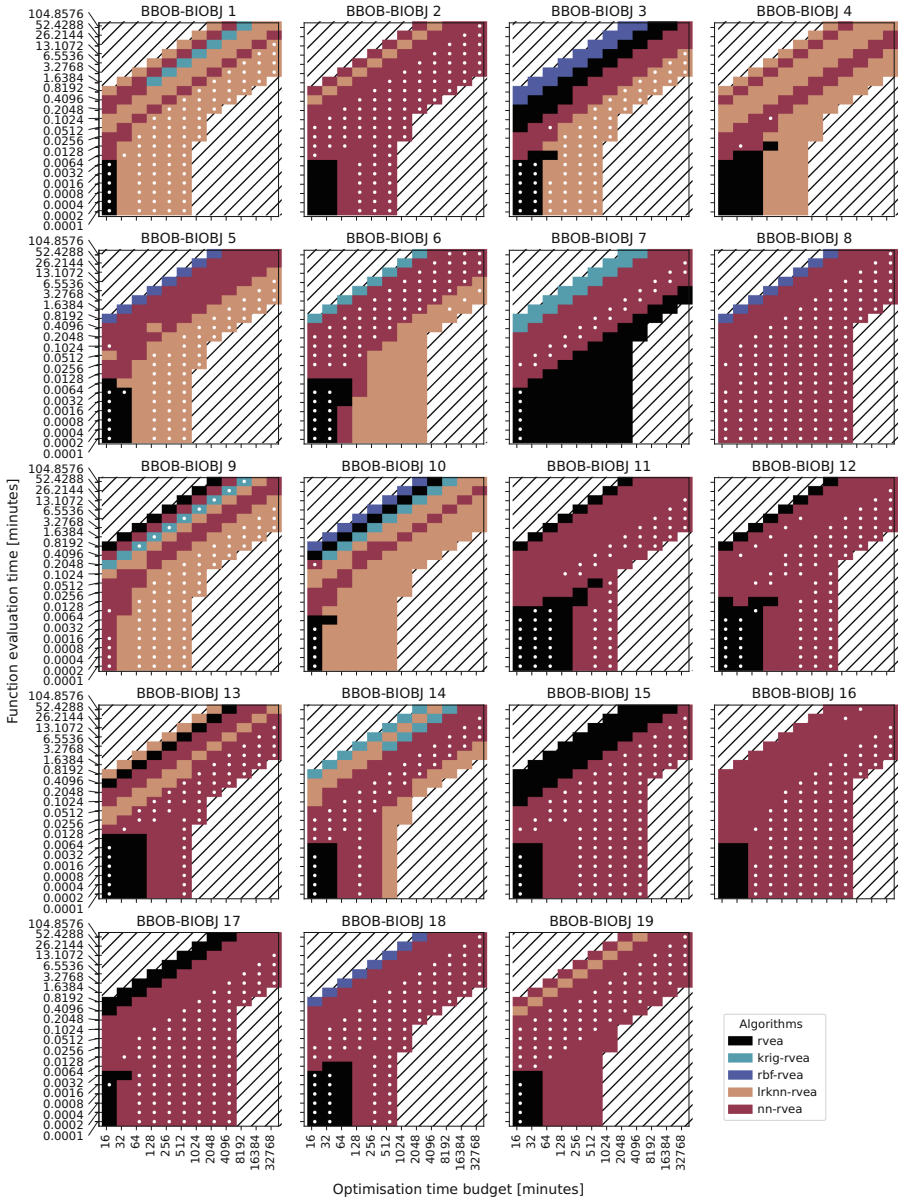


Fig. 1. Heatmap per BBOB-BIOBJ problem, visualising the algorithm with the highest median hypervolume given an optimisation time and function evaluation time. The white dots indicate if there is a statistical difference between the best and second-best performing algorithm.

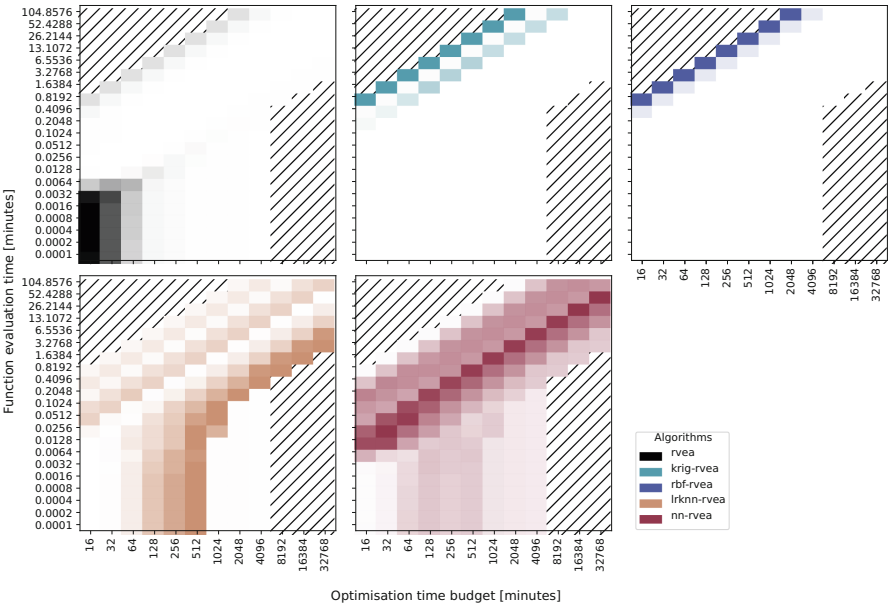


Fig. 2. Heatmap per algorithm, where a higher colour intensity indicates, given an optimisation time and function evaluation time, a higher frequency of highest median hypervolume over all BBOB-BIOBJ problems.

cervical cancer. In radiation treatment, multiple medical images of the same patient are acquired, see Fig. 3. Given that organs in the pelvic region are highly deformable for various reasons, there is often an inherent mismatch between the shapes of organs in any pair of medical images. Deformable image registration is aimed at finding the physically correct spatial mapping between two medical image pairs to transfer spatial information, such as dosimetric data.

In this work, we consider a biomechanical approach to deformable image registration in which the organs in one image are contoured by a medical professional and are subsequently used to construct a mesh representation that can be used inside a finite element method (FEM) simulation. Tissue characteristics, particularly related to elasticity, are adhered to different parts of the mesh. Subsequently, forces are defined that are applied to specific regions in the mesh. A FEM simulation then adjusts the mesh nodes according to the forces applied. Consequently, the underlying organ contours are deformed.

The optimisation task consists of finding free simulation parameters such that FEM simulation results in a mesh transformation that ensures the transformed organ contours are aligned with those in the second image. Each function evaluation necessitates a full FEM simulation, which is a computationally expensive operation.

In this work, we focus on the deformable image registration of 3D pelvic computed tomography (CT) scans for patients undergoing external beam radiation

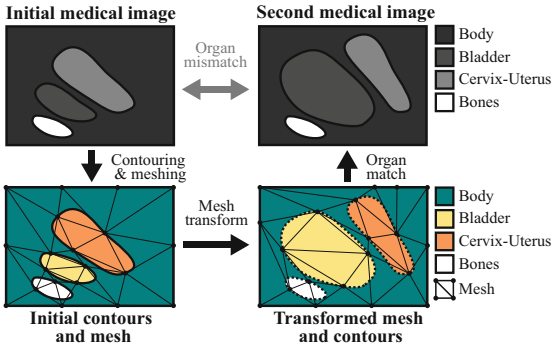


Fig. 3. An illustrative representation of deformable image registration, where the top two images represent medical images of the same patient with a difference in bladder volume and the bottom two images depict the contoured organs overlayed with the finite element method mesh that is deformed to align the two images.

treatment for cervical cancer. The treatment planning involves acquiring two 3D CT scans: one with an empty bladder and another with a full bladder. In the following section, we provide an overview of the process. For more details, we refer the interested reader to the Supplementary Material.

Contouring and Mesh Generation. Initially, a 3D CT scan with an empty bladder is used for contouring important structures such as the body, bladder, bones, cervix-uterus, bowel, and rectum. Subsequently, the contoured images are converted into triangular surface meshes for each organ, which are then integrated into a unified tetrahedral mesh for each patient. Each tetrahedron within this mesh is associated with a specific organ, facilitating a detailed anatomical representation of the patient with an empty bladder.

Mesh Transformation Biomechanical FEM Simulation. The organs in the empty bladder scan are deformed according to the FEM simulation. To optimize the match between the so-deformed organs and those in the full bladder CT scan, the parameters corresponding to various forces to be applied to the mesh are optimized. Specifically, 19 continuous simulation parameters are considered.

Organ Comparison. To determine the quality of the resulting FEM-based deformation, we utilize the Dice similarity coefficient (DSC) score [15]. The DSC score evaluates the overlap between the two volumes, with scores ranging from 0 (no volumetric overlap) to 1 (perfect volumetric overlap). However, relying solely on DSC scores could still result in large and unrealistic organ deformations. Therefore, a second objective, namely deformation energy, is considered that is

related to the magnitude of the resulting deformation. Specifically, this measure quantifies the energy required for organ deformation, calculated in Joules by summing the energy needed to deform each tetrahedral element in the patient's mesh based on its biomechanical properties. This bi-objective approach ensures that we obtain results representing different trade-offs between accurate organ registration and physical realism in the deformation process which can subsequently be inspected by a medical professional so as to ultimately decide which deformation is the preferred one to use.

Problem Settings, Algorithm Settings and Result Evaluation. Optimizing biomechanical simulations for seven patients involves stopping at either 10,000 simulations or after five days. Since this function evaluation time is not negligible compared to the surrogate optimisation time, we directly record the time duration of the surrogate optimisation cycles (excluding the function evaluation times). We run 16 simulations in parallel using 16 cores and repeat each optimisation 10 times. The same algorithms are used as in Sect. 4.2 as well as iMAMaLGaM-X. The settings for iMAMaLGaM-X are based on the guidelines in literature [3], except for the population size, which is set to 32 (equal to *rvea*). A high-performance computing system is used, composed out of nodes, each containing 2 GB of memory per core and 16-128 Intel E5/Gold series cores. The optimisation uses Octave version 7.3.0 and the simulation uses SOFA[3] v.21.12. The results are evaluated using the same procedure as in Sect. 4.3. The hypervolume is calculated using the reference point of $[0, 5000]$, which has been established empirically.

7 Results on Real-World Application

The best performing algorithms are visualised per patient using a heatmap, see Fig. 4. Here, we see, similar to the benchmarks, that *rvea* performs best for a function evaluation time under 0.0064 min (0.384 s). Furthermore, *kriging* and *rbf* surrogates seem to only show a benefit for very small evaluation budgets. Interestingly, iMAMaLGaM-X is consistently effective for intermediate evaluation budgets which constitutes a large portion of the heatmap and the use of the *nn* surrogate in RVEA is best around the 10,000 evaluations budget limits. No statistically significant differences were found.

The average simulation in the experiments took approximately 48 s. With 16 parallel cores, this results in an effective simulation time of 3 s (0.05 min). Figures 4 and 5 indicate that, for this function evaluation time and optimisation budget of 4 h (240 min), *nn-rvea* leads to the highest hypervolume on all 7 patients. Nevertheless, the border to iMAMaLGaM-X and *rvea* often is one block away, meaning that a slight change in function evaluation time or optimisation

[3] https://github.com/sofa-framework/sofa.

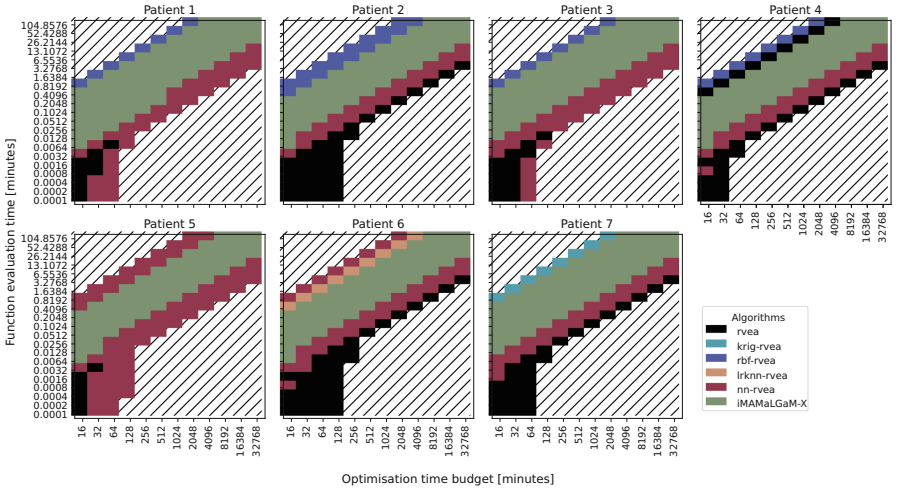


Fig. 4. Heatmap per patient, visualising the algorithm with the highest median hypervolume given an optimisation time and function evaluation time.

budget, could change the best choice of optimizer. For example, say we expect the simulation to increase in simulation time due to increased simulation complexity. This would mean, although *nn-rvea* shows a higher hypervolume now, it could then be better to select iMAMaLGaM-X as optimizer for this real-world application. When looking at the approximation front for Patient 1 in Fig. 6, we see that both *nn-rvea* and iMAMaLGaM-X, find evenly distributed solutions along the front with good DSCs of up to 0.8.

While results for different patients do not differ hugely, for different evaluation budgets and different patients, sometimes different algorithms appear as the best choice. The general approach presented in this work can therefore not be used to make patient-specific recommendations about the best SAEA to use, but rather gives generic insight into which algorithms are of interest for a certain total optimisation budget (i.e., a pre-selection). However, if desired, further refinement of algorithm selection could be done using (online) landscape analysis of both the true and surrogate fitness landscapes, as for instance in [27].

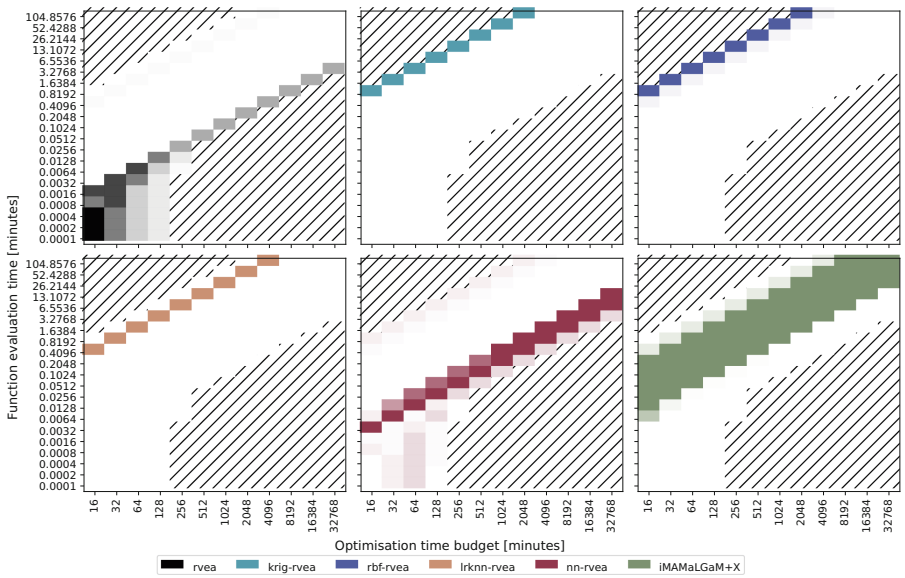


Fig. 5. Heatmap per algorithm, where a higher colour intensity indicates a higher frequency of highest median hypervolume over all patients.

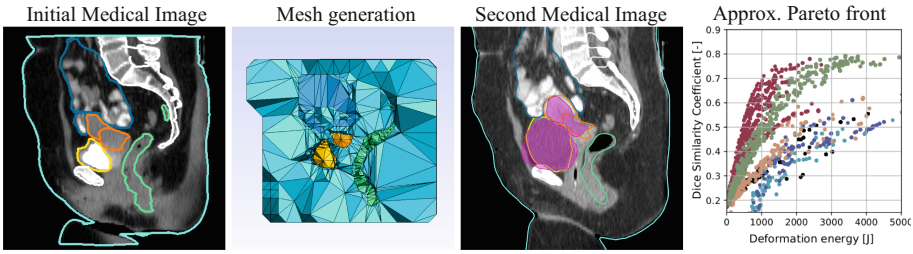


Fig. 6. optimisation process and results of patient 1, where pink visualizes the deformed bladder and cervix-uterus corresponding to the solution with the highest DSC score after optimisation.

8 Conclusions

In this work, we addressed a mostly overlooked aspect of different types of time considerations in evaluating multi-objective surrogate-assisted evolutionary algorithms (MO SAEAs), by incorporating surrogate building and surrogate evaluation times alongside true function evaluation times and total optimisation time budgets. We compared four surrogate models across 55 benchmark problems and a real-world medical application. We demonstrated that MO EAs seemed to be more effective than MO SAEAs for functions that have an evaluation time that is shorter than 0.384 s. Of course, this particular number is strongly related to the used hardware and types of surrogate models and their implementation.

However, the approach used here is a general one that can be repeated on other hardware and with different implementations within a different context (e.g., with more parallel computing power) to obtain problem- and computational-setup specific results if need be.

Our results showed that while state-of-the-art surrogate models like Kriging and radial basis functions excel up to several hundred function evaluations, simpler models such as nearest neighbour and linear regression emerge as effective options in case more function evaluations can be performed (i.e., less expensive problems or larger available computational power/budget). This work also motivates that more research should be done on reducing surrogate building and evaluation time (especially as the number of training samples increase) or the use of sparse surrogate models.

Acknowledgments. The research is part of the research programme Open Technology Programme with project number 15586, which is financed by the Dutch Research Council (NWO), Elekta (Elekta Solutions AB, Stockholm, Sweden), and Xomnia (Xomnia B.V., Amsterdam, The Netherlands). Further, the work is co-funded by the public-private partnership allowance for top consortia for knowledge and innovation (TKIs) from the Ministry of Economic Affairs.

Disclosure of Interests. All authors are involved in one or more projects supported by Elekta AB, Stockholm, Sweden. Elekta had no involvement in the study design, the data collection, analysis and interpretation, or the writing of the paper.

References

1. Ayob, A.F.M., Ray, T., Smith, W.F.: Beyond hydrodynamic design optimization of planing craft. J. Ship Prod. **27**(1), 1–13 (2011)
2. Bhattacharjee, D., Ghosh, T., Bhola, P., Martinsen, K., Dan, P.K.: Data-driven surrogate assisted evolutionary optimization of hybrid powertrain for improved fuel economy and performance. Energy **183**, 235–248 (2019)
3. Bosman, P.A.: On empirical memory design, faster selection of Bayesian factorizations and parameter-free gaussian EDAs. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 389–396 (2009)
4. Bosman, P.A.: The anticipated mean shift and cluster registration in mixture-based EDAs for multi-objective optimization. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, pp. 351–358 (2010)
5. Brockhoff, D., Auger, A., Hansen, N., Tušar, T.: Using well-understood single-objective functions in multi-objective black-box optimization test suites. Evol. Comput. **30**(2), 165–193 (2022)
6. Chegari, B., Tabaa, M., Simeu, E., Moutaouakkil, F., Medromi, H.: Multi-objective optimization of building energy performance and indoor thermal comfort by combining artificial neural networks and metaheuristic algorithms. Energy Build. **239**, 110839 (2021)
7. Chen, G., et al.: A radial basis function surrogate model assisted evolutionary algorithm for high-dimensional expensive optimization problems. Appl. Soft Comput. **116**, 108353 (2022)

8. Chen, X., Yang, H.: Integrated energy performance optimization of a passively designed high-rise residential building in different climatic zones of china. Appl. Energy **215**, 145–158 (2018)
9. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Trans. Evol. Comput. **20**(5), 773–791 (2016)
10. Chugh, T., Jin, Y., Miettinen, K., Hakanen, J., Sindhya, K.: A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. IEEE Trans. Evol. Comput. **22**(1), 129–142 (2016)
11. Deb, K., Agrawal, R.B., et al.: Simulated binary crossover for continuous search space. Complex Syst. **9**(2), 115–148 (1995)
12. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. IEEE Trans. Evol. Comput. **18**(4), 577–601 (2013)
13. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC 2002 (Cat. No. 02TH8600), vol. 1, pp. 825–830. IEEE (2002)
14. Dhariwal, J., Banerjee, R.: An approach for building design optimization using design of experiments. Build. Simul. **10**, 323–336 (2017)
15. Dice, L.R.: Measures of the amount of ecologic association between species. Ecology **26**(3), 297–302 (1945)
16. Dong, Q., Wang, C., Peng, S., Wang, Z., Liu, C.: A many-objective optimization for an eco-efficient flue gas desulfurization process using a surrogate-assisted evolutionary algorithm. Sustainability **13**(16), 9015 (2021)
17. Habib, A., Singh, H.K., Ray, T.: A multiple surrogate assisted evolutionary algorithm for optimization involving iterative solvers. Eng. Optim. **50**(9), 1625–1644 (2018)
18. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black-box setting. Optim. Methods Softw. **36**, 114–144 (2021). https://doi.org/10.1080/10556788.2020.1808977
19. He, C., Zhang, Y., Gong, D., Ji, X.: A review of surrogate-assisted evolutionary algorithms for expensive optimization problems. Expert Syst. Appl. 119495 (2023)
20. Jin, Y.: Surrogate-assisted evolutionary computation: recent advances and future challenges. Swarm Evol. Comput. **1**(2), 61–70 (2011)
21. Koziel, S., Bekasiewicz, A.: Scalability of surrogate-assisted multi-objective optimization of antenna structures exploiting variable-fidelity electromagnetic simulation models. Eng. Optim. **48**(10), 1778–1792 (2016)
22. Li, J., Wang, P., Dong, H., Shen, J.: Multi/many-objective evolutionary algorithm assisted by radial basis function models for expensive optimization. Appl. Soft Comput. **122**, 108798 (2022)
23. Li, K., Pan, L., Xue, W., Jiang, H., Mao, H.: Multi-objective optimization for energy performance improvement of residential buildings: a comparative study. Energies **10**(2), 245 (2017)
24. Lin, J., He, C., Cheng, R.: Adaptive dropout for high-dimensional expensive multiobjective optimization. Complex Intell. Syst. **8**(1), 271–285 (2022)
25. Pan, L., He, C., Tian, Y., Wang, H., Zhang, X., Jin, Y.: A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization. IEEE Trans. Evol. Comput. **23**(1), 74–88 (2018)

26. Rodriguez, C.J., de Boer, S.M., Bosman, P.A., Alderliesten, T.: Bi-objective optimization of organ properties for the simulation of intracavitary brachytherapy applicator placement in cervical cancer. In: Medical Imaging 2023: Image-Guided Procedures, Robotic Interventions, and Modeling, vol. 12466, pp. 114–125. SPIE (2023)
27. Rodriguez, C.J., Thomson, S.L., Alderliesten, T., Bosman, P.A.: Temporal true and surrogate fitness landscape analysis for expensive bi-objective optimisation. arXiv preprint arXiv:2404.06557 (2024)
28. Shi, R., Liu, L., Long, T., Wu, Y., Wang, G.G.: Multidisciplinary modeling and surrogate assisted optimization for satellite constellation systems. Struct. Multidiscip. Optim. **58**, 2173–2188 (2018)
29. Silva, R.C., Li, M., Rahman, T., Lowther, D.A.: Surrogate-based MOEA/D for electric motor design with scarce function evaluations. IEEE Trans. Magn. **53**(6), 1–4 (2017)
30. Su, S., Li, W., Li, Y., Garg, A., Gao, L., Zhou, Q.: Multi-objective design optimization of battery thermal management system for electric vehicles. Appl. Therm. Eng. **196**, 117235 (2021)
31. Taran, N., Ionel, D.M., Dorrell, D.G.: Two-level surrogate-assisted differential evolution multi-objective optimization of electric machines using 3-D FEA. IEEE Trans. Magn. **54**(11), 1–5 (2018)
32. Tian, Y., Cheng, R., Zhang, X., Jin, Y.: PlatEMO: a MATLAB platform for evolutionary multi-objective optimization. IEEE Comput. Intell. Mag. **12**(4), 73–87 (2017)
33. Tresidder, E., Zhang, Y., Forrester, A.: Acceleration of building design optimisation through the use of kriging surrogate models. Proc. Build. Simul. Optim. **2012**, 1–8 (2012)
34. Wang, N., Li, C., Li, W., Chen, X., Li, Y., Qi, D.: Heat dissipation optimization for a serpentine liquid cooling battery thermal management system: an application of surrogate assisted approach. J. Energy Storage **40**, 102771 (2021)
35. Wang, X., Jin, Y., Schmitt, S., Olhofer, M.: An adaptive Bayesian approach to surrogate-assisted evolutionary multi-objective optimization. Inf. Sci. **519**, 317–331 (2020)
36. Xu, W., Chong, A., Karaguzel, O.T., Lam, K.P.: Improving evolutionary algorithm performance for integer type multi-objective building system design optimization. Energy Build. **127**, 714–729 (2016)
37. Yu, M., Li, X., Liang, J.: A dynamic surrogate-assisted evolutionary algorithm framework for expensive structural optimization. Struct. Multidiscip. Optim. **61**(2), 711–729 (2020)
38. Zemella, G., De March, D., Borrotti, M., Poli, I.: Optimised design of energy efficient building façades via evolutionary neural networks. Energy Build. **43**(12), 3297–3302 (2011)
39. Zhang, Q., Liu, W., Tsang, E., Virginas, B.: Expensive multiobjective optimization by MOEA/D with gaussian process model. IEEE Trans. Evol. Comput. **14**(3), 456–474 (2009)
40. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., Tiwari, S., et al.: Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report, vol. 264, pp. 1–30 (2008)

41. Zhang, Z., Chen, H.C., Cheng, Q.S.: Surrogate-assisted quasi-newton enhanced global optimization of antennas based on a heuristic hypersphere sampling. IEEE Trans. Antennas Propag. **69**(5), 2993–2998 (2020)
42. Zhou, Z., Ong, Y.S., Nguyen, M.H., Lim, D.: A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In: 2005 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2832–2839. IEEE (2005)

An Adaptive Approach to Bayesian Optimization with Setup Switching Costs

Stefan Pricopie[1(✉)], Richard Allmendinger[1], Manuel López-Ibáñez[1], Clyde Fare[2], Matt Benatan[3], and Joshua Knowles[1]

[1] Alliance Manchester Business School, University of Manchester, Manchester, UK
stefan.pricopie@postgrad.manchester.ac.uk
[2] IBM Research, Daresbury, UK
[3] Sonos, London, UK

Abstract. Black-box optimization methods typically assume that evaluations of the black-box objective function are equally costly to evaluate. We investigate here a resource-constrained setting where changes to certain decision variables of the search space incur a higher switching cost, e.g., due to expensive changes to the experimental setup. In this scenario, there is a trade-off between fixing the values of those costly variables or accepting this additional cost to explore more of the search space. We adapt two process-constrained batch algorithms to this sequential problem formulation, and propose two new methods—one cost-aware and one cost-ignorant. We validate and compare the algorithms using a set of 7 scalable test functions with different switching-cost settings. Our proposed cost-aware parameter-free algorithm yields comparable results to tuned process-constrained algorithms in all settings we considered, suggesting some degree of robustness to varying landscape features and cost trade-offs. This method starts to outperform the other algorithms with increasing switching cost. Our work expands on other recent Bayesian Optimization studies in resource-constrained settings that consider a batch setting only. Although the contributions of this work are relevant to the general class of resource-constrained problems, they are particularly relevant to problems where adaptability to varying resource availability is of high importance.

Keywords: Bayesian Optimization · Switching costs · Expensive optimization

1 Introduction

In expensive black-box optimization problems, some decision variables may be more costly to change than others, leading to different evaluations that have different costs depending on which decision variables are changed. For example, in the automotive or electronics manufacturing industries, whenever a new product is needed, a retooling of the production line is implemented [25]. A line switch can delay production time due to an idle period and lead to additional

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 340–355, 2024.
https://doi.org/10.1007/978-3-031-70068-2_21

setup costs from any specialized labor employed for instrument recalibration, equipment cleaning, or replacement of consumables. A switching cost is encountered every time there is a change in the setup. In such settings, frequent setup changes are prohibitive as the cumulative switching cost between setups will rapidly increase, limiting the number of evaluations possible.

In several problems, we see that the history of the optimization process influences the cost and available resources. While in Bayesian Optimization (BO) it is typical to allow decision variables to be changed freely between iterations, this scenario does not fully capture the logistical constraints of real-world applications: the cost of changing any decision variable is not always the same.

It is expected that for any changes made throughout the optimization process, particularly in dynamic or time-dependent problems, some penalty is incurred, whether that is an increase in time, cost, resource use, or any other variables involved. Examples include heat treatment scheduling [26], robot design [19], molecular discovery [21] or path-based problems in environmental sciences [7]. In such problems, the history of the most recent evaluation(s) makes certain neighboring points cheaper to evaluate than others, thus making exploitation cheap and exploration expensive.

Similar scenarios are encountered in problems around chemical processing plants [3] and biopharmaceutical production [6] when there is a change in formula, in food processing facilities changing between recipes [15], or during large construction projects with multiple completion phases [1]. The same issues are also encountered in supply chain logistics when there is a need for a reconfiguration of the distribution network, a change of transport mode, or an inventory uncertainty driven by supply levels [5].

Our work was motivated by such restrictions in experiments involving physical resources that are usually limited and often influence the optimization cost and the overall process.

In many optimization problems, we can distinguish two types of costs:

(i) an evaluation cost, which can be fixed or variable depending on the problem formulation; and
(ii) a setup cost, which involves the preparation before the evaluation.

Although there has been substantial research on homogeneous and heterogeneous evaluation costs [17,22,26], there has been little research on setup costs and bottlenecks arising from performing physical evaluations. The main decision then becomes when and how often a switch to a cost-associated decision variable should happen given the optimization time, the number of resources used, and the improvement in the quality of the solution achieved.

An example of such resource-constrained BO was proposed by Vellanki et al. [26] in the form of process-constrained batch optimization. Their experimental setup imposes constraints on batch construction by only allowing batches that share the same values for some decision variables. In this paper, we are extending their formulation to a generalized form by transforming the batches into a sequential problem. Since we are looking at sequential rather than batch settings, the resource constraints are affected by previous evaluations throughout

the search space, which is not the case in the batch setting, where the constraint affects only the respective batch. We also soften their constraint by allowing switches with a penalty associated with them. The formulation of Vellanki et al. [26] can then be seen as a particular case of our problem where the switching cost is $+\infty$, even though the cost is not specifically modeled in their case.

The overarching research question addressed in this paper is whether there are good adaptive and/or parameter-free strategies that an optimizer can use to choose between evaluating cheaply (by reusing the same setup as before) and changing the setup altogether for a more expensive evaluation. Every change in a new setup will incur a switching cost, and we wish to be robust to different *relative* switching costs.

The main contributions of the paper are:

1. We extend the optimization problem proposed in [26] from a batch-constrained formulation to a sequential one to allow for broader generalizability. Our formulation is resource-aware by penalizing frequent changes performed in the costly dimensions and considers the setup cost for more thorough optimization.
2. We show how the optimization process is affected by both the setup and the evaluation cost. Ignoring either of the two can lead to suboptimal convergence with respect to the overall cost of optimization.
3. We adapt the two resource-unaware algorithms proposed by Vellanki et al. [26] for batch constraints to resource-aware sequential settings. Additionally, we propose two new algorithms for this problem, an Expected Improvement per Unit Cost (EIPU) and a Probabilistic Reuse BO.
4. We show that previously proposed solutions require restricting the optimization to several evaluations per setup. The choice of this number is nontrivial and depends on the setup cost and the dimensionality of the problem.

The rest of this work is organized as follows. In Sect. 2 we review the related work. The problem setting is formally introduced in Sect. 3. In Sect. 4 we describe the algorithms used, with the experimental setup given in Sect. 5 together with the results of the experiments. Finally, we discuss any limitations and directions for future research in Sect. 6.

2 Related Work

Our research builds on the academic literature of process-constrained BO, and more specifically on studies that consider cost, for example, per evaluation or setup, as part of their objective function. While the literature around BO is extensive [8,10,27], there is not an equal focus on those BO applications that use a non-fixed cost. The cost per evaluation has been a relatively more popular research problem, both in the multi-fidelity [14,23,24] and single-fidelity space [2,17,20]. With regard to incurring a switching cost for new setups, we have identified several articles that are directly related to our research [7,19,20,22,26].

Snoek et al. [22]'s seminal paper emphasized BO's potential on hyperparameter tuning in neural networks while being the first to bring to the attention the problem of cost in optimization. Whenever there is an expensive optimization problem, the assumption is usually that this cost is fixed. Nevertheless, the cost per evaluation in neural networks is proportional to the internal batch size of a neural network training or other hyperparameters such as learning epochs. Snoek et al. [22] introduced the *EI per second*, later referred to as *EI per unit cost* (EIPU) [18], aimed at addressing the heterogeneity of training costs. However, this does not address the setup costs that arise from any changes in the model architecture such as the number of layers of the model or loading and preprocessing of new data, which require a full reinitialization of the model.

Liao et al. [19] extends Vellanki et al. [26]'s batch construction, where batches need not be constrained to a single setup and can be parallelized. In this case multiple batches are running in parallel, which constrains the optimization process based on the setup cost of n number of machines (robots in this case) for a while. Since morphologies are expensive and take a lot of time to produce, the goal of the paper is to evaluate multiple morphologies (microrobots) in parallel and optimize their unconstrained decision parameters (here controllers) while minimizing cost (how many morphologies are produced). Similarly to Vellanki et al. [26], their algorithms are optimized against the number of new setups and do not account for evaluation costs, which means that the only difference between a higher batch and a lower batch is computational cost.

To our knowledge, Lin et al. [20] are the first to introduce the idea of switching costs in the context of BO in a continuous space. They partition the search space into disjoint tiles. The cost of the changes is then measured when the optimization process jumps from one tile to another. Raising similar issues to this problem is also the work presented by Folch et al. [7]. While both papers discuss the scenario where the cost of optimization arises between evaluations, Folch et al. [7] considers the case where the cost is the distance between evaluations, thus making (proximal) exploitation cheaper and (distal) exploration more expensive.

Our problem differs from previous research in a couple of ways. Firstly, we are considering a sequential setting rather than a batch one. We motivate this choice because not all optimization processes allow for parallel batch evaluations. Moreover, by evaluating sequentially, we can see the effect of reusing the same setup on performance at a more granular level as algorithms have greater control over when to change or switch a setup. This also addresses the issue of choosing an appropriate batch size for the experiments, either fixed or adaptive, without a priori knowledge. Secondly, we soften the hard constraint used in the previous papers, meaning that one can evaluate outside of the search space of the current setup by switching it, but at the cost of a penalty, i.e., a switching cost.

3 Problem Setup

This paper considers expensive optimization problems on $f\colon \mathcal{X} \to \mathbb{R}$ where the domain $\mathcal{X} \subseteq \mathbb{R}^d$ can be broken down into costly and cheap dimensions. More

formally, $\mathcal{X} = \mathcal{X}_{\text{cheap}} \times \mathcal{X}_{\text{costly}}$ is the Cartesian product of $\mathcal{X}_{\text{cheap}} \subseteq \mathbb{R}^{d_{\text{cheap}}}$ and $\mathcal{X}_{\text{costly}} \subseteq \mathbb{R}^{d_{\text{costly}}}$ with $d = d_{\text{cheap}} + d_{\text{costly}}$ being the dimensions of $\mathcal{X}$, $\mathcal{X}_{\text{cheap}}$ and $\mathcal{X}_{\text{costly}}$ respectively. We notate $\mathbf{x} = \left[\, \mathbf{x}_{\text{cheap}} \;\; \mathbf{x}_{\text{costly}} \,\right]$ where $\mathbf{x} \in \mathcal{X}$, $\mathbf{x}_{\text{cheap}} \in \mathcal{X}_{\text{cheap}}$ and $\mathbf{x}_{\text{costly}} \in \mathcal{X}_{\text{costly}}$ to differentiate between the cheap and costly spaces. We then model the cost function $c\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as:

$$c(\mathbf{x}^t, \mathbf{x}^{t-1}) = \begin{cases} c_{\text{switch}} & \text{if } \mathbf{x}^t_{\text{costly}} \neq \mathbf{x}^{t-1}_{\text{costly}}, \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

where $\mathbf{x}^t = \left[\, \mathbf{x}^t_{\text{cheap}} \;\; \mathbf{x}^t_{\text{costly}} \,\right]$ is the t^{th} function evaluation and $c_{\text{switch}} \in [1, \infty)$. We refer to a setup change or a switch when $\mathbf{x}^t_{\text{costly}} \neq \mathbf{x}^{t-1}_{\text{costly}}$. For example, when $c_{\text{switch}} = 5$, it is 5 times more expensive to switch than to reuse the same setup. When $c_{\text{switch}} = 1$, it is equivalent to the traditional optimization problem. Intuitively, c_{switch} is the penalty multiplier to change the setup for a new experiment.

In the example shown in Fig. 1 the black dot is the last evaluation and the square is the best next point to evaluate, which is on the line, meaning that the same setup applies and there is no new cost associated with the move. However, evaluating at the red square, which is the best next point to evaluate, is within a different setup, hence the function evaluation will have a new setup cost.

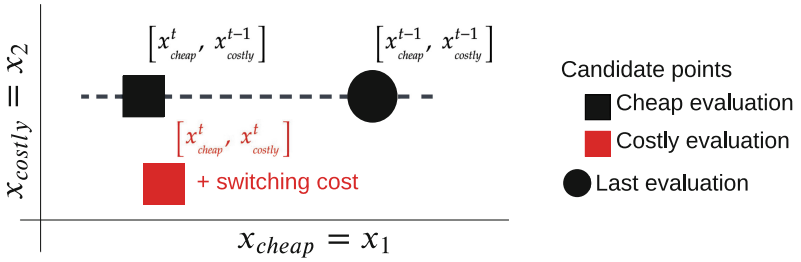


Fig. 1. Example of a $d = 2$ problem with $d_{\text{costly}} = 1$. The x-axis and y-axis represent the cheap and costly dimensions, respectively. The dotted line is the set of evaluations that use the current setup. The squares are the candidate points and the dot is the last evaluation.

In Fig. 2 we use a two-dimensional synthetic test function, $\mathcal{X} = [-4, 8] \times [-4, 8] \subset \mathbb{R}^2$ with $\mathcal{X}_{\text{costly}} = \{x_1 \in [-4, 8]\}$ and $\mathcal{X}_{\text{cheap}} = \{x_2 \in [-4, 8]\}$, to show the behavior of a solution method in the cases of setup switching costs. Depending on the next chosen point to evaluate, the algorithm can select a point which incurs an evaluation cost. For example, in evaluation policy 1, there is a switching cost associated to every move made, while in the evaluation policy 2, there are 3 moves that do not have a switching cost associated since the same setup is reused. This type of scenario then raises the question of whether the algorithm can find an optimal trade-off between the costs associated with a switch and the quality of the solution found.

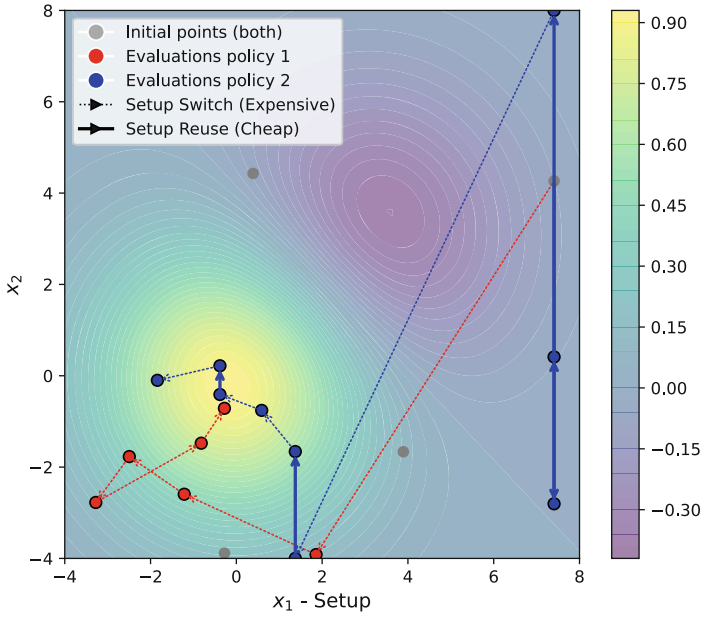


Fig. 2. Maximization problem on synthetic function with axes x_1 and x_2 representing the expensive and cheap dimensions in the search space respectively. The grey dots represent the initial starting points, which are randomly selected. The red and blue lines then denote two evaluation strategies. The red line represents a strategy where the algorithm changes the setup at each evaluation, while the strategy represented by the blue line uses the same setup for evaluations 2, 3, 5, and 8. The second evaluation policy performs 9 evaluations, while the first policy performs only 6 due to its costly evaluations. Reusing the same setup allows for more total evaluations at the expense of fewer evaluations performed across the costly dimension x_1. (Color figure online)

4 Methodology

We are interested in characterizing the trade-off between evaluating the cheap dimensions for more evaluations and evaluating the costly dimensions for less. We investigate whether it is possible to find an optimal p^* value (or range), where p is the probability of re-using a setup and hence evaluating a new solution that keeps the values of the costly dimensions without incurring a switching cost. We analyze four algorithms capable of tackling this problem, in addition to a classic BO that ignores the switching cost.

The first new algorithm that we propose is *pReuseBO*. This algorithm keeps the value of the expensive decision variables with probability $p \in [0, 1]$ independently at every time step. In Fig. 2 we illustrate one run of this algorithm with $p = 0.1$, which on average would lead to 1 out of 10 points being cheaply evaluated, and incur a switching cost for the remaining 9.

The difference in our problem to previous research (e.g., [26]) is that instead of batches, we treat the problem sequentially. Once we choose an optimal point

Algorithm 1. Initialization for Periodic Switching BO Nested

1: **Input:** f, $\mathcal{X} = \mathcal{X}_{\text{cheap}} \times \mathcal{X}_{\text{costly}}$, n initial "costly" points, periodicity length k
2: Select unique "costly" points $\{\mathbf{x}^1_{\text{costly}}, \ldots, \mathbf{x}^n_{\text{costly}}\}$, each to be used k times
3: Initialize a set of $n \cdot k$ unique "cheap" points $\{\mathbf{x}^t_{\text{cheap}}\}_{t=1}^{n \cdot k}$
4: Define the initial points $\mathbf{x}^t = \left[\mathbf{x}^t_{\text{cheap}}\ \mathbf{x}^{\lceil t/k \rceil}_{\text{costly}}\right]$ for $t = 1, \ldots, n \cdot k$
5: Evaluate f at each $\mathbf{x}^t$ to get $y^t = f(\mathbf{x}^t)$ for $t = 1, \ldots, n \cdot k$
6: Form the complete dataset $D = \{(\mathbf{x}^t, y^t)\}_{t=1}^{n \cdot k}$
7: **Return** D

to evaluate by looking at the entire search space, we automatically evaluate it without creating a batch with variations. Since we evaluate a single point, we constraint the costly dimensions for $k-1$ number of evaluations, that is, the setup is changed only after every k evaluations. Here, k represents the periodicity to which an event happens and is graphically presented in Fig. 3. In other words, the algorithm keeps the same setup for $k-1$ steps out of k. This is different to batching because we refit the Gaussian Process (GP) after each evaluation. One possible approach is to maintain the value of the costly dimensions with probability $1-p$ or change them with probability p (and pay the switch cost). Another possible approach is to maintain the value of the costly dimensions for k evaluations, then change them (and pay the switch cost) and perform another k evaluations only changing the cheap dimensions. The parameter k is a simpler planning strategy than the parameter p. We expect that setting $k = \frac{1}{1-p}$ will produce a similar behavior on average.

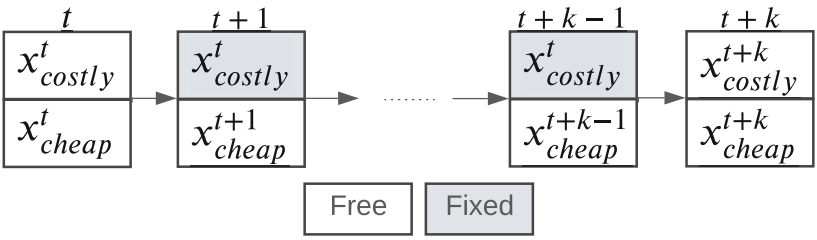


Fig. 3. Periodic switching of k. The costly decision variables are changed only every k evaluations.

We adapted two algorithms proposed by Vellanki et al. [26] to our problem. The first algorithm is a sequential version of the pc-BO-basic, here referred to as Periodic Switching BO (PSBO). In the original algorithm (pc-BO-basic), a batch would be constructed by choosing the first point freely as the best found by the chosen acquisition function, then iteratively adding optimal points constrained to the costly decision variables of the first point until the batch is complete. Here in PSBO, we simply allow changes to $\mathbf{x}_{\text{costly}}$ after each k evaluations. We also substituted their chosen acquisition functions UCB and PSD (also known as Pure Exploration) for EI as the former functions can be too exploratory [4].

Algorithm 2. Periodic Switching BO Nested

1: **Input:** Initial dataset D, $t = |D|$ ▷ From Algorithm 1
2: **while** stopping criteria not met **do**
3: **if** $t \bmod k = 0$ **then**
4: $D_{\text{costly}} = \left\{ \left(\mathbf{x}^i_{\text{costly}}, \max \left\{ y \mid \left([\mathbf{x}_{\text{cheap}}, \mathbf{x}^i_{\text{costly}}], y \right) \in D \right\} \right) \mid i = 1, \ldots, t/k \right\}$
5: Train GP_{costly} using the dataset D_{costly}
6: $\mathbf{x}_{\text{costly}} = \underset{\mathbf{x} \in \mathcal{X}_{\text{costly}}}{\arg\max}\, \alpha_{\text{costly}}(\mathbf{x}; D_{\text{costly}})$ ▷ Choose new costly variables
7: **end if**
8: Train GP using the dataset D
9: $\mathbf{x}^{t+1}_{\text{cheap}} = \underset{\mathbf{x} \in \mathcal{X}_{\text{cheap}}}{\arg\max}\, \alpha(\mathbf{x}; D \mid \mathbf{x}_{\text{costly}})$ ▷ Condition on last costly variables
10: Evaluate $y_{t+1} = f([\mathbf{x}^{t+1}_{\text{cheap}}, \mathbf{x}_{\text{costly}}])$
11: Update D with $([\mathbf{x}^{t+1}_{\text{cheap}}, \mathbf{x}_{\text{costly}}], y_{t+1})$
12: $t = t + 1$ ▷ Increment iteration counter
13: **end while**

The second algorithm is PSBO Nested, shown in Algorithm 2, which is adapted from the original pc-BO-nested in Vellanki et al. [26] designed for batch into sequential. The nested approach uses two separate BO algorithms, one for the costly space only and one for the entire space, running concomitantly. This process also requires a special initialization presented in Algorithm 1. The key difference between the two algorithms is in the retraining of the model after each step (see lines 8–12) rather than in batch and in the conditional retraining of costly GP followed by changing the costly decision variables after each k steps (see line 3). And, like in PSBO, we used EI for consistency.

We also propose an adaptation of the EI per unit cost (EIPU) to the pathological case where the cost function is discontinuous. As we are incorporating cost, and since we want to compare the penalty incurred while evaluating outside of the initial setup, we want to adapt the EIPU. Unlike previous methods used for this problem type, this is the first algorithm that is cost-aware. In a traditional sense, EI is a continuous function, which matches the general formulation of cost in the literature as continuous. However, in our case, cost is a discontinuous function (Eq. 1). To address this we perform two acquisition optimizations within the same iteration. This will produce two candidate points for the next evaluation: (i) a candidate point that changes a costly dimension and hence incurs a switching cost and (ii) a candidate point that only changes cheap dimensions and hence without having a switching cost. The algorithm will then discount the EI of the two points by their respective cost and the best point is chosen. When we discount the cost we are considering the cost cooling strategy EI-cool$(\mathbf{x}) := \frac{EI(\mathbf{x})}{c(\mathbf{x})^\gamma}$, where γ is the decay parameter where $\gamma = (B - B_t)/B$, t is the iteration number, B is the total budget, and B_t is the budget used up to time step t [18]. This cost cooling strategy lessens the impact of the cost model with the advance of iterations, allowing for more costly evaluations to escape local optimum.

5 Experiments

We run two sets of experiments: (i) finding the best p value across the different dimensions and switching costs using *pReuseBO* (Subsect. 5.2), and (ii) comparing the four algorithms' performance (Subsect. 5.3). The experimental setup corresponding only to a particular experiment will be explained subsequently in each respective subsection.

Table 1. Test functions considered in this study and their settings. For each function we considered the dimensions $d \in \{2, 3, 4\}$. For Ackley, Griwewank, and Salomon, their original domain is cropped as shown in the table, such that the optimal point was not in the centre of the domain.

Functions	Symmetrical	Domain per dimension d
Ackley	Yes	$[-15, 30]$
Griewank	Yes	$[-300, 600]$
Levy	No	$[-10, 10]$
Michalewicz	No	$[0, \pi]$
Rosenbrock	No	$[-5, 10]$
Salomon	Yes	$[-50, 100]$
Schwefel	Yes	$[-500, 500]$

5.1 Experimental Setup

Both sets of experiments are run on scalable test problems, shown in Table 1, of dimension $d \in \{2, 3, 4\}$. We choose to show the performance of the algorithms over the above test functions because the dimensionality of the problem has a direct impact on the algorithm behavior and the optimal hyperparameters. This argument will be empirically validated in the following sections. Many scalable functions are symmetrical, but we include non-symmetrical test functions as they better capture the complexities of real-world applications.

The following configurations are considered:

2D: 1 cheap & 1 costly;
3D: 1 cheap & 2 costly *OR* 2 cheap & 1 costly;
4D: 1 cheap & 3 costly *OR* 2 cheap & 2 costly *OR* 3 cheap & 1 costly;

The costly dimensions are randomly selected across 20 independent runs for robustness until the termination criteria is met. Randomizing the costly dimensions helps control asymmetries in the impact of dimensionalities for the non-symmetrical test functions, which is suitable for the generability of the paper, as opposed to arbitrarily fixing some dimensions across all runs. The values of p that we chose to test are between $[0, 1]$ inclusive with a step of 0.05 leading to

21 possible values. The value of the switching cost is reflective of how expensive an evaluation becomes when switching the setup relative to the existing one at the moment of evaluation. For example, a switching cost of 2 means that changing the setup would make the next evaluation twice as expensive as it would be by maintaining the same setup. In this paper, we set the possible values for the switching cost at $\{1, 2, 4, 8, 16\}$ for the first set of experiments to compare against a classical optimization problem, and $\{2, 4, 8, 16, 32\}$ for the second set of experiments to analyze the effect of higher costs.

Our budget is referenced in terms of cost because we want to be able to benchmark cost-efficient algorithms with classical BO. All experiments have a low budget of $N = 10d$ units of cost, where d is the dimension of the problem, once the initial random sample has been evaluated. Therefore, algorithms can perform between N evaluations at the lower end, when switching at each iteration which is the case for traditional BO, and $N \cdot c_{\text{switch}}$ evaluations at the higher end when no switching is performed and the algorithm reuses only the last setup from the initialization, such as when $p = 1$ for *pReuseBO*.

All the BO algorithms ran in this paper use GP surrogates with Matérn 5/2 kernel and Automatic Relevance Determination (ARD) as recommended by Snoek et al. [22]. We optimise the EI acquisition function with L-BFGS-B with 10 restarts and 2048 raw samples. For EIPU we optimise the acquisition function twice to get the two candidate points.

To evaluate the performance of the algorithms, we use the GAP measure, which is defined as $\text{GAP} = (y_i - y_0)/(y^* - y_0)$, assuming maximization of the objective function f without loss of generality, where y_i is the maximum observed objective function value within a single run, y^* is the true optimal (maximum) value of f, and y_0 is the objective function value of the initial starting point, which is the same for all algorithms solving the same problem [11–13,16]. GAP should be maximized and can be thought of as the inverse of the normalized regret and is preferred as it allows for generalizability in algorithm performance across multiple test functions.

5.2 Optimal Probability Value

We analyze the performance of different fixing probabilities p to maximize the value of the objective function in a set budget. Figure 4 shows that an increase in the switching cost leads to an increase in p^*. A higher switching cost means a greater trade-off between exploring the costly dimension and performing more total evaluations. As p increases, the algorithm performance diverges. High p values corresponding to frequent reusability perform better in high-cost settings but hinder performance in low-cost settings.

The optimal p^* decreases with an increase in the costly dimensionality (Fig. 4). The result is expected as the importance of exploring the costly dimension becomes higher when the number of such dimensions increases. The gain from performing more frequent switches in this scenario is greater, while the loss incurred from not exploring the lower number of cheap dimensions is low.

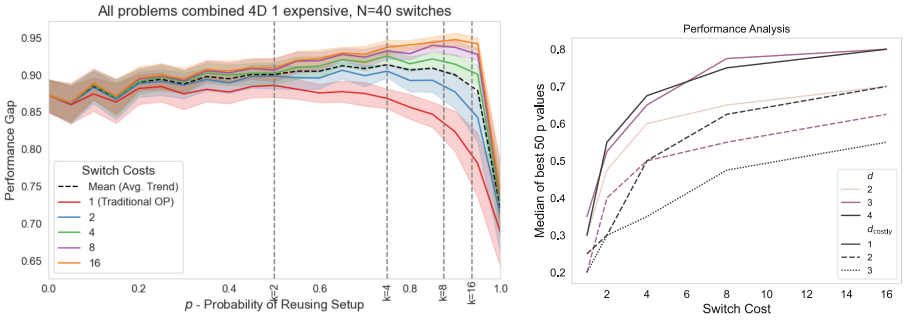


Fig. 4. Left: The mean GAP performance (and 95% CI) of the method *pReuseBO* plotted against 21 values of p, where $GAP = 1$ means that the optimal solution was found, for 20 independent runs and averaged over the 7 test functions with 4 dimensions and 1 costly dimension. The approximately equivalent k for Periodic Switching is shown on the x-axis. The dotted line shows the effect of p averaged over all the switching costs to highlight its steady increase in performance followed by its sharp decline. **Right:** The median of the highest performing p parameters for each problem and costly dimensionality setting plotted against switch cost. p is much more sensitive to tune with respect to different values of switch cost in the problems with high dimensionality and low costly dimensionality.

The lower the value of p the closer the behavior of the algorithm is to the standard BO. The performance of p in the context of switching cost is expected to decrease quickly since there is no trade-off, i.e., one evaluation on the costly dimension is changed for one evaluation on the cheaper dimension. This can be a reasonable trade-off for smaller values of p, although it is unlikely to produce a significant effect on performance. However, with increasing p, there will be a steady decrease in returns and the importance of exploring the costly dimension becomes higher. On the other hand, an increase in the switching cost results in a higher trade-off for exchanging one expensive evaluation for multiple cheap ones.

5.3 Algorithms Performance

The algorithms performance is shown in Table 2. The experiments shown in this table are from the 4D-1 setting. We have chosen this configuration based on the results obtained over the p analysis (see Fig. 4). These have shown that the optimal choice of reusing or switching a setup is most sensitive in high-dimensional, low expensive dimensionality settings.

We are most interested in comparing the parameter-free EI per unit cost with multiple instances of the other algorithms. Whenever there is a reference to "best" of an algorithm, we mean the manually tuned configuration of the algorithm that led to the best performance in terms of GAP. The findings show that despite using tuned versions of the other algorithms, these do not outperform the results obtained by the EIPU in high switching costs scenarios. In low cost settings EIPU behaves close to BO and is outperformed by *pReuseBO* and PSBO.

Table 2. Algorithm comparison for the 7 test functions, over the 5 algorithms of interest. (Tuned) *pReuseBO*, (Tuned) PSBO, and (Tuned) PSBO Nested are the configurations who yielded the best performance in terms of GAP over 5 tested configurations for parameter tuning. The algorithm with the highest GAP is bolded the second best is highlighted in *light grey*.

Switch Cost	Problem	BO	*pReuseBO* (Tuned)	EIPU	PSBO (Tuned)	PSBO Nested (Tuned)
2	ackley	0.886344	**0.920431**	0.903437	0.877309	0.760876
2	griewank	0.991388	0.992109	0.991953	**0.992834**	0.976825
2	levy	0.993641	0.994666	0.995413	**0.996521**	0.989614
2	michalewicz	0.855549	**0.921913**	0.893882	0.904125	0.741281
2	rosenbrock	0.999413	**0.999773**	0.999623	0.999728	0.999205
2	salomon	0.868022	0.894309	0.869581	**0.898732**	0.837132
2	schwefel	0.714171	0.788107	**0.789711**	0.784081	0.704648
4	ackley	0.886344	**0.928764**	0.906928	0.899564	0.773149
4	griewank	0.991388	0.992437	0.992114	**0.993251**	0.980942
4	levy	0.993641	0.995714	**0.996914**	0.996546	0.990014
4	michalewicz	0.855549	**0.950820**	0.934351	0.927506	0.760070
4	rosenbrock	0.999413	**0.999957**	0.999950	0.999934	0.999686
4	salomon	0.868022	**0.917669**	0.906176	0.913210	0.877922
4	schwefel	0.714171	**0.818559**	0.814713	0.815333	0.726128
8	ackley	0.886344	**0.931443**	0.929409	0.907566	0.773149
8	griewank	0.991388	0.992783	0.991681	**0.994011**	0.981282
8	levy	0.993641	**0.997101**	0.996722	0.996798	0.991616
8	michalewicz	0.855549	0.960074	**0.983750**	0.943899	0.760070
8	rosenbrock	0.999413	0.999969	**0.999992**	0.999973	0.999892
8	salomon	0.868022	**0.925928**	0.919973	0.916355	0.878876
8	schwefel	0.714171	0.822432	**0.845480**	0.823618	0.762085
16	ackley	0.886344	0.931443	**0.932147**	0.907566	0.773149
16	griewank	0.991388	0.993453	0.992587	**0.994011**	0.981353
16	levy	0.993641	0.997101	**0.998516**	0.997608	0.993175
16	michalewicz	0.855549	0.970236	**0.987241**	0.977924	0.764227
16	rosenbrock	0.999413	0.999987	**0.999999**	0.999992	0.999922
16	salomon	0.868022	**0.937830**	0.910404	0.924830	0.905518
16	schwefel	0.714171	0.840700	**0.867731**	0.827956	0.772887
32	ackley	0.886344	**0.931443**	0.921927	0.907566	0.773149
32	griewank	0.991388	0.993453	0.992818	**0.994011**	0.987995
32	levy	0.993641	0.997158	**0.998989**	0.997608	0.993679
32	michalewicz	0.855549	0.972642	**0.986060**	0.980103	0.764227
32	rosenbrock	0.999413	0.999993	**0.999999**	0.999992	0.999961
32	salomon	0.868022	**0.945250**	0.941213	0.927371	0.905611
32	schwefel	0.714171	0.842070	**0.902302**	0.827956	0.774452

In Table 2 we highlighted the first two algorithms with the highest GAP achieved. In the case of switching cost 2, EIPU is outperformed for all test functions by BO Random and PSBO. However, its performance increase in comparison to the other algorithms is evident starting with a switching cost of 4. The EIPU outperforms the other algorithms starting with a switching cost of 8 for the Schwefel test function. Nonetheless, for a switch cost of 16, it performs best in 4 out of the 7 test function and second best in 1 of them. Similarly, for the highest switching cost of 32, it performs best in 4 out of 7 cases and second best in 2 of them.

Since *pReuseBO* and PSBO are manually tuned, for matter of reference, we have also compared the results obtained from their untuned equivalent. Without configuring the parameters of the other two methods, EIPU outperforms the untuned variants in almost all cases with a switching cost higher than 2. Generally, as switching cost increases, EIPU performs better as it is adaptable, exploiting good setups more. Compared to the other algorithms, the cost-aware EI switches a setup only when the current one does not yield large enough improvements. The other algorithms switch following a predetermined sequence by p or k.

We observe that in a cost-aware sequential setting, PSBO does perform better than PSBO nested in any of our test functions. This is in contrast to the findings obtained by Vellanki et al. [26] who have shown that PSBO nested is the better algorithm in batch settings. The difference in performance is caused by the fact that the GP_{costly} (line 5 of Algorithm 2), which determines the values in the expensive dimensions, does not train well long term over the best found points for each setup. Therefore the algorithm will progress for a while but then plateau since the dimensionality reduction will prohibit the algorithm from finding the best point. Our results seem more consistent with the results from Ghadimi and Wang [9], which look at bi-level optimization methods where there is an inner and an outer loop problem. The authors highlight that a large number of iterations is required for nested algorithms to perform well.

6 Conclusion

In this paper, we have looked at a generalizable version of cost-aware problems in light of the limitations encountered in real-world experiments requiring physical resources. We have analyzed the behavior of multiple algorithms when a setup switch is associated with a cost and shown that the choice of switching throughout an optimisation process is non-trivial. This number is primarily influenced by the total dimensionality of the problem in relation to the number of costly dimensions, and the cost associated with each switch.

We have adapted a number of algorithms that can be used to solve the above class of problems. The parameter-free EIPU algorithm proposed as one of the solution methods was shown to yield comparable results with other finely tuned algorithms used in process-constrained BO problems. The practical impact of these algorithms can help reduce the cost of optimization processes in several fields, from automotive to biopharmaceutical settings.

A potential extension of the work presented in this paper comes from the inherent characteristic of BO being only interested in finding the next best point to evaluate. In our case, before evaluating the next point we want to know both how good that point is and its evaluation cost. In other words, analyzing an evaluation point is done through an aggregate between cost and value. It thus becomes important for BO to understand the overall quality of a certain setup before committing to it. A potential solution worth pursuing is the analysis of the performance lookahead-based heuristics, which take into consideration how good a setup might be and design BO algorithms that are less greedy.

References

1. Aslam, M., Baffoe-Twum, E., Saleem, F.: Design changes in construction projects – causes and impact on the cost. Civil Eng. J. **5**(7), 1647–1655 (2019). https://doi.org/10.28991/cej-2019-03091360. ISSN 2476-3055
2. Astudillo, R., Jiang, D., Balandat, M., Bakshy, E., Frazier, P.: Multi-step budgeted bayesian optimization with unknown evaluation costs. In: Advances in Neural Information Processing Systems, vol. 34, pp. 20197–20209. Curran Associates, Inc. (2021)
3. Behr, A., et al.: New developments in chemical engineering for the production of drug substances. Eng. Life Sci. **4**(1), 15–24 (2004). https://doi.org/10.1002/elsc.200406127. ISSN 1618-0240, 1618-2863
4. De Ath, G., Everson, R.M., Rahat, A.A.M., Fieldsend, J.E.: Greed is good: exploration and exploitation trade-offs in Bayesian optimisation. ACM Trans. Evol. Learn. Optim. **1**(1), 1–22 (2021). https://doi.org/10.1145/3425501. ISSN 2688-299X, 2688-3007
5. Dev, N.K., Shankar, R., Gunasekaran, A., Thakur, L.S.: A hybrid adaptive decision system for supply chain reconfiguration. Int. J. Prod. Res. **54**(23), 7100–7114 (2016). https://doi.org/10.1080/00207543.2015.1134842. ISSN 0020-7543, 1366-588X
6. Eberle, L.G., Sugiyama, H., Schmidt, R.: Improving lead time of pharmaceutical production processes using Monte Carlo simulation. Comput. Chem. Eng. **68**, 255–263 (2014). https://doi.org/10.1016/j.compchemeng.2014.05.017. ISSN 00981354
7. Folch, J.P., et al.: SnAKe: Bayesian Optimization with Pathwise Exploration (2022). arXiv:2202.00060. https://doi.org/10.48550/arXiv.2202.00060
8. Frazier, P.I., Wang, J.: Bayesian optimization for materials design. In: Lookman, T., Alexander, F.J., Rajan, K. (eds.) Information Science for Materials Discovery and Design. SSMS, vol. 225, pp. 45–75. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-23871-5_3
9. Ghadimi, S., Wang, M.: Approximation Methods for Bilevel Programming (2018). arXiv:1802.02246
10. Griffiths, R.R., Hernández-Lobato, J.M.: Constrained Bayesian optimization for automatic chemical design using variational autoencoders. Chem. Sci. **11**(2), 577–586 (2020). https://doi.org/10.1039/C9SC04026A. ISSN 2041-6520, 2041-6539
11. Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models. J. Glob. Optim. **34**(3), 441–466 (2006). https://doi.org/10.1007/s10898-005-2454-3. ISSN 1573-2916

12. Jiang, S., Chai, H., Gonzalez, J., Garnett, R.: BINOCULARS for Efficient, Nonmyopic Sequential Experimental Design (2019). https://doi.org/10.48550/arXiv.1909.04568. https://github.com/shalijiang/bo
13. Jiang, S., Malkomes, G., Converse, G., Shofner, A., Moseley, B., Garnett, R.: Efficient nonmyopic active search. In: Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Proceedings of Machine Learning Research, vol. 70, pp. 1714–1723 (2017)
14. Kandasamy, K., Dasarathy, G., Oliva, J., Schneider, J., Póczos, B.: Gaussian process bandit optimisation with multi-fidelity evaluations. In: Advances in Neural Information Processing Systems (NeurIPS 2016), vol. 29, pp. 992–1000 (2016)
15. Kopanos, G.M., Puigjaner, L., Georgiadis, M.C.: Resource-constrained production planning in semicontinuous food industries. Comput. Chem. Eng. **35**(12), 2929–2944 (2011). https://doi.org/10.1016/j.compchemeng.2011.04.012. ISSN 00981354
16. Lam, R., Willcox, K., Wolpert, D.H.: Bayesian optimization with a finite budget: an approximate dynamic programming approach. In: Advances in Neural Information Processing Systems, vol. 29. Curran Associates, Inc. (2016)
17. Lee, E.H., Eriksson, D., Perrone, V., Seeger, M.: A nonmyopic approach to cost-constrained Bayesian optimization. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence, pp. 569–577 (2021)
18. Lee, E.H., Perrone, V., Archambeau, C., Seeger, M.: Cost-aware Bayesian Optimization (2020). arXiv:2003.10870. https://doi.org/10.48550/arXiv.2003.10870
19. Liao, T., et al.: Data-efficient learning of morphology and controller for a microrobot. In: 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, pp. 2488–2494. IEEE (2019). https://doi.org/10.1109/ICRA.2019.8793802. ISBN 978-1-5386-6027-0
20. Lin, C.H., Miano, J.D., Dyer, E.L.: Bayesian optimization for modular black-box systems with switching costs. In: Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, pp. 1024–1034. PMLR (2021). ISSN: 2640-3498
21. Pricopie, S., et al.: Expensive optimization with production-graph resource constraints: a first look at a new problem class. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 840–848. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3512290.3528741. ISBN 978-1-4503-9237-2
22. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, Advances in Neural Information Processing Systems, vol. 4, pp. 2951–2959. Morgan Kaufmann Publishers, Inc., Lake Tahoe, USA (2012)
23. Song, J., Chen, Y., Yue, Y.: A general framework for multi-fidelity Bayesian optimization with gaussian processes. In: Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, pp. 3158–3167. PMLR (2019). ISSN: 2640-3498
24. Swersky, K., Snoek, J., Adams, R.P.: Multi-task Bayesian optimization. In: Advances in Neural Information Processing Systems, vol. 26. Curran Associates, Inc. (2013)
25. Teunter, R., Kaparis, K., Tang, O.: Multi-product economic lot scheduling problem with separate production lines for manufacturing and remanufacturing. Eur. J. Oper. Res. **191**(3), 1241–1253 (2008). https://doi.org/10.1016/j.ejor.2007.08.003. ISSN 03772217

26. Vellanki, P., et al.: Process-constrained batch Bayesian optimisation. In: Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
27. Zhang, Y., Apley, D.W., Chen, W.: Bayesian optimization for materials design with mixed quantitative and qualitative variables. Sci. Rep. **10**(1), 4924 (2020). https://doi.org/10.1038/s41598-020-60652-9. ISSN 2045-2322

Re-examining Supervised Dimension Reduction for High-Dimensional Bayesian Optimization

Quanlin Chen[1], Jing Huo[1], Yiyu Chen[1], Tianyu Ding[2], Yang Gao[1(✉)], Dong Li[3], and Xu He[3]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
quanlinchen@smail.nju.edu.cn, {huojing,gaoy}@nju.edu.cn, yiyuiii@foxmail.com

[2] Applied Sciences Group, Microsoft, Redmond, WA 98034, USA
tianyuding@microsoft.com

[3] Noah's Ark Laboratory, Huawei 518129, Shenzhen, China
{lidong106,hexu27}@huawei.com

Abstract. Bayesian optimization (BO) has been broadly applied to optimize expensive-to-evaluate black-box functions, but it is still challenging to scale BO to high dimensions while retaining sample efficiency. A solution in the existing literature is to assume that there exists a lower-dimensional structure for objective functions and learn the lower-dimensional embedding via supervised dimension reduction. For example, BO based on Sliced Inverse Regression (SIR) directly uses SIR to discover the intrinsic lower-dimensional structure of the objective function. However, the assumption of SIR leads to a mismatch in BO, and maximizing a high-dimensional acquisition function also leads to its poor performance. To reduce the mismatch between dimension reduction methods and BO, we introduce Kernel Dimension Reduction (KDR) and manifold KDR to BO. Furthermore, to improve the performance of acquisition functions, we construct a constrained low-dimensional acquisition function, where the constraint is constructed by the inverse mapping from the central subspace back to the original space using a batch of Gaussian Process models. We verify empirically that tackling these two issues improves the performance of methods based on supervised dimension reduction on a wide range of problems.

Keywords: Bayesian optimization · High-dimensional Bayesian optimization · Black-box optimization · Dimensionality reduction

1 Introduction

Many science and engineering tasks can be abstracted as black-box optimization. Moreover, their objective functions are expensive to evaluate. Bayesian

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 356–373, 2024.
https://doi.org/10.1007/978-3-031-70068-2_22

optimization (BO) has been broadly applied to optimize these expensive-to-evaluate black-box functions, because of its sampling efficiency, such as neural architecture search [33], hyperparameters tuning [13], vehicle design [2], and so on. The sampling efficiency of BO results from its surrogate models and acquisition functions, where surrogate models can estimate the objective function more cheaply and acquisition functions (AF) can select candidate points by balancing exploration and exploitation.

Although BO has achieved good performance in these applications with low dimensions, it is still challenging to scale BO to high dimensions due to the curse of dimensionality [29]. Specifically, the accuracy of surrogate models decreases severely in high-dimensional space, and the computational complexity of maximizing acquisition functions is exponential in dimensions [17]. Hence, most successful applications of BO are in D ($D \leq 20$) dimensions [8].

There are some methods proposed to extend BO to high-dimensional spaces. Most of the existing works make additional assumptions about the objective function. The first assumption is that there exists a lower-dimensional structure for objective functions, and then the low-dimensional embedding is learned [20,44]. The second assumption is that there exists an additive structure for objective functions, and then the additive structures are learned [17]. However, these assumptions are too restrictive for these methods to be widely used in real applications. What's more, several works impose no additional assumptions but directly improve the high-dimensional BO [7,27,30]. However, these methods cannot be scaled to too high dimensions. Hence, it is still an open problem to scale BO to high-dimensional spaces.

In this paper, we study the methods based on the lower-dimensional assumption, and in particular, we re-examine prior works utilizing supervised dimension reduction methods for learning low-dimensional embeddings (SIR-BO [44]). The sliced inverse regression (SIR) is attractive because it could automatically learn the low-dimensional embedding given a moderate amount of data. However, SIR assumes that the input covariate X has an elliptical distribution, which leads to a mismatch of BO. Another limitation of SIR-BO is that it is difficult to achieve optimal points, because of directly maximizing high-dimensional acquisition functions. Our goal is to tackle these two limitations and improve the performance of methods based on sufficient dimension reduction. The contributions of this paper are:

- We provide new results identifying why SIR-BO has poor performance. Firstly, we show that the mismatch between BO and SIR leads to poor performance of SIR. Secondly, the poor performance results from directly maximizing high-dimensional acquisition functions.
- To reduce the mismatch between BO and dimension reduction methods, we leverage kernel dimension reduction (KDR [9]) and manifold KDR [26] to get the projection matrix which requires no assumptions on the distribution of input covariates.
- To improve the performance when optimizing acquisition functions, we maximize the low-dimensional constrained acquisition function. Its constraint is

constructed by the inverse mapping using a batch of Gaussian Process models. Then it is easier for the acquisition function to discover better candidate points.
– We show empirically that our methods outperform a wide range of high-dimensional BO methods on several synthetic and real-world problems, including different kinds of synthetic functions and real-world problems. The experimental results show empirically that we have identified several important issues that affect the performance of BO methods based on supervised dimension reduction.

2 Related Work

In the realm of high-dimensional BO, there are generally three kinds of methods. The first kind of method assumes that there exists a lower-dimensional structure for objective functions. This kind of method typically contains three main stages: producing a low-dimensional embedding, performing standard BO in this low-dimensional space, and projecting found optimal points back to the original space. In REMBO [41], the low-dimensional embedding is achieved by using a random projection matrix. But many points produced by REMBO are outside the box bounds of the original space and they need to be clipped to the facet of the box, which results in a harmful distortion. Subsequently, several techniques are proposed to fix this problem [1,20]. In addition, the random low-dimensional embedding can be also achieved by randomized hashing functions [25,28]. The advantage of the hashing functions is that they can easily map candidate points back to the original space, avoiding clipping to facets of box bounds. Some works achieve linear embeddings based on learning. For example, SIR-BO achieves the projection matrix by using SIR, SI-BO [5] learns the projection matrix by low-rank matrix recovery, and [10] learns the projection matrix by maximizing the marginal likelihood of Gaussian process regression. Besides, some works achieve the nonlinear embedding based on VAE [11,23], but they need much more samples. Apart from producing an embedding, a few works select a subset of the dimensions and perform standard BO [19,21,35].

The second kind of method assumes that there exists an additive structure for the objective function, and the objective function can be decomposed into the sums of several low-dimensional functions. The additive objective function can be modeled by additive GPs [17]. With the additive GPs, the acquisition function based on GPs is also additive which can be maximized more efficiently to avoid the curse of dimensions. However, the true additive structure still remains challenging to learn. Several works propose to learn the underlying additive structure from the training data. For example, [40] assigns input variables into several groups by the Dirichlet process. [32] models the interactions between input variables using a dependency graph and learns the structure of the dependency graph from a few samples. The dependency graph can be used to assign input variables to overlapping groups. [12] restricts the dependency graph to a tree to reduce the computational complexity of maximizing acquisition functions.

Except for data-driven decompositions, RDUCB [46] learns a random tree-based decomposition to reduce the mismatch between the objective function and additive GPs.

The third kind of method directly improves high-dimensional BO, such as TuRBO [7] and EBO [39] based on local search, BO based on partitioning the whole space [18,38,42], better initialization method for AF [30,45], and so on.

3 Background

3.1 Bayesian Optimization

Bayesian optimization considers an optimization problem $\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ where f is a black-box and derivative-free function, and the feasible set $\mathcal{X}$ is a hyper-rectangle. BO is a sequential model-based approach, consisting of two main components: a surrogate model and an acquisition function. The surrogate model is used to model the objective function, and the acquisition function based on the surrogate model is used to decide where to sample next. Typically, we leverage Gaussian Process (GP) regression as the surrogate model [31], $f \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$ with mean function $m(\cdot)$ and a kernel $k(\cdot,\cdot)$. More specifically, GP assumes that evaluations of any finite number sampling point $\mathbf{x}_{1:n}$ have a joint Gaussian distribution, $\mathbf{f} \sim \mathcal{N}(\mathbf{m}(\mathbf{x}_{1:n}), \mathbf{K}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}))$. Under this assumption, given training data $\mathcal{D}_n = \{\mathbf{x}_{1:n}, \mathbf{y}_{1:n}\}$ and any new point $\mathbf{x}_*$, the joint distribution is given by

$$\begin{bmatrix} \mathbf{y}_{1:n} \\ f(\mathbf{x}_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m}(\mathbf{x}_{1:n}) \\ m(\mathbf{x}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) + \sigma_n^2 \mathbf{I} & \mathbf{k}(\mathbf{x}_{1:n}, \mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*, \mathbf{x}_{1:n}) & k(\mathbf{x}_*, \mathbf{x}_*), \end{bmatrix} \right)$$

where σ_n^2 is the variance of Gaussian noise added to the observations. It follows from the Sherman-Morrison-Woodbury formula that the posterior normal distribution for $f(\mathbf{x}_*)$ is given by $f(\mathbf{x}_*)|\mathcal{D}_n, \mathbf{x}_* \sim \mathcal{N}(\mu_n(\mathbf{x}_*), \sigma_n^2(\mathbf{x}_*))$ where

$$\mu_n(\mathbf{x}_*) = m(\mathbf{x}_*) + \mathbf{k}(\mathbf{x}_*, \mathbf{x}_{1:n})(\mathbf{K}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) + \sigma_n^2\mathbf{I})^{-1}(\mathbf{y}_{1:n} - \mathbf{m}(\mathbf{x}_{1:n}))$$
$$\sigma_n^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*, \mathbf{x}_{1:n})(\mathbf{K}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}) + \sigma_n^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}_{1:n}, \mathbf{x}_*)$$

Under that posterior, an acquisition function can be constructed to specify the utility of sampling points. In this paper, we use Expected Improvement (EI [15]) as the acquisition function. EI is defined as

$$\mathrm{EI}(\mathbf{x} \mid \mathcal{D}_n) = \mathbb{E}[\max\{f(\mathbf{x}) - f^*, 0\}]$$

where $f^* := \max \mathbf{y}_{1:n}$, $f(\mathbf{x}) \sim \mathcal{N}(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x}))$. And then the next sample point is given by maximizing the acquisition function, $\mathbf{x}_{n+1} \in \arg\max_{\mathbf{x}\in\mathcal{X}} \mathrm{EI}(\mathbf{x})$. After obtaining $\mathbf{x}_{n+1}$, the objective function at this point will be evaluated. Then the process proceeds to the next iteration.

3.2 Kernel Dimension Reduction

Kernel Dimension Reduction (KDR) [9] is one of the methods for sufficient dimension reduction whose aim is to find the sufficient subspace S such that $Y \perp\!\!\!\perp X \mid \Pi_S X$, where X is the input covariate and Y is its response. Its main advantage is that it requires no assumptions on the distribution of X or the conditional distribution of $Y|\Pi_S X$.

To begin with, KDR defines the *cross-covariance operator* of (X, Y) to characterize the correlations between X and Y, and then defines the *conditional covariance operator* $\Sigma_{YY|X}$ to characterize the conditional independence $Y \perp\!\!\!\perp X \mid \Pi_S X$. Under the conditional covariance operator, if $\Sigma_{YY|\mathbf{B}_1^\top X} \geq \Sigma_{YY|\mathbf{B}_2^\top X}$, then $\mathbf{B}_2^\top X$ captures more dependency between X and Y than $\mathbf{B}_1^\top X$. Hence, the conditional covariance operator results in a criterion of dimension reduction, i.e.

$$\mathbf{B}^* = \arg\min_{\mathbf{B}} \Sigma_{YY|\mathbf{B}^\top X}.$$

In practice, the empirical cross-covariance operator is estimated by $\hat{\mathbf{\Sigma}}_{\mathbf{YX}} := \hat{\mathbf{K}}_{\mathbf{Y}}\hat{\mathbf{K}}_{\mathbf{X}}$, where $\hat{\mathbf{K}}_{\mathbf{X}}$ and $\hat{\mathbf{K}}_{\mathbf{Y}}$ are the centered Gram matrices of training data $\mathbf{x}_{1:n}$ and $\mathbf{y}_{1:n}$, respectively. And the empirical conditional covariance operator is estimated by $\hat{\mathbf{\Sigma}}_{\mathbf{YY}|\mathbf{B}^\top\mathbf{X}} = \hat{\mathbf{K}}_{\mathbf{Y}}(\hat{\mathbf{K}}_{\mathbf{B}^\top\mathbf{X}} + n\epsilon_n\mathbf{I}_n)^{-1}$. Using the trace to evaluate the partial order of self-adjoint operators, minimizing the empirical conditional covariance operator is reduced to

$$\begin{aligned} \underset{\mathbf{B}}{\text{minimize}} \quad & \mathrm{Tr}[\hat{\mathbf{K}}_{\mathbf{Y}}(\hat{\mathbf{K}}_{\mathbf{B}^\top\mathbf{X}} + n\epsilon_n\mathbf{I}_n)^{-1}] \\ \text{subject to} \quad & \mathbf{B}^\top\mathbf{B} = \mathbf{I} \end{aligned} \tag{1}$$

3.3 Manifold KDR

Manifold KDR (MKDR) [26] extends KDR to nonlinear regression. To start with, MKDR achieves nonlinear information via unsupervised manifold learning. Then it performs KDR on manifolds.

Specifically, in this paper, we choose Diffusion Maps [4] and Geometric harmonics [3] as the unsupervised manifold learning methods to get the m-dimensional embedding $\mathbf{U} \in \mathbb{R}^{m\times n}$ for input covariates. MKDR approximates kernel function k via a linear map from the nonlinear embedding, i.e. $\mathbf{k}(\cdot, \mathbf{B}^\top\mathbf{x}_i) \approx \mathbf{\Phi}\mathbf{u}_i$ where $\mathbf{\Phi} \in \mathbb{R}^{m\times m}$. Under this approximation, the optimization problem 1 is formulated as

$$\begin{aligned} \underset{\mathbf{\Omega}}{\text{minimize}} \quad & \mathrm{Tr}[\hat{\mathbf{K}}_{\mathbf{Y}}(\mathbf{U}^\top\mathbf{\Omega}\mathbf{U} + n\epsilon_n\mathbf{I}_n)^{-1}] \\ \text{subject to} \quad & \mathbf{\Omega} \geq 0, \\ & \mathrm{Tr}(\mathbf{\Omega}) = 1. \end{aligned} \tag{2}$$

where $\mathbf{\Omega} := \mathbf{\Phi}^\top\mathbf{\Phi}$, and $\mathbf{\Omega} \geq 0$ means that $\mathbf{\Omega}$ is a semi-definitive matrix. According to the representation theory, the projection of $\mathbf{B}^\top\mathbf{x}$ on RKHS is $\mathbf{k}(\cdot, \mathbf{B}^\top\mathbf{x})$ that is approximated by $\mathbf{\Phi}\mathbf{u}$. Hence, $\mathbf{\Phi}\mathbf{U}$ is an estimator of the nonlinear sufficient subspace.

4 Challenges with SIR-BO

In this section, we discuss two issues that impact the performance of SIR-BO [44].

A Mismatch Between SIR and BO. The mismatch means that the distribution of data sampled by BO does not satisfy the assumptions of the dimensionality reduction methods. More specifically, SIR [22] imposes an assumption on conditional distribution $P(Y|X)$, whose performance can be powerful if data meets the assumption. However, if data violates the assumption, there is no guarantee for SIR to find the sufficient space. More specifically, SIR makes the assumption of linearity of conditional expectation, i.e. $\mathbb{E}[\mathbf{b}^\top \mathbf{x}|\mathbf{u}_1^\top \mathbf{x}, \ldots, \mathbf{u}_d^\top \mathbf{x}] = c_0 + c_1 \mathbf{u}_1^\top \mathbf{x} + \cdots + c_q \mathbf{u}_d^\top \mathbf{x},\ \forall \mathbf{b} \in \mathbb{R}^D$. The linearity condition is equivalent to that the distribution of X is elliptic (Condition 3.1 in [22]).

However, the distribution of X in BO does not follow an elliptic distribution. Actually, the sampling points of BO are determined by the acquisition function,

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x} \mid \mathcal{D}_t).$$

By the convergence of BO [36], the sampling points usually are clustered in one or several areas. Moreover, we also show this empirically, testing BO with three acquisition functions (EI [15], UCB [36], PI [37]) on the Ackley function, running each of them with 200 iterations. As shown in Fig. 1, the points produced by BO with UCB and PI are clustered in one area, blatantly violating elliptic symmetry. The sampling points of BO with EI are clustered in several areas, which apparently do not follow an elliptic distribution. Hence, in practice, we cannot expect the sampling points from BO will have an elliptic distribution.

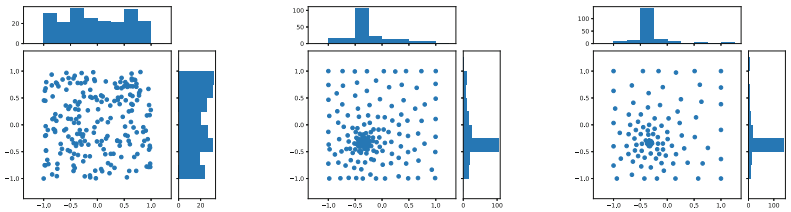


Fig. 1. We test BO with three acquisition functions on the Ackley function respectively: EI (left), PI (middle), and UCB (right).

The High-Dimensional Acquisition Functions with a Linear Subspace are Still Difficult to be Maximized. After achieving a projection matrix $\mathbf{B} \in \mathbb{R}^{D \times d}$, SIR-BO constructs the acquisition function $\alpha_d(\mathbf{B}^\top \mathbf{x})$. Although the function $\alpha_d(\cdot)$ is low-dimensional, the whole function is still high-dimensional. We will show this as follows.

We fit a Gaussian Process model on the 2-dimensional Ackley function, and construct three 2-dimensional acquisition functions: EI, PI, and UCB. We also construct three 60-dimensional acquisition functions by mapping three 2-dimensional acquisition functions to a 60-dimensional space using an orthogonal matrix. We run the typical zero-order optimization method DIRECT [14] on these 2-dimensional acquisition functions and 60-dimensional acquisition functions, respectively. As shown in Fig. 2, DIRECT finds the optimal point in 2-dimensional acquisition functions rapidly. However, it is difficult to find the optimal point on 60-dimensional acquisition functions with a 2-dimensional linear subspace.

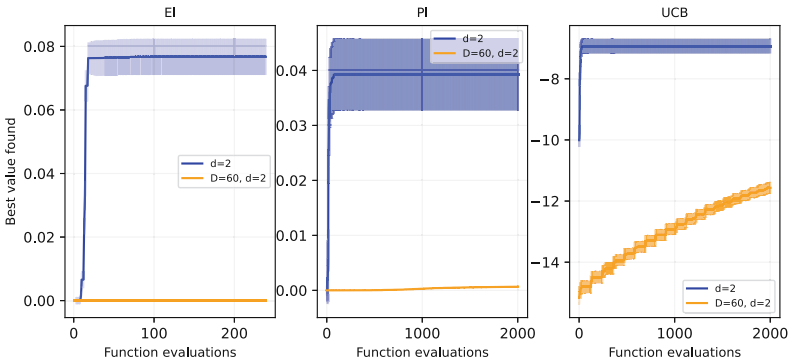


Fig. 2. We run DIRECT on three 2-dimensional acquisition functions and three 60-dimensional acquisition functions with 2-dimensional linear subspaces, showing optimal values by each iteration averaged over 20 repeated runs.

5 Algorithm

We now describe how to tackle the two issues described in Sect.4. On the one hand, we make use of KDR to learn the projection matrix $\mathbf{B}$ to estimate the central subspace such that $Y \perp\!\!\!\perp X \mid \mathbf{B}^\top X$. Note that KDR does not make assumptions of the distribution of X, which is more appropriate to BO than SIR. On the other hand, by adding the constraint $-\mathbf{1} \leq \mathbf{Bz} \leq \mathbf{1}$ to the acquisition function in the embedding space, we avoid maximizing the high-dimensional acquisition function.

In addition, to discover nonlinear manifolds of objective functions, we also combine MKDR and BO, where we use MKDR to achieve a nonlinear central subspace $\mathcal{M}$ and construct a batch of GPs to map the central subspace back to the original space, i.e. $F^{-1} : \mathcal{M} \to \mathcal{X}$. Under the inverse mapping, we can add the constraint $-\mathbf{1} \leq F^{-1}(\mathbf{z}) \leq \mathbf{1}$ to the acquisition function on the nonlinear manifold, which avoids maximizing the high-dimensional acquisition function.

Bayesian optimization based on dimension reduction usually contains three main procedures: producing a low-dimensional embedding, doing standard BO in this low-dimensional space, and projecting up to the original space. We will describe our methods following these three procedures.

5.1 Subspace Learning

Firstly, we consider that the objective function f has an effective linear subspace.

Definition 1 (Definition 1 in [41]). *A function $f : \mathbb{R}^D \to \mathbb{R}$ is said to have effective dimensionality d ($d < D$), if there exists a linear subspace $\mathcal{T}$ of dimension d such that for all $\mathbf{x}_\top \in \mathcal{T} \subset \mathbb{R}^D$ and $\mathbf{x}_\perp \in \mathcal{T}^\perp \subset \mathbb{R}^D$, we have $f(\mathbf{x}) = f(\mathbf{x}_\top + \mathbf{x}_\perp) = f(\mathbf{x}_\top)$, where $\mathcal{T}^\perp$ denotes the orthogonal complement of $\mathcal{T}$. $\mathcal{T}$ is called the effective linear subspace of f.*

Under this assumption, the following theorem shows that problems with an effective linear subspace can be solved via KDR.

Theorem 1. *Given a function $f : \mathbb{R}^D \to \mathbb{R}$ with an effective linear subspace $\mathcal{T}$ and the projection matrix $\mathbf{B}$ of KDR, then for any $\mathbf{x} \in \mathbb{R}^D$, there exists a $\mathbf{y} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = f(\mathbf{B}^\top \mathbf{y})$.*

Proof. Let $\mathbf{T} \in \mathbb{R}^{D \times d}$ be a matrix whose columns form an orthonormal basis for $\mathcal{T}$. Then for any $\mathbf{x}_\top \in \mathcal{T}$, there exists $\mathbf{c} \in \mathbb{R}^d$ such that $\mathbf{x}_\top = \mathbf{T}\mathbf{c}$. It is sufficient to show that there exists a $\mathbf{y} \in \mathbb{R}^d$ such that $\mathbf{B}^\top \mathbf{y} = \mathbf{T}\mathbf{c}$. Note that $(\mathbf{T}^\top \mathbf{B}^\top)(\mathbf{T}^\top \mathbf{B}^\top)^\top = \mathbf{I}$, hence $\mathbf{T}^\top \mathbf{B}^\top$ is an invertible matrix. Then there exists $\mathbf{y}$ such that $\mathbf{T}^\top \mathbf{B}^\top \mathbf{y} = \mathbf{c}$. Left multiplying both sides of the above equation by $\mathbf{T}$, it gets that $\mathbf{B}^\top \mathbf{y} = \mathbf{T}\mathbf{c} = \mathbf{x}_\top$.

The proof only requires the orthogonality of the projection matrix, and gets a similar conclusion of REMBO. Moreover, since $\mathbf{B}$ learned by KDR is an estimator of the central subspace, it leads to less mismatch between the actual objective function and the surrogate model on central subspaces when f does not have effective dimensionality.

Secondly, we consider that the objective function f has an effective smooth manifold $\mathcal{M}$ of dimension d. Formally, we suppose that there exists a smooth function $f_\mathcal{M} : \mathcal{M} \to \mathbb{R}$ such that $\forall \mathbf{x} \in \mathbb{R}^D, \exists \mathbf{y} \in \mathcal{M}, f(\mathbf{x}) = f_\mathcal{M}(\mathbf{y})$. Although manifold learning can bring the nonlinear information of manifold, yet the unsupervised learning usually leads to a mismatch between the true manifold and an estimated manifold. To reduce mismatches, we suppose there is a linear map $\mathbf{T}$ from an estimated manifold $\hat{\mathcal{M}}$ to the true manifold $\mathcal{M}$, and we try to find this linear map. Next, we extend the Proposition 1 in [20] to the nonlinear case. That is stationarity in a function with an effective smooth manifold implying stationarity in the estimated manifold.

Proposition 1. *Given a function on the true manifold is drawn from a GP with an RBF kernel: $f_\mathcal{M} \sim \mathcal{GP}(m(\cdot); k_{RBF}(\cdot, \cdot))$, and suppose there is a linear map*

$\mathbf{T}$ *from an estimated manifold* $\hat{\mathcal{M}}$ *to the true manifold* $\mathcal{M}$*. Then for any pair of points* $\mathbf{y}$ *and* $\mathbf{y}'$ *in the estimated manifold* $\hat{\mathcal{M}}$*,*

$$\mathrm{Cov}[f(\mathbf{x}), f(\mathbf{x}')] = \sigma^2 \exp\left[(\mathbf{y}-\mathbf{y}')^\top \mathbf{\Gamma}(\mathbf{y}-\mathbf{y}')\right]$$

where $\mathbf{x}, \mathbf{x}'$ *in the ambient space are pre-images of* $\mathbf{y}, \mathbf{y}'$ *respectively,* σ^2 *is the kernel variance of* $f_{\mathcal{M}}$*, and* $\mathbf{\Gamma}$ *is a symmetric and positive definite matrix.*

Proof. Under the assumption, we have $f(\mathbf{x}) = f_{\mathcal{M}}(\mathbf{Ty})$. Then the covariance function is given by

$$\begin{aligned}\mathrm{Cov}[f(\mathbf{x}), f(\mathbf{x}')] &= \mathrm{Cov}\left[f_{\mathcal{M}}(\mathbf{Ty}), f_{\mathcal{M}}(\mathbf{Ty}')\right] \\ &= \sigma^2 \exp\left[(\mathbf{Ty}-\mathbf{Ty}')^\top \mathbf{D}(\mathbf{Ty}-\mathbf{Ty}')\right] \\ &= \sigma^2 \exp\left[(\mathbf{y}-\mathbf{y}')^\top \mathbf{T}^\top \mathbf{DT}(\mathbf{y}-\mathbf{y}')\right]\end{aligned}$$

where $\mathbf{D} = \mathrm{diag}\left(\frac{1}{2\ell_1^2}, \ldots, \frac{1}{2\ell_d^2}\right)$ are inverse lengthscales of the RBF kernel. Let $\mathbf{\Gamma} = \mathbf{T}^\top \mathbf{DT}$. Since $\mathbf{D}$ is positive definite, $\mathbf{\Gamma}$ is symmetric and positive definite.

This proposition shows that an RBF kernel in the true manifold implies a Mahalanobis kernel in the estimated manifold. We apply the idea of manifold KDR to fit the linear map $\mathbf{T}$. Specifically, the diffusion map is applied to discover an estimated manifold $\hat{\mathcal{M}}$. Then we model the linear map $\mathbf{T}$ as the dimension reduction from the estimated manifold to the true manifold, and KDR is applied to find this dimension reduction. Finally, the $\mathbf{D}$ is fit by maximizing the marginal likelihood of GP with a Mahalanobis kernel in the estimated manifold.

5.2 Constrained Acquisition Function

The curse of dimensionality from acquisition functions (seen in Fig. 2) also leads to the poor performance of SIR-BO. We can avoid this by maximizing acquisition functions in the low-dimensional subspace. However, note that the objective function is restricted to box bounds, so we need to ensure sample points projected from the subspace are not outside the bounds. To do this, we need to add additional constraints to acquisition functions in the low-dimensional subspace.

The Constrained Acquisition Function in a Linear Subspace. Firstly, we consider that the low-dimensional subspace is linear. Since the projection matrix $\mathbf{B}$ is orthogonal, its Moore-Penrose inverse is $\mathbf{B}^\top$. Hence, we use $\mathbf{B}$ to project sample points in the subspace to the original space. Then following [20], we impose the linear constraint $-\mathbf{1} \leq \mathbf{By} \leq \mathbf{1}$ to the acquisition function,

$$\begin{aligned}\underset{\mathbf{z}\in\mathbb{R}^d}{\text{maximize}} \quad & \alpha(\mathbf{z}) \\ \text{subject to} \quad & -\mathbf{1} \leq \mathbf{Bz} \leq \mathbf{1}\end{aligned} \tag{3}$$

where $\alpha(\mathbf{z})$ is an acquisition function defined in the low-dimensional subspace. The above linearly constrained optimization problem can be solved via sequential quadratic programming (SQP) or interior-point methods [16]. We choose

interior-point methods to solve this problem instead of SQP, because SQP may produce solutions that violate the linear constraint. We combine KDR and the above constrained acquisition function into a new method for high-dimensional BO, called KDR-BO, summarized in Algorithm 1.

The Constrained Acquisition Function on a Nonlinear Subspace. Secondly, we consider that the low-dimensional subspace is nonlinear. To begin with, we need to construct the inverse mapping from the subspace to the original space, $F^{-1} : \mathcal{M} \to \mathcal{X}$. Let $F_1^{-1}, \ldots, F_d^{-1} : \mathcal{M} \to \mathbb{R}$ are component functions of the vector-valued function F^{-1}. Since these component functions are independent, we model each component function as an independent GP, and gather them in batches to gather with independent hyperparameters.

Then we impose the nonlinear constraint $-\mathbf{1} \leq F^{-1}(\mathbf{z}) \leq \mathbf{1}$ on the acquisition function. In addition, taking account of prediction errors, a 95% Bayesian credible interval, $\boldsymbol{\mu}(\mathbf{z}) \pm 1.96 \times \boldsymbol{\sigma}(\mathbf{z})$ should be included in the box bound,

$$\begin{aligned} \underset{\mathbf{z}\in\mathbb{R}^d}{\text{maximize}} \quad & \alpha(\mathbf{z}) \\ \text{subject to} \quad & \boldsymbol{\mu}_{\text{GPs}}(\mathbf{z}) - 1.96\boldsymbol{\sigma}_{\text{GPs}}(\mathbf{z}) \geq -\mathbf{1} \\ & \boldsymbol{\mu}_{\text{GPs}}(\mathbf{z}) + 1.96\boldsymbol{\sigma}_{\text{GPs}}(\mathbf{z}) \leq \mathbf{1}, \end{aligned} \tag{4}$$

The above problem is a nonlinearly constrained optimization problem, which can also be solved by interior-point methods and active-set SQP methods [16]. Here we choose interior-point methods to solve the above problem, because SQP sometimes produces solutions violating the nonlinear constraint. We combine MKDR and the above constrained acquisition function into a new method for high-dimensional BO, called MKDR-BO, summarized in Algorithm 2.

Alternatively, [24] uses multi-task GPs to model the inverse mapping. However, since component functions are independent, there is no reason to model correlations between the outputs by multi-task GPs.

The Initial Design for Constrained Acquisition Function. When maximizing constrained acquisition functions, the interior-point methods need to specify several initial points. However, the low-dimensional subspace is $\mathbb{R}^d$, in which it is difficult to select initial points. Hence, we firstly select random points in $\mathcal{X}$, and then transform them into the subspace. If the subspace is linear, then we directly use the projection matrix $\mathbf{B}^\top$ to transform initial points into the subspace. If the subspace is nonlinear, then we need to develop an out-of-sample method for MKDR, with which we subsequently transform initial points into the manifold. The out-of-sample method for MKDR is shown in Algorithm 3.

6 Benchmark Experiments

In this section, we evaluate our methods (KDR-BO, MKDR-BO) on a wide range of benchmarks: 50-dimensional synthetic functions, a 36-dimensional Neural Architecture Search problem, a 180-dimensional Lasso tuning problem, and a 124-dimensional vehicle design problem.

Algorithm 1: KDR-BO for linear embedding BO.

Input: n, T
Output: The sample points and their evaluations $\mathcal{D}_T$

$\mathcal{D}_1 = \{\mathbf{x_{1:n}}, \mathbf{y_{1:n}}\} \leftarrow$ Randomly sample n points from the feasible set $\mathcal{X}$ and then evaluate these points;
for $t \leftarrow n$ **to** T **do**
  $\mathbf{B} \leftarrow$ calculate kernel dimension reduction directions based on Eq. 1;
  Build Gaussian process regression based on low-dimensional data $\{\mathbf{z}_{1:t} = \mathbf{B}^\top \mathbf{x}_{1:t}, \mathbf{y}_{1:t}\}$;
  $\mathbf{z}^* \leftarrow$ maximize constrained acquisition function defined by Eq. 3 via interior-point methods;
  $\mathbf{x}_{t+1} = \mathbf{B}\mathbf{z}^* \leftarrow$ project the candidate point up to the original space;
  $y_{t+1} \leftarrow$ evaluate the candidate point;
  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{\mathbf{x}_{t+1}, y_{t+1}\}$;
return $\mathcal{D}_T$

We compare our methods (KDR-BO, MKDR-BO) to a broad selection of existing methods: linear embedding methods (REMBO, ALEBO, SIR-BO), nonlinear embedding methods (KSIR-BO [44]), BO using additive models (Add-GP-UCB [17]), local search methods (TuRBO), and quasirandom search (Sobol). For BO using embedding, we take $d = 10$ for these experiments. For Add-GP-UCB, we take $d = 4$ for each group. We test all methods using 15 initial points. Our code is available at https://github.com/qlchen2117/KDR-BO.

6.1 Synthetic Experiments

We consider the 50-dimensional Ackley function in the domain $[-5, 10]^{50}$, the 50-dimensional Levy function in the domain $[-5, 10]^{50}$, and the 50-dimensional Griewank function in the domain $[-300, 600]^{50}$. Each of these three functions has many local minima and a global minimum that is suitable for testing global optimization methods. Figure 3 shows that KDR-BO and MKDR-BO outperform other methods on the objective functions without additional structure. This shows that the central subspaces learned by KDR or MKDR lead to less mismatch between the actual objective function and the surrogate model on central subspaces even if f does not have effective dimensionality. Runtime performance analysis for synthetic experiments can be found in supplementary materials[1].

6.2 Real-World Problems

Neural Architecture Search. We consider the problem of neural architecture search (NAS) using models from NAS-Bench-101 [43]. The search space of neural architectures consists of all directed acyclic graphs with $V (V \leq 7)$ nodes and $E (E \leq 9)$ edges. Following [20], the above search space can be encoded to a

[1] https://github.com/qlchen2117/KDR-BO.

Algorithm 2: MKDR-BO for nonlinear embedding BO.

Input: n, T
Output: The sample points and their evaluations $\mathcal{D}_T$
$\mathcal{D}_1 = \{\mathbf{x_{1:n}}, \mathbf{y_{1:n}}\} \leftarrow$ Randomly sample n points from the feasible set $\mathcal{X}$ and then evaluate these points;
for $t \leftarrow n$ **to** T **do**
  $\mathbf{u}_{1:t} \leftarrow$ get nonlinear embeddings of $\mathbf{x}_{1:t}$ by Diffusion Map;
  $\mathbf{\Phi} \leftarrow$ calculate kernel dimension reduction directions based on Eq. 2;
  Build Gaussian process regression based on low-dimensional data $\{\mathbf{z}_{1:t} = \mathbf{\Phi}\mathbf{u}_{1:t}, \mathbf{y}_{1:t}\}$;
  $\mathbf{z}^* \leftarrow$ maximize constrained acquisition function defined by Eq. 4 via interior-point methods;
  Build a batch of Gaussian process models based on $\{\mathbf{z}_{1:t}, \mathbf{x}_{1:t}\}$;
  $\mathbf{x}_{t+1} = \boldsymbol{\mu}_{\mathrm{GPs}}(\mathbf{z}^*) \leftarrow$ project the candidate point up to the original space;
  $y_{t+1} \leftarrow$ evaluate the candidate point;
  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{\mathbf{x}_{t+1}, y_{t+1}\}$;
return $\mathcal{D}_T$

Algorithm 3: The out-of-sample extension for MKDR

Input: Out-of-sample points $\{\mathbf{x}'_{1:m}\}$
Output: The embeddings of out-of-sample points
$\mathbf{u}_{1:m} \leftarrow$ get nonlinear embeddings of $\mathbf{x}'_{1:m}$ via Geometric harmonics;
$\mathbf{\Phi} \leftarrow$ calculate kernel dimension reduction directions on training data based on Eq. 2;
$\mathbf{z}_{1:m} \leftarrow \mathbf{\Phi}\mathbf{u}_{1:m}$;
return $\mathbf{z}_{1:m}$

36-dimensional space $[0,1]^{15} \times \{0,1\}^{21}$. The goal of optimization is to find architectures that maximize the testing accuracy on CIFAR-10. Figure 4 shows that KDR-BO and MKDR-BO get higher accuracy than other embedding methods or additive BO on NAS. TuRBO gets the best performance of all methods.

Weighted Lasso Tuning. We consider the problem of tuning the Lasso (Least Absolute Shrinkage and Selection Operator) regression models. LassoBench [34] is a set of benchmark problems to tune penalty terms for Lasso models. In Lasso each regression coefficient corresponds to a penalty term, so the number of hyperparameters equals the number of features in the dataset. We consider tuning Lasso based on a 180-dimensional DNA dataset. The benchmark on DNA data has 43 effective dimensions. Figure 4 shows that KDR-BO gets the best performance of all methods on Lasso-DNA, and MKDR-BO also outperforms other methods on Lasso-DNA. This shows that KDR-BO and MKDR-BO can identify the effective subspaces of the Lasso problem rapidly.

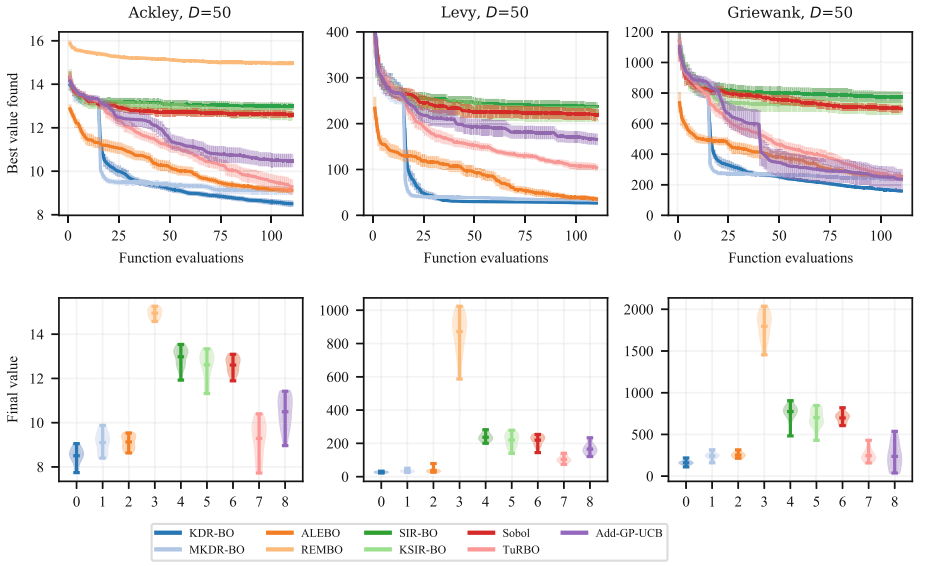


Fig. 3. We compare KDR-BO and MKDR-BO to baseline methods on three 50-dimensional functions, showing (Top row) optimal values by each iteration averaged over 20 repeated runs, and (Bottom row) the distribution over the final optimal values over 20 repeated runs.

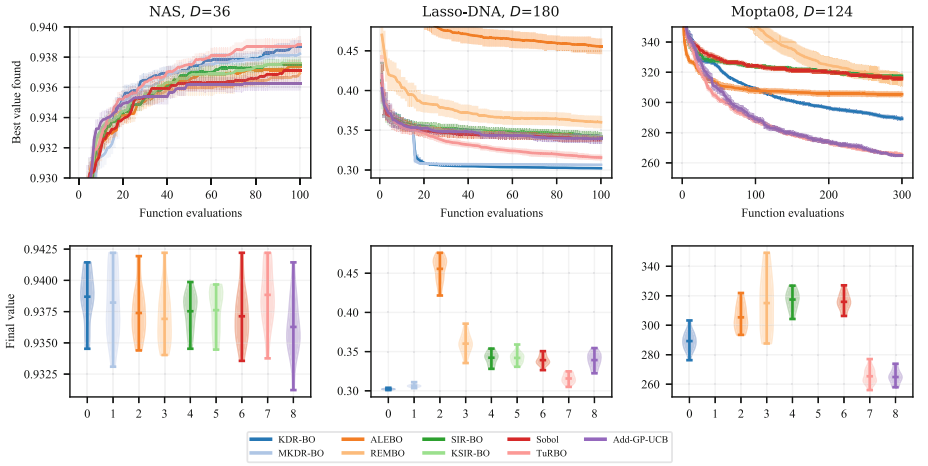


Fig. 4. We compare KDR-BO and MKDR-BO to baseline methods on the NAS problem ($D = 36$), Lasso hyperparameter tuning ($D = 180$), and MOPTA vehicle design ($D = 124$), showing (Top row) optimal values by each iteration averaged over 20 repeated runs, and (Bottom row) the distribution over the final optimal values over 20 repeated runs.

Vehicle Design. We consider the vehicle design problem with a soft penalty defined in [6]. The goal is to minimize the mass of a vehicle with 124 design variables describing materials, gauges, and vehicle shape. This results in a 124-dimensional BO problem. Figure 4 shows that KDR-BO outperforms other linear or nonlinear embedding methods on MOPTA08. TuRBO gets the best performance of all methods. The good performance of Add-GP-UCB is particularly interesting, because MOPTA08 has no additive structures and thus should be challenging for BO based on additive models.

7 Conclusion

Our work highlights the importance of two basic requirements for BO based on sufficient dimension reduction: (1) the assumption of the selected sufficient dimension reduction method should follow the data distribution of BO; and (2) it is necessary to construct a low-dimensional acquisition function, even though the high-dimensional acquisition function has an effective subspace. To the first point, we showed how data of BO violates the assumption of SIR, and we introduced KDR and MKDR to BO to reduce the mismatch between dimension reduction methods and BO. To the second point, to construct the constrained low-dimensional acquisition function, we construct the constraint via the inverse mapping from the central subspace back to the original space using a batch of GPs. We verify empirically that tackling these two issues improves the poor performance of SIR-BO and KSIR-BO on synthetic problems and real-world problems.

While we have obtained good performance using GPs to construct the inverse mapping, it is difficult to scale to much higher-dimensional input spaces or scale to a larger scale due to the computational complexity of GPs. In the future, we plan on extending MKDR-BO to higher-dimensional spaces and a larger scale.

Acknowledgments. This study was funded by Huawei-Nanjing University Technical Cooperation Project (20221108058).

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

References

1. Binois, M., Ginsbourger, D., Roustant, O.: On the choice of the low-dimensional domain for global optimization via random embeddings. J. Glob. Optim. **76**(1), 69–90 (2020). https://doi.org/10.1007/S10898-019-00839-1
2. Candelieri, A., Perego, R., Archetti, F.: Bayesian optimization of pump operations in water distribution systems. J. Glob. Optim. **71**(1), 213–235 (2018). https://doi.org/10.1007/S10898-018-0641-2
3. Coifman, R.R., Lafon, S.: Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions. Appl. Comput. Harmon. Anal. **21**(1), 31–52 (2006)

4. Coifman, R.R., Lafon, S.: Diffusion maps. Appl. Comput. Harmonic Anal. **21**(1), 5–30 (2006). https://doi.org/10.1016/j.acha.2006.04.006
5. Djolonga, J., Krause, A., Cevher, V.: High-dimensional gaussian process bandits. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States, pp. 1025–1033 (2013). https://proceedings.neurips.cc/paper/2013/hash/8d34201a5b85900908db6cae92723617-Abstract.html
6. Eriksson, D., Jankowiak, M.: High-dimensional bayesian optimization with sparse axis-aligned subspaces. In: de Campos, C.P., Maathuis, M.H., Quaeghebeur, E. (eds.) Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021, Virtual Event, 27-30 July 2021. Proceedings of Machine Learning Research, vol. 161, pp. 493–503. AUAI Press (2021). https://proceedings.mlr.press/v161/eriksson21a.html
7. Eriksson, D., Pearce, M., Gardner, J.R., Turner, R., Poloczek, M.: Scalable global optimization via local bayesian optimization. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp. 5497–5508 (2019). https://proceedings.neurips.cc/paper/2019/hash/6c990b7aca7bc7058f5e98ea909e924b-Abstract.html
8. Frazier, P.I.: A tutorial on bayesian optimization. CoRR arXiv:1807.02811 (2018)
9. Fukumizu, K., Bach, F.R., Jordan, M.I.: Kernel dimension reduction in regression. Ann. Stat. **37**(4), 1871–1905 (2009). http://www.jstor.org/stable/30243690
10. Garnett, R., Osborne, M.A., Hennig, P.: Active learning of linear embeddings for gaussian processes. In: Zhang, N.L., Tian, J. (eds.) Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI 2014, Quebec City, Quebec, Canada, July 23-27, 2014, pp. 230–239. AUAI Press (2014). https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2458&proceeding_id=30
11. Gómez-Bombarelli, R., et al.: Automatic chemical design using a data-driven continuous representation of molecules. ACS Cent. Sci. **4**(2), 268–276 (2018)
12. Han, E., Arora, I., Scarlett, J.: High-dimensional bayesian optimization via tree-structured additive models. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, pp. 7630–7638. AAAI Press (2021).https://doi.org/10.1609/AAAI.V35I9.16933
13. Hvarfner, C., Stoll, D., Souza, A.L.F., Lindauer, M., Hutter, F., Nardi, L.: πbo: augmenting acquisition functions with user beliefs for bayesian optimization. In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net (2022). https://openreview.net/forum?id=MMAeCXIa89
14. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the lipschitz constant. J. Optim. Theory Appl. **79**, 157–181 (1993)
15. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. **13**(4), 455–492 (1998). https://doi.org/10.1023/A:1008306431147
16. Jorge, N., Stephen, J.W.: Numerical Optimization. Spinger (2006). https://doi.org/10.1007/978-0-387-40065-5

17. Kandasamy, K., Schneider, J.G., Póczos, B.: High dimensional bayesian optimisation and bandits via additive models. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. JMLR Workshop and Conference Proceedings, vol. 37, pp. 295–304. JMLR.org (2015). http://proceedings.mlr.press/v37/kandasamy15.html
18. Kawaguchi, K., Kaelbling, L.P., Lozano-Pérez, T.: Bayesian optimization with exponential convergence. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 2809–2817 (2015). https://proceedings.neurips.cc/paper/2015/hash/0ebcc77dc72360d0eb8e9504c78d38bd-Abstract.html
19. Kirschner, J., Mutny, M., Hiller, N., Ischebeck, R., Krause, A.: Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 3429–3438. PMLR (2019). http://proceedings.mlr.press/v97/kirschner19a.html
20. Letham, B., Calandra, R., Rai, A., Bakshy, E.: Re-examining linear embeddings for high-dimensional bayesian optimization. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual (2020). https://proceedings.neurips.cc/paper/2020/hash/10fb6cfa4c990d2bad5ddef4f70e8ba2-Abstract.html
21. Li, C., Gupta, S., Rana, S., Nguyen, V., Venkatesh, S., Shilton, A.: High dimensional bayesian optimization using dropout. In: Sierra, C. (ed.) Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017, pp. 2096–2102. ijcai.org (2017). https://doi.org/10.24963/ijcai.2017/291
22. Li, K.C.: Sliced inverse regression for dimension reduction. J. Am. Stat. Assoc. **86**(414), 316–327 (1991). http://www.jstor.org/stable/2290563
23. Lu, X., González, J., Dai, Z., Lawrence, N.D.: Structured variationally auto-encoded optimization. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 3273–3281. PMLR (2018). http://proceedings.mlr.press/v80/lu18c.html
24. Moriconi, R., Deisenroth, M.P., Kumar, K.S.S.: High-dimensional bayesian optimization using low-dimensional feature spaces. Mach. Learn. **109**(9–10), 1925–1943 (2020). https://doi.org/10.1007/S10994-020-05899-Z
25. Nayebi, A., Munteanu, A., Poloczek, M.: A framework for bayesian optimization in embedded subspaces. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 4752–4761. PMLR (2019). http://proceedings.mlr.press/v97/nayebi19a.html
26. Nilsson, J., Sha, F., Jordan, M.I.: Regression on manifolds using kernel dimension reduction. In: Ghahramani, Z. (ed.) Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007. ACM International Conference Proceeding Series, vol. 227, pp. 697–704. ACM (2007). https://doi.org/10.1145/1273496.1273584

27. Oh, C., Gavves, E., Welling, M.: BOCK : Bayesian optimization with cylindrical kernels. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 3865–3874. PMLR (2018). http://proceedings.mlr.press/v80/oh18a.html
28. Papenmeier, L., Nardi, L., Poloczek, M.: Increasing the scope as you learn: adaptive bayesian optimization in nested subspaces. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022). https://openreview.net/forum?id=e4Wf6112DI
29. Powell, W.B.: A unified framework for stochastic optimization. Eur. J. Oper. Res. **275**(3), 795–821 (2019). https://doi.org/10.1016/J.EJOR.2018.07.014
30. Rana, S., Li, C., Gupta, S., Nguyen, V., Venkatesh, S.: High dimensional bayesian optimization with elastic gaussian process. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 2883–2891. PMLR (2017). http://proceedings.mlr.press/v70/rana17a.html
31. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. Adaptive Computation and Machine Learning, MIT Press (2006). https://www.worldcat.org/oclc/61285753
32. Rolland, P., Scarlett, J., Bogunovic, I., Cevher, V.: High-dimensional bayesian optimization via additive models with overlapping groups. In: Storkey, A.J., Pérez-Cruz, F. (eds.) International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain. Proceedings of Machine Learning Research, vol. 84, pp. 298–307. PMLR (2018). http://proceedings.mlr.press/v84/rolland18a.html
33. Ru, B.X., Wan, X., Dong, X., Osborne, M.A.: Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021). https://openreview.net/forum?id=j9Rv7qdXjd
34. Sehic, K., Gramfort, A., Salmon, J., Nardi, L.: Lassobench: a high-dimensional hyperparameter optimization benchmark suite for lasso. In: Guyon, I., Lindauer, M., van der Schaar, M., Hutter, F., Garnett, R. (eds.) International Conference on Automated Machine Learning, AutoML 2022, 25-27 July 2022, Johns Hopkins University, Baltimore, MD, USA. Proceedings of Machine Learning Research, vol. 188, pp. 2/1–24. PMLR (2022). https://proceedings.mlr.press/v188/sehic22a.html
35. Shen, Y., Kingsford, C.: Computationally efficient high-dimensional bayesian optimization via variable selection. In: Faust, A., Garnett, R., White, C., Hutter, F., Gardner, J.R. (eds.) Proceedings of the Second International Conference on Automated Machine Learning. Proceedings of Machine Learning Research, vol. 224, pp. 15/1–27. PMLR (2023). https://proceedings.mlr.press/v224/shen23a.html
36. Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.W.: Gaussian process optimization in the bandit setting: no regret and experimental design. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel, pp. 1015–1022. Omnipress (2010). https://icml.cc/Conferences/2010/papers/422.pdf
37. Viana, F., Haftka, R.: Surrogate-based optimization with parallel simulations using the probability of improvement. In: 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference, p. 9392 (2010)

38. Wang, L., Fonseca, R., Tian, Y.: Learning search space partition for black-box optimization using monte carlo tree search. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual (2020). https://proceedings.neurips.cc/paper/2020/hash/e2ce14e81dba66dbff9cbc35ecfdb704-Abstract.html
39. Wang, Z., Gehring, C., Kohli, P., Jegelka, S.: Batched large-scale bayesian optimization in high-dimensional spaces. In: Storkey, A.J., Pérez-Cruz, F. (eds.) International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain. Proceedings of Machine Learning Research, vol. 84, pp. 745–754. PMLR (2018). http://proceedings.mlr.press/v84/wang18c.html
40. Wang, Z., Li, C., Jegelka, S., Kohli, P.: Batched high-dimensional bayesian optimization via structural kernel learning. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 3656–3664. PMLR (2017). http://proceedings.mlr.press/v70/wang17h.html
41. Wang, Z., Hutter, F., Zoghi, M., Matheson, D., de Freitas, N.: Bayesian optimization in a billion dimensions via random embeddings. J. Artif. Intell. Res. **55**, 361–387 (2016). https://doi.org/10.1613/jair.4806
42. Wang, Z., Shakibi, B., Jin, L., de Freitas, N.: Bayesian multi-scale optimistic optimization. In: Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014. JMLR Workshop and Conference Proceedings, vol. 33, pp. 1005–1014. JMLR.org (2014). http://proceedings.mlr.press/v33/wang14d.html
43. Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: Nas-bench-101: towards reproducible neural architecture search. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 7105–7114. PMLR (2019). http://proceedings.mlr.press/v97/ying19a.html
44. Zhang, M., Li, H., Su, S.W.: High dimensional bayesian optimization via supervised dimension reduction. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, pp. 4292–4298. ijcai.org (2019). https://doi.org/10.24963/ijcai.2019/596
45. Zhao, J., Yang, R., QIU, S., Wang, Z.: Unleashing the potential of acquisition functions in high-dimensional bayesian optimization. Trans. Mach. Learn. Res. (2024). https://openreview.net/forum?id=0CM7Hfsy61
46. Ziomek, J.K., Bou-Ammar, H.: Are random decompositions all we need in high dimensional bayesian optimisation? In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA. Proceedings of Machine Learning Research, vol. 202, pp. 43347–43368. PMLR (2023). https://proceedings.mlr.press/v202/ziomek23a.html

Evolve Cost-Aware Acquisition Functions Using Large Language Models

Yiming Yao[1,2,3(✉)], Fei Liu[2,3], Ji Cheng[2,3], and Qingfu Zhang[2,3(✉)]

[1] City University of Hong Kong (Dongguan), Dongguan 523000, China
yimingyao3-c@my.cityu.edu.hk
[2] Department of Computer Science, City University of Hong Kong, Hong Kong, China
{fliu36-c,J.Cheng}@my.cityu.edu.hk, qingfu.zhang@cityu.edu.hk
[3] The City University of Hong Kong Shenzhen Research Institute, Shenzhen, China

Abstract. Many real-world optimization scenarios involve expensive evaluation with unknown and heterogeneous costs. Cost-aware Bayesian optimization stands out as a prominent solution in addressing these challenges. To approach the global optimum within a limited budget in a cost-efficient manner, the design of cost-aware acquisition functions (AFs) becomes a crucial step. However, traditional manual design paradigm typically requires extensive domain knowledge and involves a labor-intensive trial-and-error process. This paper introduces EvolCAF, a novel framework that integrates large language models (LLMs) with evolutionary computation (EC) to automatically design cost-aware AFs. Leveraging the crossover and mutation in the algorithmic space, EvolCAF offers a novel design paradigm, significantly reduces the reliance on domain expertise and model training. The designed cost-aware AF maximizes the utilization of available information from historical data, surrogate models and budget details. It introduces novel ideas not previously explored in the existing literature on acquisition function design, allowing for clear interpretations to provide insights into its behavior and decision-making process. In comparison to the well-known EIpu and EI-cool methods designed by human experts, our approach showcases remarkable efficiency and generalization across various tasks, including 12 synthetic problems and 3 real-world hyperparameter tuning test sets.

Keywords: Cost-aware Bayesian optimization · Acquisition functions · Large language models · Evolutionary computation

1 Introduction

Bayesian optimization (BO) is a powerful tool for solving expensive optimization problems and has found wide application in many real-world scenarios [7,8,27,32]. It typically employs a surrogate model to approximate the expensive function and well-designed acquisition functions (AFs) to select potential solutions in a sample-efficient manner. Popular acquisition functions include probability of

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 374–390, 2024.
https://doi.org/10.1007/978-3-031-70068-2_23

improvement (PI) [15], expected improvement (EI) [25], upper confidence bound (UCB) [31], knowledge gradient (KG) [6], etc.

Vanilla BO typically sets the number of evaluations as the budget constraint, implicitly assuming a uniform evaluation cost in the design space [17]. However, this is rarely the case in many real-world applications, leading to the concept of cost-aware BO. For example, in hyperparameter optimization (HPO) tasks for machine learning models, the costs with different hyperparameter configurations may even differ in the order of magnitudes [17]. Under the budget constraint of accumulated costs, approaching the global optimum is a challenge for traditional AFs due to their unawareness of heterogeneous evaluation costs, which highlights the importance of designing efficient cost-aware AFs.

Previous works have proposed several heuristics to take the cost information into account [16,17,30]. Representative ones include EI per unit cost (EIpu) [30] which divides EI by the cost function to promote solutions with both low cost and high improvement, and EI-cool [17] that introduces the cost-cooling strategy to make EIpu adapt to problems with expensive global optimum. Based on EIpu and EI-cool, several enhanced approaches have been suggested [9,23,28]. However, designing these AFs typically necessitates significant involvement of domain experts and extensive trial-and-error testing to refine and improve upon previous methods. This manual design paradigm is labor-intensive and non-automated. Furthermore, the information currently used to define cost-aware AFs is inadequate, as it only considers the EI metric and budget details while overlooking the complete historical data, which severely restricts the exploration in the algorithmic space. Simply integrating the EI metric with budget information does not inherently stimulate innovative ideas, thereby greatly limiting the performance and generalization of the designed AFs. While some model-based methods have been proposed to automatically learn AFs parameterized by neural networks in meta-BO community [12,24,34], they often require substantial effort in complex framework design and model training. Additionally, the resulting AFs are represented by network parameters, leading to poor interpretability compared to widely used AFs that have explicit mathematical expressions. Besides, these methods are designed for problems with uniform costs and are not applicable to many real-world applications with heterogeneous costs.

In the past three years, large language models (LLMs) have been widely used in code generation [11,18,22], mathematical reasoning [29] and automatic algorithm design [19,20]. While recent studies have explored the use of LLMs to enhance vanilla BO [21,36], these approaches rely on querying LLMs to directly suggest candidate solutions. The absence of a clearly defined search strategy results in inadequate explainability. Moreover, whenever a new problem arises, it needs to conduct a substantial number of queries for LLMs from scratch, which can be expensive and impractical for real-world applications.

We propose a novel paradigm, named EvolCAF, which integrates LLMs in an evolutionary framework to automatically design explicit AFs to enhance cost-aware BO. Different from the existing works, it enjoys good automation and explainability outperforming existing human-crafted methods. To the best of

our knowledge, this is the first attempt to utilize LLMs for automatic AF design for Bayesian optimization. Our main contributions are as follows:

- We introduce EvolCAF, which integrates large language models (LLMs) with evolutionary computation (EC) to automatically design cost-aware AFs. It enables crossover and mutation in the algorithmic space to iteratively search for elite AFs, significantly reducing the reliance on expert knowledge and domain model training.
- We leverage EvolCAF to design a cost-aware AF that fully utilizes the available history information. Remarkably, the designed AF introduces novel ideas that have not been explored in existing literature on acquisition function design. The designed AF can be expressed explicitly, allowing for clear interpretations to provide insights into its behavior and decision-making process.
- We evaluate the designed AF on diverse synthetic functions as well as practical hyperparameter optimization (HPO) problems. Compared to the popular EIpu and EI-cool methods designed by domain experts, our approach demonstrates remarkable efficiency and generalization, which highlights the promising potential in addressing many related real-world applications.

2 Background and Related Works

2.1 Background

In vanilla Bayesian optimization (BO), we consider finding the optimal solution $\mathbf{x}^*$ that maximizes the black-box objective function f: $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, where $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$, we assume $f : \mathcal{X} \to \mathbb{R}$ is continuously differentiable and expensive to evaluate.

Gaussian Processes. To approximate the expensive objective function, BO typically employs a Gaussian process (GP) model [1] as the surrogate. A GP is an infinite distribution over functions f specified by a prior mean function $\mu(\cdot)$ and covariance function $k(\cdot,\cdot)$: $f(\mathbf{x}) \sim \mathcal{GP}_f\left(\mu(\mathbf{x}), k\left(\mathbf{x}, \mathbf{x}'\right)\right)$. Suppose in iteration t, the historical data set $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$ are obtained from the observation model $y_i = f(\mathbf{x}_i) + \epsilon_i$ with observation noise $\epsilon_i \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$. Given the test point $\mathbf{x}^*$, the predictive distribution $p(y|\mathbf{x}^*, \mathcal{D}_t)$ is also Gaussian with mean $\mu(\mathbf{x}^*) = K_{\mathbf{x}^*,\mathbf{X}}(K_{\mathbf{X},\mathbf{X}} + \sigma_\epsilon^2\mathrm{I})^{-1}\mathbf{y}$ and variance $\sigma^2(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}^*) - K_{\mathbf{x}^*,\mathbf{X}}(K_{\mathbf{X},\mathbf{X}} + \sigma_\epsilon^2\mathrm{I})^{-1}K_{\mathbf{X},\mathbf{x}^*}$, where $\mathbf{y} = [y_1, y_2, \cdots, y_t]^T$ are noisy output values observed from the latent functions $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_t)]^T$ at training points $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t]^T$, $K_{\mathbf{X},\mathbf{X}} = [k\left(\mathbf{x}_i, \mathbf{x}_j\right)]_{\mathbf{x}_i,\mathbf{x}_j\in\mathbf{X}}$ is the covariance matrix and $K_{\mathbf{X},\mathbf{x}^*} = [k\left(\mathbf{x}_i, \mathbf{x}^*\right)]_{\mathbf{x}_i\in\mathbf{X}}$ is the correlation vector for all training and test points.

Acquisition Functions. The acquisition function (AF) defines a utility that measures the benefit of evaluating an unknown point $\mathbf{x}$. We denote the definition of AF as $\alpha(\mathbf{x})$, which may contain the historical data set $\mathcal{D}_t$, model information $\mu(\mathbf{x})$, $\sigma^2(\mathbf{x})$, etc. One of the popular AFs is expected improvement (EI) [25], which quantifies the expected amount of improvement over the current best observation $y^* = \max_i y_i$ at a given point $\mathbf{x}$ in the search space:

$$\alpha_{\mathrm{EI}}(\mathbf{x}) = \mathbb{E}_{f(\mathbf{x})}[[f(\mathbf{x}) - y^*]_+] = \sigma(\mathbf{x})\ h(\frac{\mu(\mathbf{x}) - y^*}{\sigma(\mathbf{x})}), \tag{1}$$

where $[\cdot]_+$ is the $\max(0, \cdot)$ operation, $h(z) = \phi(z) + z\Phi(z)$, ϕ and Φ are the standard normal density and distribution functions, respectively.

2.2 Cost-Aware Bayesian Optimization

Problem Setting. In cost-aware Bayesian optimization, it is assumed that evaluating the objective function is expensive. Additionally, the evaluation in different regions will incur heterogeneous costs, the unknown cost function is denoted as $c(\mathbf{x})$. For every query $\mathbf{x}_i$, we can obtain the noisy observation y_i with cost $z_i = c(\mathbf{x}_i) + \eta_i$, where $\eta_i \sim \mathcal{N}\left(0, \sigma_\eta^2\right)$. Similar to the objective function f, the black-box cost function is modeled as a draw from the Gaussian process $c(\mathbf{x}) \sim \mathcal{GP}_c$. We use the posterior predictive mean of $\mathcal{GP}_c$ for calculating the cost function as [23,30] do.

Given the historical data set $\bar{\mathcal{D}}_t = \{(\mathbf{x}_i, y_i, z_i)\}_{i=1}^t$ with t evaluated samples and the limited total budget B_{total}, we can only find a near-optimal solution with the constraint of cumulative cost $\sum_{i=1}^T z_i \leq B_{\mathrm{total}}$, where T is the maximum number of evaluated samples that satisfies the budget constraint. The general framework followed by the vast majority of existing cost-aware BO methods is shown in Algorithm 1, the main difference lies in the different definitions of cost-aware AFs, which will be introduced below.

Algorithm 1. Cost-aware BO

Input:

B_{total}: total budget, $\bar{\mathcal{D}}_t$: initial data set with t evaluated samples

1: Initialize used budget $B_{\mathrm{used}} = \sum_{i=1}^t z_i$
2: Train objective and cost models $\mathcal{GP}_f$ and $\mathcal{GP}_c$ using $\bar{\mathcal{D}}_t$
3: **while** $B_{\mathrm{used}} < B_{\mathrm{total}}$ **do**
4: Query candidate: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x})$
5: Evaluate candidate: $y_{t+1}, z_{t+1} \leftarrow f(\mathbf{x}_{t+1}), c(\mathbf{x}_{t+1})$
6: Update data set $\bar{\mathcal{D}}_{t+1} \leftarrow \bar{\mathcal{D}}_t \cup \{(\mathbf{x}_{t+1}, y_{t+1}, z_{t+1})\}$
7: Update models $\mathcal{GP}_f$ and $\mathcal{GP}_c$ using $\bar{\mathcal{D}}_{t+1}$
8: $B_{\mathrm{used}} \leftarrow B_{\mathrm{used}} + z_{t+1}$
9: $t \leftarrow t + 1$
10: **end while**
11: Total number of evaluated samples $T \leftarrow t$

Output: Best configuration $\arg\max_{(\mathbf{x}_i, y_i) \in \bar{\mathcal{D}}_T} y_i$

EI per Unit Cost (EIpu). To balance the cost and quality of evaluations, Snoek et al. [30] proposed EI per unit cost (EIpu), which normalizes the EI metric by the cost function $c(\mathbf{x})$:

$$\alpha_{\mathrm{EIpu}}(\mathbf{x}) = \frac{\alpha_{\mathrm{EI}}(\mathbf{x})}{c(\mathbf{x})}. \tag{2}$$

By using the cost function to penalize EI, EIpu tends to carefully select candidate points with low cost and high improvement, making the search process cost-aware. Benefiting from the preference for cheaper regions, EIpu can be consistently improved when the optimum is cheap to evaluate since higher EI and lower cost are both encouraged. However, the preference becomes a drawback when the optimum lies in the expensive regions of the design space, which is common in many real-world applications. The cost penalty term prevents EIpu from exploring near-optimal regions that are expensive to evaluate, experiments have shown that sometimes EIpu performs even worse than EI [17].

EI-Cool. To alleviate the above problem, Lee et al. [17] introduced a cost-cooling factor α in EIpu called EI-cool:

$$\alpha_{\mathrm{EI-cool}}(\mathbf{x}) = \frac{\alpha_{\mathrm{EI}}(\mathbf{x})}{c(\mathbf{x})^{\alpha}}, \tag{3}$$

where $\alpha = (B_{\mathrm{total}} - B_{\mathrm{used}})/(B_{\mathrm{total}} - B_{\mathrm{init}})$, B_{init} is the budget spent in evaluating the initial sample points, B_{used} is the budget already used and B_{total} is the given total budget. As B_{used} increases from B_{init} to B_{total} during the search process, the factor α gradually decays from 1 to 0, resulting in the transition of EI-cool from EIpu to EI. Intuitively, the cost-cooling strategy diminishes the significance of the cost model as the budget is consumed, making EI-cool to operate in an *early and cheap, late and expensive* fashion to encourage exploring expensive regions when the remaining budget is tight.

Although EI-cool alleviates the problem of performance degradation when searching for the expensive optimum, it always uses EIpu as the starting strategy, which is not flexible and may not adapt well to different problems [23]. Besides, previous analysis and experiments have shown that when the remaining budget is gradually tight, although deemphasizing the cost can increase the likelihood of exploring expensive regions, it is still possible to miss the optimum in high-cost regions before the budget is exhausted [23], as the exploration or exploitation in very cheap regions can still result in very large values of EI-cool metric, which is called low-cost-preference weakness in [28].

Variants Based on EIpu and EI-Cool. Based on EIpu and EI-cool, some improved methods have been proposed such as using a multi-armed bandit algorithm to automatically select either EI or EIpu [23], developing more aggressive methods to alleviate the low-cost-preference weakness of EI-cool [28], and cost-aware EI based on Pareto optimality to achieve the trade-off between cost and

improvement [9]. It is evident that the design process typically requires significant involvement of domain knowledge and extensive trial-and-error testing based on the flaws of previous methods, which are labor-intensive and non-automated. Besides, the existing AFs just combine the EI metric with budget information in different ways, which severely restricts the exploration in the algorithmic space and does not inherently foster the generation of innovative ideas, so the performance and generalization are greatly limited.

2.3 Automatic Design for Acquisition Functions

In the meta-BO community, which focuses on meta-learning or learning to learn to enhance vanilla BO [1,4,5,33], there has been research dedicated to automatically generating efficient and generalizable AFs via a learning model. While these works are not tailored for cost-aware contexts, we will review them to emphasize the strengths and potential of our framework.

To learn a meta-acquisition function, Volpp et al. [34] replaced the hand-designed AF with a neural network named neural acquisition function (NAF), which is meta-trained on related source tasks by policy-based reinforcement learning. Hsieh et al. [12] utilized a deep Q-network (DQN) as a surrogate differentiable AF to achieve a few-shot fast adaptation of AFs. Maraval et al. [24] introduced an end-to-end differentiable framework based on transformer architectures called neural acquisition process (NAP) to meta-learn acquisition functions with the surrogate model jointly. Nevertheless, despite achieving promising results, these model-based methods often demand substantial effort in framework design and model training. Moreover, the resulting AFs are represented by network parameters, resulting in poor interpretability compared to widely used AFs that have explicit mathematical expressions.

3 EvolCAF: Evolve Cost-Aware Acquisition Functions with LLMs

3.1 Framework

The proposed EvolCAF framework embraces the basic components of evolutionary computing (EC), including initialization, crossover, mutation, and population management. In EvolCAF, each individual represents an acquisition function solving a branch of synthetic instances, which is represented with an algorithm description and a code block implementation instead of an encoded vector in traditional EC. During the evolution process, the initialization, crossover, and mutation operations on the individuals are all performed by prompting LLM in the algorithmic space. The entire process is completely automated without any intervention from human experts.

Figure 1 illustrates the detailed flowchart of EvolCAF. At each generation, we maintain a population of N AFs, each AF is evaluated on a set of synthetic instances in a cost-aware BO loop to calculate the fitness value, which is the

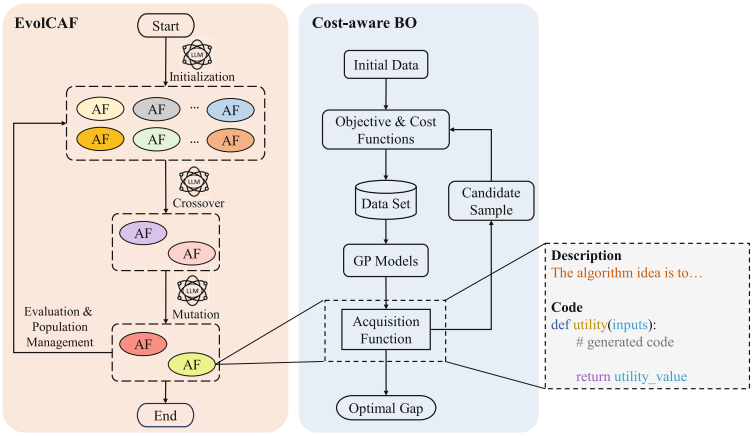


Fig. 1. Flowchart of EvolCAF framework. The left box presents the evolution of cost-aware AFs enabled by EvolCAF, wherein each individual in the population is an AF, represented with an algorithm description and a code block implementation. The initialization, crossover, and mutation are facilitated by LLMs. The middle box shows the cost-aware BO loop, each AF is evaluated on a set of synthetic instances to calculate the optimal gap as its fitness value.

optimal gap between the true optimal value and the optimal value obtained by the AF. After new individuals are added to the population, the worst individuals are deleted according to the fitness values.

3.2 General Definition for Evolved AFs

The general definition for evolved cost-aware AFs can be formulated as follows:

$$\begin{aligned}\alpha_{\text{EvolCAF}}(\mathbf{x}) &= \alpha(\mathbf{x}; \boldsymbol{\theta}_{\text{data}}, \boldsymbol{\theta}_{\text{model}}, \boldsymbol{\theta}_{\text{budget}}) \\ &= \alpha(\mathbf{x}; \mathbf{X}, \mathbf{y}, \mathbf{x}^*, y^*, \mu(\mathbf{x}), \sigma(\mathbf{x}), c(\mathbf{x}), B_{\text{used}}, B_{\text{total}}).\end{aligned} \tag{4}$$

The inputs of cost-aware AFs incorporate three groups of information: (1) historical data $\boldsymbol{\theta}_{\text{data}}=\{\mathbf{X}, \mathbf{y}, \mathbf{x}^*, y^*\}$, (2) prediction and uncertainty provided by the model $\boldsymbol{\theta}_{\text{model}}=\{\mu(\mathbf{x}), \sigma(\mathbf{x})\}$, and (3) budget information during the optimization $\boldsymbol{\theta}_{\text{budget}}=\{c(\mathbf{x}), B_{\text{used}}, B_{\text{total}}\}$. The first two groups include the data and model information for searching with uncertainties, while the last group informs the AF of the budget constraints in optimization. During the evolutionary process, EvolCAF is encouraged to explore the algorithmic space to generate and refine elite cost-aware AFs.

3.3 Prompt Engineering

The general format of prompt engineering used to inform LLMs consists of four parts: (1) a general description of the task, (2) code instructions for implementing

Fig. 2. Prompts used in EvolCAF for initialization, crossover, and mutation.

algorithms, including the function name, inputs and output, (3) interpretations for the inputs and output, including their detailed meanings in our task, the variable formats and dimensions implemented in the code, (4) helpful hints to inform LLMs to generate executable codes and utilize input information as much as possible to create novel ideas. Following the general format, in initialization, we instruct LLMs to create a completely new AF to promote population diversity. In crossover, we suggest combining the selected parent AFs to facilitate the preservation of high-performing components in the following generations. While in mutation, we aim to encourage the exploration of better AFs based on parent AFs in the algorithmic space. The details of prompts for initialization, crossover, and mutation are shown in Fig. 2.

4 Experimental Studies

4.1 Experimental Settings

Settings for AF Evolution. In the evolutionary process, EvolCAF maintains 10 AFs and evolves over 20 generations. We generate 1 offspring individual in each generation based on 2 parent individuals, with the crossover probability set to 1.0 and mutation probability set to 0.5. The GPT-3.5-turbo pre-trained LLM is used for generating AFs.

To generate AFs that can be efficiently optimized, we set a time threshold of 60 s for completing the cost-aware BO loop, which serves as a selection pressure together with the fitness value. Any AF that exceeds this time limit will be automatically eliminated during the evolutionary process.

Settings for Cost-Aware BO. To calculate the fitness value, we evaluate each evolved AF on 2D Ackley and 2D Rastrigin functions with 10 different random seeds in the experimental design, resulting in a total of 20 instances. The aim is to achieve improved generalization results across various initial surrogate landscapes with an acceptable evaluation time during evolution, as the training and inference of GP models on a large number of instances in each generation can be expensive. The fitness value for each evolved AF is calculated by averaging the optimal gaps obtained from the 20 instances.

To simulate the scenarios in many real-world applications, we carefully design a cost function that is most expensive to evaluate at the global optimum $\mathbf{x}^*$ of the synthetic function, the formulation is similar to that used in [16] and can be expressed as:

$$c(\mathbf{x}) = \exp\left(-\left\|\mathbf{x} - \mathbf{x}^*\right\|_2\right), \tag{5}$$

where each dimension of $\mathbf{x}$ and $\mathbf{x}^*$ is normalized to [0,1]. In order to achieve good results for the evolved AF given a small budget, we set the total budget B_{total} as 30 in the evolutionary process, indicating that the smallest number of evaluations is 30, we will further verify the generalization using sufficient budget in the following experiments. We initialize $2d$ random samples using experimental design, where d is the dimension of the decision variable.

All BO methods are implemented using BoTorch [2]. In the BO loop, the acquisition functions are optimized through multi-start optimization using scipy's L-BFGS-B optimizer, using 20 restarts seeded from 100 pseudo-random samples through BoTorch's initialization heuristic for efficient optimization.

4.2 Evolution Results

Figure 3 demonstrates the evolutionary process. In each generation, we maintain 10 AFs represented by blue dots. The mean and optimal fitness values of the population are represented with orange and red lines, respectively. With a population size of 10 and 20 generations, the fitness value of the evolving AFs can converge to a notably low level. The results show the capability of our framework to automatically generate and evolve elite AFs.

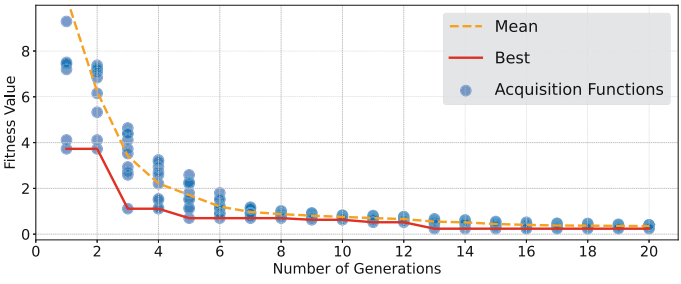


Fig. 3. The evolutionary process of acquisition functions.

Figure 4 shows the optimal AF designed by EvolCAF with the minimum fitness value, including a general description of the algorithmic idea in defining the AF and a detailed code implementation. After converting the code into easily understandable mathematical expressions, we observe that the optimal AF consists of three parts:

$$\alpha_{\text{EvolCAF}}(\mathbf{x}) = \alpha_1(\mathbf{x}) + \alpha_2(\mathbf{x}) + \alpha_3(\mathbf{x}). \tag{6}$$

Specifically, $\alpha_1(\mathbf{x})$ combines a modified EI with uncertainty information:

$$\begin{aligned}\alpha_1(\mathbf{x}) =&[(\mu(\mathbf{x}) - y^*)\Phi(\mathbf{z}) + \sqrt{\sigma^2(\mathbf{x}) + \sigma^2(\mathbf{y})} \cdot \phi(\mathbf{z})]\cdot \\ &(1 - \log \sqrt{\frac{\sigma^2(\mathbf{x}) + \sigma^2(\mathbf{y})}{\sigma^2(\mathbf{y})}}),\end{aligned} \tag{7}$$

where $\mathbf{z} = \frac{\mu(\mathbf{x}) - y^*}{\sqrt{\sigma^2(\mathbf{x}) + \sigma^2(\mathbf{y})}}$, ϕ and Φ are the standard normal density and distribution functions, respectively, $\sigma^2(\mathbf{y})$ represents the variance of the current historical observations.

Similar to EI, $\alpha_1(\mathbf{x})$ encourages searching regions close to the current best observation with uncertainties. However, an improvement is that $\alpha_1(\mathbf{x})$ also incorporates the uncertainty of all historical observations rather than solely focusing on the current best observation value and the uncertainty of the unknown point $\mathbf{x}$.

$\alpha_2(\mathbf{x})$ mainly focuses on the current remaining budget and the cost of evaluating the unknown point $\mathbf{x}$:

$$\alpha_2(\mathbf{x}) = -\frac{B_{\text{total}} - B_{\text{used}}}{e^{c(\mathbf{x})}}. \tag{8}$$

It can be observed that $\alpha_2(\mathbf{x})$ enables the optimization to focus on EI regardless of the cost when the remaining budget is tight, which is similar to the cost-cooling strategy used in EI-cool. However, the difference is that $\alpha_2(\mathbf{x})$ will not only keep the optimization encouraging higher EI metrics but also promote the exploration of expensive regions when there is a sufficient budget. This feature addresses the low-cost-preference weakness in EI-cool, allowing for a more comprehensive search.

$\alpha_3(\mathbf{x})$ considers the distance between the unknown inputs and historical observed locations when optimizing AF:

$$\alpha_3(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} \min_j A_{ij}, \tag{9}$$

where $A_{ij} = dist(\mathbf{u}_i, \mathbf{v}_j)$ is the distance matrix, $\{\mathbf{u}_i\}_{i=1}^m$ are multiple starting points used in the efficient multi-start optimization scheme of BoTorch, $\mathbf{v}_j$ is the jth element in observed locations $\mathbf{X}$. Therefore, $\alpha_3(\mathbf{x})$ utilizes the information of all the distances between the unobserved location in each optimization trajectory and observed ones in the historical data set. When optimizing $\alpha_{\text{EvolCAF}}(\mathbf{x})$,

Description

The new algorithm idea is to calculate the utility of each test input by incorporating the concept of uncertainty reduction through mutual information, where the focus is on maximizing the information gain from the unobserved test input in relation to the learned surrogate model and the cost and budget constraints. It involves adjusting the utility value based on the reduction in uncertainty, the cost of observation, the remaining budget, and the distance of the test input from the known solutions, to provide an efficient and adaptive approach to prioritizing the evaluation of test inputs.

Code

```
import torch
import torch.distributions as tdist

def utility(train_x, train_y, best_x, best_y, test_x, mean_test_y, std_test_y, cost_test_y, budget_used, budget_total):

    # Calculate the expected improvement (EI) for each test input
    with torch.no_grad():
        z = (mean_test_y - best_y) / torch.sqrt(torch.pow(std_test_y, 2) + torch.pow(train_y.std(), 2))
        ei = (mean_test_y - best_y) * tdist.Normal(0, 1).cdf(z) + torch.sqrt(torch.pow(std_test_y, 2) + torch.pow(train_y.std(), 2)) * tdist.Normal(0, 1).log_prob(z).exp()

    # Calculate the reduction in uncertainty through mutual information
    with torch.no_grad():
        mi = torch.max(torch.tensor(0), (torch.log(torch.pow(std_test_y, 2) + torch.pow(train_y.std(), 2)) - torch.log(torch.pow(train_y.std(), 2))) / 2)

    # Adjust the utility value based on the reduction in uncertainty, cost of evaluation, and remaining budget
    utility_value = (ei * (1 - mi)) - torch.exp(-cost_test_y) * (budget_total - budget_used)

    # Calculate the distance of the test input from the known solutions
    distance_to_known = torch.cdist(test_x, train_x)

    # Adjust the utility value based on the diversity and coverage of the unobserved test input space
    diversity_coverage_factor = torch.mean(torch.min(distance_to_known, dim=1).values)
    utility_value += diversity_coverage_factor
    return utility_value
```

Fig. 4. Optimal acquisition function designed by EvolCAF. The results include a linguistic description of the algorithmic idea, as well as a code implementation with annotations, all the contents are produced by LLMs.

maximizing the average of the minimum distances contributed by $\alpha_3(\mathbf{x})$ forces the multi-start optimization away from explored regions.

The analyses above suggest that the designed AF can introduce novel ideas that have not been previously explored in existing literature on acquisition function design. Benefiting from the evolution in the algorithmic space, the designed AF can be expressed explicitly, allowing for clear interpretations to provide insights into its behavior and decision-making process.

4.3 Evaluation of the Optimal Acquisition Function

Synthetic Problems. In this subsection, we evaluate the optimal AF on 12 different synthetic instances with different landscapes and input dimensions. We define the cost functions according to Equation (5). As the evolution is conducted within a total budget of 30, which is to enable the optimal AF to achieve better results using a small budget, therefore, we also tested the generalization of the optimal AF within a sufficient budget of 300. Each test instance is conducted with 10 independent runs, the results are shown in Table 1.

Based on the experimental results, it can be observed that in the vast majority of cases, the optimal AF achieves significantly better performance than EI and other cost-aware AFs. The optimal AF demonstrates strong generalization capabilities across unseen instances with diverse landscapes and sufficient budget constraints. In addition to the promising performance, an interesting phenomenon is that within a fixed budget, the optimal AF uses fewer evaluations in most cases, which is more pronounced when the budget is sufficient.

Table 1. Means of optimal gaps (number of evaluated samples) obtained by different AFs on all synthetic instances over 10 independent runs. The best mean result for each row is highlighted in bold.

Test Instances	Budget	EI	EIpu	EI-cool	EvolCAF
Ackley 2D	30	2.6600(40)	2.3302(40)	2.7369(40)	**0.4277(34)**
	300	1.2295(395)	0.8582(399)	0.8317(399)	**0.0505(306)**
Rastrigin 2D	30	4.7425(41)	5.6155(41)	5.7754(40)	**0.0511(34)**
	300	1.6656(410)	1.6678(408)	1.8518(408)	**0.0046(306)**
Griewank 2D	30	0.4875(35)	0.3384(36)	0.3374(36)	**0.1762(33)**
	300	0.1305(323)	0.1195(323)	0.1360(323)	**0.0361(307)**
Rosenbrock 2D	30	1.2609(41)	2.3601(44)	2.2909(42)	**0.0304(33)**
	300	0.0332(369)	0.0406(394)	**0.0317(372)**	0.0402(307)
Levy 2D	30	0.0056(38)	0.0098(38)	0.0116(38)	**0.0013(33)**
	300	1.1517e-4(314)	**5.9321e-5(316)**	8.1046e-5(317)	3.7248e-4(307)
ThreeHumpCamel 2D	30	0.0483(39)	0.1182(40)	0.0710(39)	**0.0007(33)**
	300	5.0446e-4(322)	7.4557e-4(326)	**2.6392e-4(325)**	7.5310e-4(306)
StyblinskiTang 2D	30	0.0286(41)	0.0233(42)	0.0266(41)	**0.0071(33)**
	300	1.4420e-4(332)	1.8616e-4(339)	**6.1798e-5(343)**	2.0142e-3(306)
Hartmann 3D	30	5.6696e-5(40)	1.0364e-4(41)	**4.6158e-5(40)**	4.8127e-4(36)
	300	1.8263e-5(420)	1.3089e-5(429)	**9.0599e-6(432)**	2.3656e-4(311)
Powell 4D	30	18.8892(48)	19.8281(51)	14.9481(49)	**0.1285(38)**
	300	2.9839(376)	1.1173(395)	1.6806(391)	**0.0136(316)**
Shekel 4D	30	7.9123(48)	7.9210(49)	8.2132(48)	**2.6367(39)**
	300	6.5193(545)	6.9044(545)	7.0135(551)	**0.1993(315)**
Hartmann 6D	30	0.0326(52)	0.0296(52)	**0.0278(52)**	0.0384(44)
	300	0.0122(710)	0.0054(705)	0.0154(695)	**0.0042(327)**
Cosine8 8D	30	0.4723(48)	0.4738(48)	0.5351(48)	**0.4357(53)**
	300	0.1707(532)	0.2364(533)	0.2779(527)	**0.0148(342)**

To verify the contribution of each component in the optimal AF, we further display the performance of EvolCAF after removing each of the three components, as shown in Table 2. It can be observed that removing $\alpha_2(\mathbf{x})$, which takes into account budget information, has the greatest impact on the final performance of EvolCAF. Compared with EI, EIpu and EI-cool, the cost-aware AFs that remove $\alpha_1(\mathbf{x})$ or $\alpha_3(\mathbf{x})$ can still achieve better results. The results indicate the effectiveness and superiority of the designed acquisition function.

Hyperparameter Tuning Task. Here we evaluate the optimal AF on 3 practical hyperparameter tuning test sets to further validate the effectiveness of our method. We utilize the surrogate benchmark implemented in JAHS-Bench-201 [3] to train a randomly generated neural network architecture with 2 continuous and 4 categorical hyperparameters: learning rate in $[10^{-3}, 1]$, weight decay in $[10^{-5}, 10^{-2}]$, depth multiplier in $\{1, 3, 5\}$, width multiplier in $\{4, 8, 16\}$, resolution multiplier in $\{0.25, 0.5, 1.0\}$, and training epochs in $\{1, 2, ..., 200\}$. We

Table 2. Means of optimal gaps (number of evaluated samples) obtained by different AFs on all synthetic instances over 10 independent runs. The best, second best, and worst mean results for each row are highlighted with bold fonts, underlines, and shaded background, respectively.

Test Instances	Budget	EI	EIpu	EI-cool	w/o alpha1	w/o alpha2	w/o alpha3	EvolCAF
	30	2.6600(40)	2.3302(40)	2.7369(40)	0.6422(34)	4.8046(40)	1.5008(33)	**0.4277(34)**
Ackley 2D	300	1.2295(395)	0.8582(399)	0.8317(399)	0.7301(308)	1.2677(345)	**0.0501(305)**	0.0505(306)
	30	4.7425(41)	5.6155(41)	5.7754(40)	0.0746(34)	5.7126(42)	0.2648(33)	**0.0511(34)**
Rastrigin 2D	300	1.6656(410)	1.6678(408)	1.8518(408)	0.0842(308)	0.8550(383)	0.0157(306)	**0.0046(306)**
	30	0.4875(35)	0.3384(36)	0.3374(36)	0.1913(34)	0.6671(36)	0.3535(33)	**0.1762(33)**
Griewank 2D	300	0.1305(323)	0.1195(323)	0.1360(323)	0.0522(308)	0.1954(324)	0.0598(306)	**0.0361(307)**
	30	1.2609(41)	2.3601(44)	2.2909(42)	0.0399(33)	2.1626(39)	2.3467(33)	**0.0304(33)**
Rosenbrock 2D	300	0.0332(369)	0.0406(394)	0.0317(372)	**0.0104(308)**	0.0587(348)	1.8554(307)	0.0402(307)
	30	0.0056(38)	0.0098(38)	0.0116(38)	**0.0006(34)**	0.0335(37)	0.0195(33)	0.0013(33)
Levy 2D	300	1.1517e-4(314)	**5.9321e-5(316)**	8.1046e-5(317)	0.0003(307)	0.0009(327)	0.0010(306)	3.7248e-4(307)
	30	0.0483(39)	0.1182(40)	0.0710(39)	**0.0006(34)**	0.0940(38)	0.0188(33)	0.0007(33)
ThreeHumpCamel 2D	300	5.0446e-4(322)	7.4557e-4(326)	**2.6392e-4(325)**	0.0005(308)	0.0020(332)	0.0099(307)	7.5310e-4(306)
	30	0.0286(41)	0.0233(42)	0.0266(41)	0.0136(33)	1.7123(41)	0.0713(33)	**0.0071(33)**
StyblinskiTang 2D	300	1.4420e-4(332)	1.8616e-4(339)	**6.1798e-5(343)**	0.0069(307)	0.0246(332)	0.0042(306)	2.0142e-3(306)
	30	5.6696e-5(40)	1.0364e-4(41)	**4.6158e-5(40)**	0.0007(36)	0.1438(51)	0.0731(36)	4.8127e-4(36)
Hartmann 3D	300	1.8263e-5(420)	1.3089e-5(429)	**9.0599e-6(432)**	0.0004(311)	0.0124(446)	0.0005(311)	2.3656e-4(311)
	30	18.8892(48)	19.8281(51)	14.9481(49)	0.1719(39)	36.0514(56)	4.3751(37)	**0.1285(38)**
Powell 4D	300	2.9839(376)	1.1173(395)	1.6806(391)	0.0205(316)	1.7473(450)	0.2997(317)	**0.0136(316)**
	30	7.9123(48)	7.9210(49)	8.2132(48)	**2.3629(39)**	9.2281(60)	3.5714(37)	2.6367(39)
Shekel 4D	300	6.5193(545)	6.9044(545)	7.0135(551)	0.3430(316)	8.1076(554)	**0.1583(313)**	0.1993(315)
	30	0.0326(52)	0.0296(52)	**0.0278(52)**	0.0343(44)	1.7641(83)	0.1305(42)	0.0384(44)
Hartmann 6D	300	0.0122(710)	0.0054(705)	0.0154(695)	0.0044(326)	0.4572(750)	0.0127(327)	**0.0042(327)**
	30	0.4723(48)	0.4738(48)	0.5351(48)	0.4438(53)	1.5106(88)	0.6058(46)	**0.4357(53)**
Cosine8 8D	300	0.1707(532)	0.2364(533)	0.2779(527)	0.0161(343)	1.1518(755)	0.0425(347)	**0.0148(342)**

use ReLU [10] activations and stochastic gradient descent (SGD) optimizer, and do not use trivial augment [26] for data augmentation in the training pipeline. The hyperparameter tuning task is conducted on three different image classification datasets: CIFAR-10 [14], Colorectal-Histology [13] and Fashion-MNIST [35], we record the validation accuracy and total runtime as the observations of the objective and evaluation cost for each candidate configuration, respectively. For more details and implementation, please refer to [3].

Table 3 shows the results achieved by all AFs using different total runtimes (denoted as C, in minutes) as budgets. It can be observed that the optimal AF achieves the best results on CIFAR-10 and Fashion-MNIST datasets. In addition, we demonstrate the results within 25 and 50 total evaluations (denoted as T) when the total runtime is sufficient. It can be observed that the optimal AF can achieve the best results in most cases, which further proves that our method is still sample-efficient, while EIpu and EI-cool perform worse than EI.

To make a further illustration, we present the histograms of evaluation cost frequency on CIFAR-10 data set with C=4000 as an example, as shown in Fig. 5. It is evident that the majority of search frequencies of all cost-aware AFs are concentrated in cheap regions. While EIpu and EI-cool demonstrate a greater ability to explore expensive regions compared to EI, the optimal AF has the potential to explore regions that are significantly more expensive than those searched by EIpu and EI-cool, resulting in superior performance.

Table 3. Means (stds) of validation accuracies obtained by different AFs on all data sets over 10 independent runs. The best mean result for each row is highlighted in bold.

Data Set	Budget	EI	EIpu	EI-cool	EvolCAF
CIFAR-10	C=2000	0.6885(0.05)	0.6934(0.06)	0.6849(0.05)	**0.7495(0.04)**
	C=4000	0.7975(0.03)	0.7951(0.04)	0.7721(0.03)	**0.8002(0.03)**
	T=25	0.7065(0.06)	0.6765(0.08)	0.6744(0.08)	**0.8030(0.04)**
	T=50	**0.8394(0.03)**	0.7980(0.07)	0.7744(0.08)	0.8351(0.02)
Colorectal-Histology	C=1000	0.8883(0.04)	**0.9140(0.01)**	0.9125(0.02)	0.8901(0.02)
	C=2000	0.9039(0.05)	**0.9273(0.01)**	0.9196(0.01)	0.9158(0.01)
	T=25	0.8779(0.04)	0.8592(0.08)	0.8588(0.09)	**0.9072(0.02)**
	T=50	0.9040(0.02)	0.8944(0.06)	0.8910(0.08)	**0.9199(0.01)**
Fashion-MNIST	C=5000	0.9021(0.03)	0.9191(0.01)	0.9188(0.01)	**0.9370(0.01)**
	C=10000	0.9238(0.02)	0.9318(0.01)	0.9355(0.008)	**0.9425(0.007)**
	T=25	0.8986(0.03)	0.8711(0.06)	0.8730(0.06)	**0.9349(0.01)**
	T=50	0.9218(0.02)	0.8810(0.06)	0.8928(0.05)	**0.9445(0.002)**

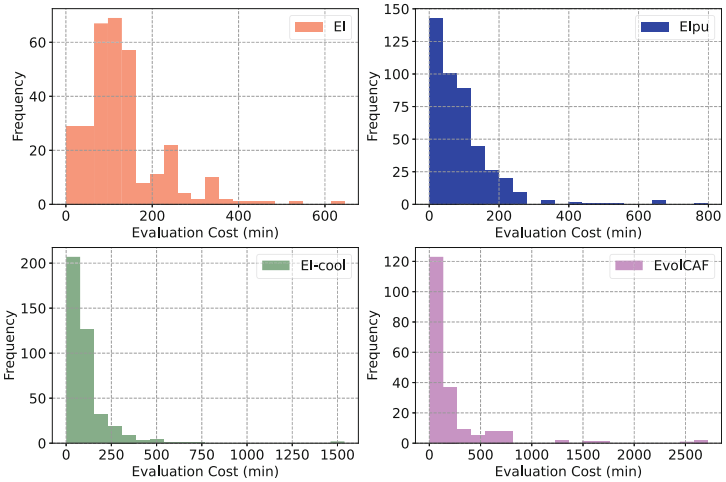


Fig. 5. Histograms of evaluation cost frequency collected in 10 independent runs on CIFAR-10 data set with C=4000.

5 Conclusion

This paper introduces EvolCAF, a novel framework that integrates large language models (LLMs) with evolutionary computation (EC) to automatically design cost-aware AFs. Leveraging crossover and mutation in the algorithmic space, EvolCAF offers a novel design paradigm that significantly reduces the reliance on domain expertise and model training. The designed AF showcases novel ideas not previously explored in the existing literature on AF design, allowing for clear interpretations to provide insights into its behavior and

decision-making process. Compared to the well-known EIpu and EI-cool methods designed by human experts, our approach demonstrates remarkable efficiency and generalization across various tasks, including 12 synthetic problems and 3 real-world hyperparameter tuning test sets. We have deployed our method to the latest proposed automatic heuristic design platform named EoH [19], the source code can be found in https://github.com/FeiLiu36/EoH/tree/main/examples/user_bo_caf.

In future work, we expect that the EvolCAF framework can be well adapted to other popular BO settings, such as high-dimensional BO, batch BO, multi-objective BO. Furthermore, we are going to explore the integration of different types of cost functions into the evolutionary process to enhance the robustness of the designed AF.

Acknowledgments. The work described in this paper was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China [GRF Project No. CityU-11215723], the Natural Science Foundation of China (Project No: 62276223), and the Key Basic Research Foundation of Shenzhen, China (JCYJ20220818100005011).

Disclosure of Interests. The authors declare that they have no known competing interest that could appear to influence the work reported in this paper.

References

1. Bai, T., Li, Y., Shen, Y., Zhang, X., Zhang, W., Cui, B.: Transfer learning for bayesian optimization: a survey. arXiv preprint arXiv:2302.05927 (2023)
2. Balandat, M., et al.: Botorch: a framework for efficient monte-carlo bayesian optimization. Adv. Neural. Inf. Process. Syst. **33**, 21524–21538 (2020)
3. Bansal, A., Stoll, D., Janowski, M., Zela, A., Hutter, F.: JAHS-bench-201: a foundation for research on joint architecture and hyperparameter search. Adv. Neural. Inf. Process. Syst. **35**, 38788–38802 (2022)
4. Chen, Y., et al.: Learning to learn without gradient descent by gradient descent. In: International Conference on Machine Learning, pp. 748–756. PMLR (2017)
5. Chen, Y., et al.: Towards learning universal hyperparameter optimizers with transformers. Adv. Neural. Inf. Process. Syst. **35**, 32053–32068 (2022)
6. Frazier, P.I., Powell, W.B., Dayanik, S.: A knowledge-gradient policy for sequential information collection. SIAM J. Control. Optim. **47**(5), 2410–2439 (2008)
7. Frazier, P.I., Wang, J.: Bayesian optimization for materials design. In: Lookman, T., Alexander, F.J., Rajan, K. (eds.) Information Science for Materials Discovery and Design. SSMS, vol. 225, pp. 45–75. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-23871-5_3
8. Garnett, R., Osborne, M.A., Roberts, S.J.: Bayesian optimization for sensor set selection. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pp. 209–219 (2010)
9. Guinet, G., Perrone, V., Archambeau, C.: Pareto-efficient acquisition functions for cost-aware bayesian optimization. arXiv preprint arXiv:2011.11456 (2020)
10. Hahnloser, R.H., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S.: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature **405**(6789), 947–951 (2000)

11. Hemberg, E., Moskal, S., O'Reilly, U.M.: Evolving code with a large language model. arXiv preprint arXiv:2401.07102 (2024)
12. Hsieh, B.J., Hsieh, P.C., Liu, X.: Reinforced few-shot acquisition function learning for bayesian optimization. Adv. Neural. Inf. Process. Syst. **34**, 7718–7731 (2021)
13. Kather, J.N., Weis, C.A., Bianconi, F., Melchers, S.M., Schad, L.R., Gaiser, T., Marx, A., Zöllner, F.G.: Multi-class texture analysis in colorectal cancer histology. Sci. Rep. **6**(1), 1–11 (2016)
14. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009)
15. Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. J. Basic Eng. **86**(1), 97–106 (1964)
16. Lee, E.H., Eriksson, D., Perrone, V., Seeger, M.: A nonmyopic approach to cost-constrained bayesian optimization. In: Uncertainty in Artificial Intelligence, pp. 568–577. PMLR (2021)
17. Lee, E.H., Perrone, V., Archambeau, C., Seeger, M.: Cost-aware bayesian optimization. arXiv preprint arXiv:2003.10870 (2020)
18. Lehman, J., Gordon, J., Jain, S., Ndousse, K., Yeh, C., Stanley, K.O.: Evolution through large models. In: Banzhaf, W., Machado, P., Zhang, M. (eds.) Handbook of Evolutionary Machine Learning. Genetic and Evolutionary Computation. Springer, Singapore (2022). https://doi.org/10.1007/978-981-99-3814-8_11
19. Liu, F., Tong, X., Yuan, M., Lin, X., Luo, F., Wang, Z., Lu, Z., Zhang, Q.: Evolution of heuristics: towards efficient automatic algorithm design using large language model. In: Proceedings of International Conference on Machine Learning (2024)
20. Liu, F., Tong, X., Yuan, M., Zhang, Q.: Algorithm evolution using large language model. arXiv preprint arXiv:2311.15249 (2023)
21. Liu, T., Astorga, N., Seedat, N., van der Schaar, M.: Large language models to enhance bayesian optimization. arXiv preprint arXiv:2402.03921 (2024)
22. Liventsev, V., Grishina, A., Härmä, A., Moonen, L.: Fully autonomous programming with large language models. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1146–1155 (2023)
23. Luong, P., Nguyen, D., Gupta, S., Rana, S., Venkatesh, S.: Adaptive cost-aware bayesian optimization. Knowl.-Based Syst. **232**, 107481 (2021)
24. Maraval, A., Zimmer, M., Grosnit, A., Bou Ammar, H.: End-to-end meta-bayesian optimisation with transformer neural processes. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
25. Močkus, J.: On bayesian methods for seeking the extremum. In: Marchuk, G.I. (ed.) Optimization Techniques 1974. LNCS, vol. 27, pp. 400–404. Springer, Heidelberg (1975). https://doi.org/10.1007/3-540-07165-2_55
26. Müller, S.G., Hutter, F.: Trivialaugment: tuning-free yet state-of-the-art data augmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 774–782 (2021)
27. Negoescu, D.M., Frazier, P.I., Powell, W.B.: The knowledge-gradient algorithm for sequencing experiments in drug discovery. Informs J. Comput. **23**(3), 346–363 (2011)
28. Qian, W., He, Z., Li, L., Liu, X., Gao, F.: Cobabo: a hyperparameter search method with cost budget awareness. In: 2021 IEEE 7th International Conference on Cloud Computing and Intelligent Systems (CCIS), pp. 408–412. IEEE (2021)
29. Romera-Paredes, B., et al.: Mathematical discoveries from program search with large language models. Nature **625**(7995), 468–475 (2024)

30. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, vol. 25 (2012)
31. Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.: Gaussian process optimization in the bandit setting: no regret and experimental design. arXiv preprint arXiv:0912.3995 (2009)
32. Turner, R., et al.: Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In: NeurIPS 2020 Competition and Demonstration Track, pp. 3–26. PMLR (2021)
33. TV, V., Malhotra, P., Narwariya, J., Vig, L., Shroff, G.: Meta-learning for black-box optimization. In: Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C. (eds.) ECML PKDD 2019. LNCS (LNAI), vol. 11907, pp. 366–381. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46147-8_22
34. Volpp, M., et al.: Meta-learning acquisition functions for transfer learning in bayesian optimization. arXiv preprint arXiv:1904.02642 (2019)
35. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
36. Zhang, M.R., Desai, N., Bae, J., Lorraine, J., Ba, J.: Using large language models for hyperparameter optimization. In: NeurIPS 2023 Foundation Models for Decision Making Workshop (2023)

A Surrogate-Assisted Partial Optimization for Expensive Constrained Optimization Problems

Kei Nishihara(✉) and Masaya Nakata

Yokohama National University, Yokohama, Kanagawa 2408501, Japan
nishihara-kei-jv@ynu.jp, nakata-masaya-tb@ynu.ac.jp

Abstract. Surrogate-assisted evolutionary algorithms (SAEAs) are gradually gaining attention as a method for solving expensive optimization problems with inequality constraints. Most SAEAs construct a surrogate model for each objective/constraint function and then aggregate approximation functions of constraints to estimate the feasibility of unevaluated solutions. However, because of the aggregation, the differences in the scales among constraints are ignored. Constraints with smaller scales do not benefit from constraint handling techniques as much as larger constraints, while the effects of handling constraints with larger scales scatter to the other many constraints. This results in an inefficient constraint optimization. Accordingly, this work proposes a new SAEA that partially optimizes each objective/constraint, namely surrogate-assisted partial optimization (SAPO). Solutions with better values of objective/constraint are selected from the evaluated solutions as the parent solutions and a focused objective/constraint is independently optimized using surrogate models one by one. Experimental results reveal the superiority of SAPO compared to the state-of-the-art SAEAs on a single-objective optimization problem suite with inequality constraints under an expensive optimization scenario.

Keywords: Surrogate-assisted Evolutionary Algorithm · Constrained Optimization Problem · Expensive Optimization Problem · Radial Basis Function Network · Differential Evolution

1 Introduction

Such as wind turbine structure optimization [19] and aerospace and automotive design [17], expensive constrained optimization problems (ECOPs) can often be seen in the real world. In ECOPs, the function evaluations (FEs) are computationally or financially expensive because time-consuming simulations or expensive physical experiments are used as FEs [11]. Hence, finding an optimal and feasible solution within a limited number of FEs is required when solving ECOPs. Expensive single-objective optimization problems with inequality constraints are focused in this work.

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 391–407, 2024.
https://doi.org/10.1007/978-3-031-70068-2_24

The formulation of an ECOP is as follows;

$$\begin{aligned} &\text{Minimize } f(\boldsymbol{x}), \\ &\quad \text{s.t. } g_m(\boldsymbol{x}) \leq 0,\ m = 1, 2, \ldots, M\ , \\ &\quad\quad x_j^l \leq \boldsymbol{x} \leq x_j^u,\ j = 1, 2, \ldots, D\ , \end{aligned} \tag{1}$$

where $\boldsymbol{x}$ is a solution, $f : \mathbb{R}^D \to \mathbb{R}$ is the expensive objective function, D is the number of decision variables, $g_m : \mathbb{R}^D \to \mathbb{R}$ is the m-th expensive constraint of M constraints, and x_j^l and x_j^u are the lower and upper values for the j-th decision variable, respectively. The feasibility of a solution is judged by the degree of constraint violation given by;

$$G(\boldsymbol{x}) = \sum_{m=1}^{M} \max\left(g_m(\boldsymbol{x}), 0\right). \tag{2}$$

The solution $\boldsymbol{x}$ is feasible when $G(\boldsymbol{x}) = 0$, otherwise $\boldsymbol{x}$ is infeasible.

Surrogate-assisted evolutionary algorithms (SAEAs) have begun to be applied to ECOPs [10]. SAEAs can save the consumption number of FEs by prescreening candidate solutions to be evaluated using surrogate models of objective/constraints made by machine learning (ML) techniques [22].

Most SAEAs construct surrogate models for the objective and each constraint in Eq. (1) and form a response surface set (RSS), $\left\{\hat{f}(\boldsymbol{x}), \hat{g}_1(\boldsymbol{x}), \hat{g}_2(\boldsymbol{x}), \ldots, \hat{g}_M(\boldsymbol{x})\right\}$ [30]. An RSS can determine the structure of each constraint and capture the characteristics of the feasible region boundaries [28]. In the early study on SAEAs with an RSS, Regis *et al.* [20,21,23] found the radial basis function network (RBFN) [18] can accurately approximate each in an RSS to some extent and work well with various optimizers. To obtain a more accurate RSS, ASAGA [24] and SACOBRA-MQcubic [1] adaptively select ML model types, e.g., RBFN and Kriging [15], and radial basis function (RBF) types in RBFN, respectively. Miranda-Varela and Mezura-Montes conducted an empirical study on constraint handling techniques using the original SA-DECV framework [16] with a k Nearest Neighbor [4,9]-based RSS. SACCDE [31] divides its population into two by the feasibility rule [5] as a constraint handling technique and selects solutions for the mutation strategy of differential evolution (DE) [25] from both groups to generate a variety of offspring solutions. An RSS is used to estimate the feasibility of offspring solutions. GLoSADE [28] constructs global and local surrogate models with RSSs and optimizes them with DE and the interior-point method, respectively. This global-local structure is also adopted in FMSADE [3] and SA-TSDE [14]. FMSADE employs the offspring generation method of SACCDE in the global search while SA-TSDE introduces a surrogate-based repair strategy to obtain a feasible solution using an RSS. MPMLS [12] constructs multiple local surrogate models with different penalty coefficients for the approximation of the degree of constraint violation calculated by an RSS. This mechanism contributes to maintaining a good population diversity [12].

Although these SAEAs create RSSs, prescreening of solutions is performed based on the approximation of the degree of constraint violation $\hat{G}(\boldsymbol{x})$, computed

in the manner of Eq. (2) as follows;

$$\hat{G}(\boldsymbol{x}) = \sum_{m=1}^{M} \max\left(\hat{g}_m(\boldsymbol{x}), 0\right). \tag{3}$$

Considering, however, $\hat{G}(\boldsymbol{x})$ is an aggregation of each $\hat{g}_m(\boldsymbol{x})$, the errors between $\hat{g}_m(\boldsymbol{x})$ and its true constraint $g_m(\boldsymbol{x})$ accumulate in $\hat{G}(\boldsymbol{x})$. What is worse, failing to account for the differences in scales between constraints hinders an effective handling of constraints. In other words, relying on only $\hat{G}(\boldsymbol{x})$ to estimate the feasibility of $\boldsymbol{x}$ lacks efficiency and robustness because the approximation accuracy and the effect of constraint handling techniques are highly affected by the scales of $\hat{g}_m(\boldsymbol{x})$ among different m. For example, the improvement of unfulfilled constraints with small scales is prevented by other constraints with large scales because optimization effects of $\hat{G}(\boldsymbol{x})$ less contribute to small-scaled constraints. On the other hand, constraints with larger scales converge slowly as the optimization effect scatters to the other many constraints. One way to tackle this challenge is the normalization of constraints like in SACOBRA [2]. However, even if the normalization is performed, handling only $\hat{G}(\boldsymbol{x})$ works well only when constraints are correlated with each other. Furthermore, in general, obtaining feasible solutions becomes more difficult as the problem dimension D increases [27]. Thereby, optimizing each $\hat{g}_m(\boldsymbol{x})$ with a small number of FEs becomes more important with the increase of D.

Accordingly, this work proposes an SAEA named surrogate-assisted partial optimization (SAPO), which optimizes each objective/constraint in turn. In other words, SAPO partially (independently) optimizes each objective/constraint while putting the other objective/constraints on hold. The RBFN is used to construct an RSS. Unlike existing SAEAs, SAPO prescreens candidate solutions in terms of each approximated constraint $\hat{g}_m(\boldsymbol{x})$ or objective $\hat{f}(\boldsymbol{x})$ in an RSS in place of $\hat{G}(\boldsymbol{x})$. SAPO can obtain solutions specialized for each $\hat{g}_m(\boldsymbol{x})$ or $\hat{f}(\boldsymbol{x})$ and thus each $g_m(\boldsymbol{x})$ or $\hat{f}(\boldsymbol{x})$ can be optimized effectively. To promote the entire optimization of the objective and constraints, SAPO selects solutions with better objective and constraint values to form parent solutions of DE. Thus, the solution diversity is kept high even if SAPO focuses on a certain objective/constraint and improves it. To the best of our knowledge, this is the first attempt to directly utilize the approximated constraint $\hat{g}_m(\boldsymbol{x})$ as the criterion of solution prescreening. The partial optimization of the objective and constraints can be a new constraint handling technique. This work empirically demonstrates that this new technique improves the performance of SAEAs on ECOPs because more feasible solutions are found with fewer FEs.

The remainder of this work is organized as follows. Section 2 introduces DE and RBFN as the component techniques of SAPO. Section 3 provides the design concept and the detailed algorithm of SAPO. Section 4 compares the performance of SAPO with state-of-the-art SAEAs on the CEC 2017 constrained real-parameter optimization benchmark suite [29] within an expensive scenario. In Sect. 5, we show the effectiveness of our partial optimization as an ablation study. Lastly, Sect. 6 concludes our work and discusses future directions.

2 Preliminary

This section introduces DE as the optimizer of the objective and constraints. Successively, RBFN as an ML technique for surrogate models is explained.

2.1 DE: Differential Evolution

DE is an evolutionary algorithm originally proposed for real-parameter bound-constrained single-objective optimization problems. The optimization process of DE consists of initialization, mutation, crossover, and solution selection, where procedures from mutation to solution selection are repeated till the termination.

DE initializes its population $\mathcal{P} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, where N is the population size. Specifically, DE samples each solution $\boldsymbol{x}_i = [x_{i,1}, \ldots, x_{i,D}]^\mathsf{T}$ using a uniform distribution within $x_{i,j} \in [x_j^l, x_j^u]$. The definition of x_j^l and x_j^u follows Eq. (1).

In the mutation procedure, a mutant solution $\boldsymbol{v}_i = [v_{i,1}, \cdots, v_{i,D}]^\mathsf{T}$ of each $\boldsymbol{x}_i$, i.e., a parent solution, is produced using an employed mutation strategy. To generate a variety of solutions, this work adopts two mutation strategies. The *rand/1* and *best/1* strategies contribute to exploration and exploitation, respectively. Definitions of these strategies are as follows;

$$\textit{rand/1}: \qquad \boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}), \tag{4}$$

$$\textit{best/1}: \qquad \boldsymbol{v}_i = \boldsymbol{x}_{best} + F(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}), \tag{5}$$

where $\boldsymbol{x}_{r_1}$, $\boldsymbol{x}_{r_2}$, and $\boldsymbol{x}_{r_3}$ are mutually exclusive solutions randomly selected from $\mathcal{P}$, also different from $\boldsymbol{x}_i$. The best solution $\boldsymbol{x}_{best}$ is a solution in $\mathcal{P}$ having the best fitness value. A scaling factor $F \in [0, 1]$ controls the contribution of differential vectors $(\boldsymbol{x}_r - \boldsymbol{x}_{r'})$.

Next, DE generates a trial vector $\boldsymbol{u}_i = [u_{i,1}, \cdots, u_{i,D}]^\mathsf{T}$ as an offspring solution via a crossover strategy and crossover rate $CR \in [0, 1]$. We use the most popular *binomial* crossover strategy given by;

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}, & \text{otherwise}, \end{cases} \tag{6}$$

where j_{rand} is an integer randomly selected from $[1, D]$ and $rand(0, 1)$ is a uniformly sampled real random value in $(0, 1)$.

Finally, $\boldsymbol{u}_i$ is evaluated and the solution for the next generation is selected, i.e., $\boldsymbol{x}_i$ is replaced with $\boldsymbol{u}_i$ if $f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i)$ for minimization problems.

2.2 RBFN: Radial Basis Function Network

RBFN is a feed-forward neural network with a three-layer structure. We employ the RBFN as an ML technique for the surrogate model because its construction or prediction time is relatively short and the model accuracy is scalable to the increase of problem dimension [7]. Let a training dataset (size n), vector of function values, and an RBF be $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^{n}$, $\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n)]^\mathsf{T}$,

and $\phi(r)$, respectively. The *cubic* RBF $\phi(r) = r^3$ is used as it can establish a fitness landscape that is more stable and exhibits improved convergence, thanks to its ability to avoid ill-conditioning [26]. Note that RBFN can also be used to approximate constraints by replacing $f(\boldsymbol{x})$ with $g_m(\boldsymbol{x})$.

The approximation of $f(\boldsymbol{x})$ for $\boldsymbol{x}$ can be formulated as follows;

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \lambda_i \phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|\right) + p(\boldsymbol{x}), \tag{7}$$

where $p(\boldsymbol{x}) = \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} + c_0$ ($\boldsymbol{c} \in \mathbb{R}^D, c_0 \in \mathbb{R}$) is regularization terms using a linear polynomial function, and $\lambda_i \in \mathbb{R}$ is the i-th element of the weight vector $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_n]^{\mathsf{T}}$.

Three parameters $\boldsymbol{\lambda}$, $\boldsymbol{c}$, and c_0 in Eq. (7) are obtained by solving the following equation;

$$\begin{bmatrix} \Phi & P \\ P^{\mathsf{T}} & 0_{\mathrm{m}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{c}' \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}, \tag{8}$$

where Φ is the $n \times n$ matrix with $\phi_{ij} = \phi\left(|\boldsymbol{x}_i - \boldsymbol{x}_j|\right)$, $P \in \mathbb{R}^{n \times (D+1)}$ is a matrix whose i-th row is $\left[1,\ \boldsymbol{x}_i^{\mathsf{T}}\right]$, 0_{m} is a $(D+1)\times(D+1)$ matrix of zeros, $\boldsymbol{c}' = \left[\boldsymbol{c}^{\mathsf{T}},\ c_0\right]^{\mathsf{T}}$, and $\boldsymbol{0}$ is a column vector of zeros whose length is $(D+1)$.

3 SAPO: Surrogate-Assisted Partial Optimization

This section starts by sharing the concept of SAPO and then explicates its algorithm.

3.1 Concept

Different from existing SAEAs for ECOPs where all constraints are dealt with in one bundle as the approximation of the degree of constraint violation, SAPO addresses an objective/constraint one by one, namely a partial optimization. This idea is inspired by the partial differential equation (PDE). Focusing on one element in PDE improves the efficiency of structure analysis or optimization [8,13]. That is, complex systems must be disassembled, and processed one by one.

Going back to ECOPs, the partial optimization has advantages below;

- Solutions suitable to an objective/constraint function can be screened by surrogate models. Compared to only using the approximation of the degree of constraint violation, this mechanism has a clearer intent to improve each function. Thus, the optimization efficiency for each function is enhanced.
- Solution diversity is kept high until the end of the search because SAPO has multiple criteria to prescreen offspring solutions.

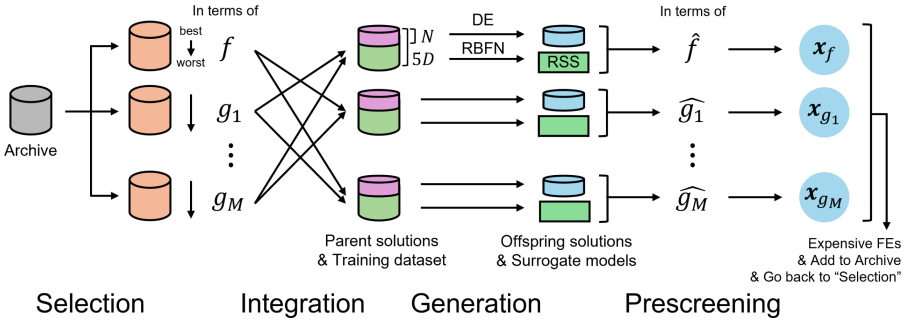


Fig. 1. A diagram of SAPO.

Algorithm 1. SAPO

Initialize the archive $\mathcal{A} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i), \{g_m(\boldsymbol{x})\}_{m=1}^{M})\}_{i=1}^{N_{init}}$, $FE = N_{init}$
while $FE < FE_{\max}$ **do**
  for all Target Function $\leftarrow \{f, g_1, g_2, \ldots, g_M\}$ **do**
    for all Selection Criterion $\leftarrow \{f, g_1, g_2, \ldots, g_M\} \setminus$ Target Function **do**
      $\mathcal{S} \leftarrow$ Algorithm 2 ($\mathcal{A}$, Selection Criterion, Target Function) // Selection
    end for
    $\mathcal{P}, \mathcal{D} \leftarrow N, 5D$ solutions from $\mathcal{S}$, respectively // Integration
    $\mathcal{U} \leftarrow$ Generate offspring from $\mathcal{P}$ with DE Eqs. (4–6) // Generation
    $\mathcal{R} \leftarrow$ Generate RSS using $\mathcal{D}$ with RBFN Eq. (7) // Generation
    $\boldsymbol{x}^* \leftarrow$ Algorithm 3 ($\mathcal{U}$, $\mathcal{R}$, Target Function) // Prescreening
    Evaluate $\boldsymbol{x}^*$ and add it to $\mathcal{A}$, $FE = FE + 1$
  end for
end while

– The partial optimization can handle the case where the scales among constraints are highly different. Unsatisfied constraints with relatively small scales are also focused while they are likely to be ignored when only the degree of constraint violation is used. Constraints with relatively large scales are also improved with fewer FEs because they are exclusively considered.

To fully utilize the partial optimization idea and the solution diversity mentioned above, parent solutions are selected from top solutions in terms of the objective/constraints that are not focused on now. Thus, SAPO can generate and prescreen offspring solutions that improve a focused objective/constraint from parent solutions with good values of the other objective/constraints.

3.2 Mechanism

Algorithm 1 provides the complete pseudocode of SAPO. The algorithm consists of three procedures; 1) Initialization, 2) Selection and Integration of evaluated solutions to obtain parent solutions and training dataset, and 3) Generation of offspring solutions and an RSS and Prescreening of offspring solutions. After

Algorithm 2. Selection

Input: Archive $\mathcal{A}$, Objective/constraint as the selection criterion f or $g_{m'}$, Target objective/constraint to be optimized f or g_m
Output: Selected and sorted set $\mathcal{S}$

if f is the target to be optimized **then**
  $\mathcal{T}_1 \leftarrow$ Sort feasible solutions in $\mathcal{A}$ in ascending order of f
  $\mathcal{T}_2 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \wedge g_{m'}(\boldsymbol{x}) \leq 0 \wedge f(\boldsymbol{x}) < f^*_{fea}\}$ in ascending order of $g_{m'}$
  $\mathcal{T}_3 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \wedge g_{m'}(\boldsymbol{x}) \leq 0 \wedge f(\boldsymbol{x}) \geq f^*_{fea}\}$ in ascending order of f
  $\mathcal{T}_4 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \wedge g_{m'}(\boldsymbol{x}) > 0 \wedge f(\boldsymbol{x}) < f^*_{fea}\}$ in ascending order of $g_{m'}$
  $\mathcal{T}_5 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \wedge g_{m'}(\boldsymbol{x}) > 0 \wedge f(\boldsymbol{x}) \geq f^*_{fea}\}$ in ascending order of f
else if g_m is the target to be optimized **then**
  $\mathcal{G} \leftarrow \{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{A} \wedge g_m(\boldsymbol{x}) > 0\}$ // $\boldsymbol{x}$ that satisfy $g_m(\boldsymbol{x})$ need not be optimized
  if f is the selection criterion **then**
    $\mathcal{T}_1 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \wedge f(\boldsymbol{x}) < f^*_{fea}\}$ in ascending order of g_m
    $\mathcal{T}_2 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \wedge f(\boldsymbol{x}) \geq f^*_{fea}\}$ in ascending order of f
  else if $g_{m'}$ is the selection criterion **then**
    $\mathcal{T}_1 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \wedge g_{m'}(\boldsymbol{x}) \leq 0 \wedge f(\boldsymbol{x}) < f^*_{fea}\}$ in ascending order of g_m
    $\mathcal{T}_2 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \wedge g_{m'}(\boldsymbol{x}) \leq 0 \wedge f(\boldsymbol{x}) \geq f^*_{fea}\}$ in ascending order of f
    $\mathcal{T}_3 \leftarrow$ Sort $\{\boldsymbol{x} | \ \boldsymbol{x} \in \mathcal{G} \wedge g_{m'}(\boldsymbol{x}) > 0\}$ in ascending order of $g_{m'}$
  end if
end if
$\mathcal{S} = [\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{last}]$

initialization, SAPO repeats 2) and 3) until the termination criteria are met. Figure 1 shows the diagram of 2) and 3). For each generation, SAPO sets a target objective/constraint to be optimized and this sequentially changes to the next one, i.e., in the order of $f, g_1, g_2, \ldots, g_M, f, g_1, \ldots$ and so on.

Initialization. Latin hypercube sampling is employed to sample N_{init} points. These points are evaluated with the objective and constraint functions. SAPO creates an archive $\mathcal{A} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i), \{g_m(\boldsymbol{x}_i)\}_{m=1}^{M})\}_{i=1}^{N_{init}}$ to store all evaluated solutions and their objective/constraint values.

Selection and Integration. This phase intends to extract good solutions from the archive to prepare parent solutions and a training dataset for an RSS. Here, we define f^*_{fea} as the best fitness value among feasible solutions in $\mathcal{A}$. If there are no feasible solutions in $\mathcal{A}$, f^*_{fea} is set to ∞. Solutions satisfying $f(\boldsymbol{x}) < f^*_{fea}$ are infeasible but may be useful in the optimization if their constraints are improved. Thus these solutions are utilized below.

First, SAPO makes M sets of selected and sorted solutions, where M is the number of constraints. Each set is generated corresponding to each objective/constraint except for the target objective/constraint. Note that each solution in $\mathcal{A}$ can be selected multiple times. The selecting and sorting method is summarized in Algorithm 2.

Algorithm 3. Prescreening

Input: Offspring solutions $\mathcal{U} = \{\boldsymbol{x}_i\}_{i=1}^{|\mathcal{U}|}$, RSS $\mathcal{R}$, Target objective/constraint f or g_m
Output: Selected solution $\boldsymbol{x}^*$

1: **if** f is the target **then**
2: Calculate $\hat{G}(\boldsymbol{x})$ values of $\forall \boldsymbol{x} \in \mathcal{U}$ by Eq. (3) using $\mathcal{R}$
3: $\mathcal{F} \leftarrow \{\boldsymbol{x} |\ \boldsymbol{x} \in \mathcal{U} \wedge \hat{G}(\boldsymbol{x}) = 0\}$ // $\boldsymbol{x}$ estimated to be feasible
4: $\boldsymbol{x}^* = \begin{cases} \underset{x \in \mathcal{F}}{\arg\ \min}\, \hat{f}(\boldsymbol{x}), & \mathcal{F} \neq \emptyset \\ \underset{x \in \mathcal{U}}{\arg\ \min}\, \hat{G}(\boldsymbol{x}), & \text{otherwise} \end{cases}$
5: **else if** g_m is the target **then**
6: $\mathcal{G} \leftarrow \{\boldsymbol{x} |\ \boldsymbol{x} \in \mathcal{U} \wedge \hat{g}_m(\boldsymbol{x}) \leq 0\}$ // $\boldsymbol{x}$ that satisfy $\hat{g}_m(\boldsymbol{x})$
7: $\boldsymbol{x}^* = \begin{cases} \underset{x \in \mathcal{G}}{\arg\ \min}\, \hat{f}(\boldsymbol{x}), & \mathcal{G} \neq \emptyset \\ \underset{x \in \mathcal{U}}{\arg\ \min}\, \hat{g}_m(\boldsymbol{x}), & \text{otherwise} \end{cases}$
8: **end if**

– When f is the target to be optimized, SAPO selects and sorts solutions for each $g_{m'}$ as Lines 1–6 of Algorithm 2. Let $g_{m'}$ be a focused constraint as the selection criterion in this phase. First, feasible solutions are selected from $\mathcal{A}$ and sorted in ascending order of f and form $\mathcal{T}_1$. Next, $\boldsymbol{x}$s that satisfy $g_{m'}$ are corrected. Among these solutions, ones that give $f(\boldsymbol{x}) < f^*_{fea}$ are preferred as mentioned before. As $g_{m'}$ is the criterion now, they are sorted with the $g_{m'}$ values, forming $\mathcal{T}_2$. Then, $\boldsymbol{x}$s that satisfy $g_{m'}$ but give $f(\boldsymbol{x}) \geq f^*_{fea}$ are used. As they are inferior to $\boldsymbol{x} \in \mathcal{T}_2$, the sorting order of $\mathcal{T}_3$ follows f values. Similar procedures are performed to make $\mathcal{T}_4$ and $\mathcal{T}_5$ for $\boldsymbol{x}$s that violate $g_{m'}$. Finally, SAPO returns a set $\mathcal{S}$ by joining $\mathcal{T}_1, \mathcal{T}_2, \ldots,$ and $\mathcal{T}_5$ in this order.
– When g_m is the target to be optimized, solutions that already satisfy g_m are ignored as they need not be optimized anymore, as shown in Line 8. Solutions that violate g_m are selected from $\mathcal{A}$ and consist of $\mathcal{G}$. The remained procedures are as Lines 9–15. If f is the selection criterion, solutions in $\mathcal{G}$ and $f(\boldsymbol{x}) < f^*_{fea}$ are selected to utilize the possibility of improvement, forming $\mathcal{T}_1$. However, sorting order follows the ascending order of the target constraint g_m as solutions in $\mathcal{T}_1$ lack feasibility. Next, solutions with $f(\boldsymbol{x}) \geq f^*_{fea}$ form $\mathcal{T}_2$ where solutions are ordered with f values. An output set is $\mathcal{S} = [\mathcal{T}_1, \mathcal{T}_2]$. On the other hand, $g_{m'}$ is taken into account when $g_{m'}$ is the selection criterion. The fulfillment of $g_{m'}$ has the first priority. However, as $g_{m'} \leq 0$ is enough, the next priority becomes comparison with f^*_{fea}. Similarly, $\mathcal{T}_1$ and $\mathcal{T}_2$ are obtained. Finally, solutions that violate $g_{m'}$ are selected and sorted In increasing order of the degree of violation of $g_{m'}$, forming $\mathcal{S} = [\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3]$.

After gaining M sets of sorted solutions, they are integrated as follows. The elements of each set are taken in order from the first one, let this be the parent solution set $\mathcal{P}$ when there are N elements, and let this be the training dataset $\mathcal{D}$ for constructing an RSS when its size becomes $5D$.

Generation and Prescreening. This phase begins by generating offspring solutions and constructing an RSS. For the evolution of $\mathcal{P}$, the mutation strategies shown in Eqs. (4–5) and the crossover strategy in Eq. (6) are applied to $\mathcal{P}$. Note that the size of the offspring solution set $\mathcal{U}$ becomes double that of $\mathcal{P}$, i.e., $2N$. To generate a variety of offspring solutions, $\mathcal{P}$ is copied, and two mutation strategies are independently used for each $\mathcal{P}$. Successively, an RSS is constructed using RBFNs and $\mathcal{D}$ as presented in Eq. (7).

Finally, SAPO prescreens $\mathcal{U}$ in terms of the target objective/constraint. The detailed procedures are indicated in Algorithm 3. When f is the target, the feasibility rule is employed. Specifically, $\hat{G}(\boldsymbol{x})$ calculated by Eq. (3) using an RSS estimates the feasibility of solutions in $\mathcal{U}$. If there are expected to be feasible solutions, the solution having the minimum $\hat{f}(\boldsymbol{x})$ is selected. Otherwise, the solution having the minimum $\hat{G}(\boldsymbol{x})$ is chosen. When g_m is the target, however, solutions that satisfy $\hat{g}_m(\boldsymbol{x})$ are esteemed. Among them, $\boldsymbol{x}$ having the minimum $\hat{f}(\boldsymbol{x})$ is selected, denoting $\boldsymbol{x}^*$. If no solution fulfills $\hat{g}_m(\boldsymbol{x})$, the solution with the least violation of $\hat{g}_m(\boldsymbol{x})$ is selected. The selected solution is evaluated with the expensive function, and it and its objective/constraint values are added to $\mathcal{A}$.

4 Experiment

Through a comparison of performances between SAPO and SAEAs for ECOPs, we evaluate the effectiveness of SAPO.

4.1 Experimental Design

We use the IEEE CEC 2017 constrained real-parameter optimization benchmark suite [29]. Nine problems with inequality constraints are selected from the suite, i.e., F1, F2, F4, F5, F12, F13, F20, F21, and F22. Note that F19 and F28 are excluded although they are problems with inequality constraints as they have no feasible solutions according to the problem definition [29]. The problem dimensions are set to $D \in \{30, 50, 100\}$ to evaluate the scalability of the performance of SAPO against the increase of D. The other settings follow the regulation of competition [29]. All experiments are done with Intel(R) Core(TM) i7-10700 (2.90 GHz) CPU and 16 GB RAM.

Four state-of-the-art SAEAs, GLoSADE [28], FMSADE [3], MPMLS [12], and SA-TSDE [14], are employed for the compared algorithms. All algorithms use DE and RBFN, so we can fairly compare the performances. Note that GLoSADE, FMSADE, and SA-TSDE adopt the interior point algorithm for the local search. While these three SAEAs utilize global and local searches, MPMLS decomposes the approximation of the degree of constraint violation $\hat{G}(\boldsymbol{x})$ using penalty coefficients. MPMLS and SAPO construct only local surrogate models. Consequently, this work can investigate the impact of differences among global/local structures and the deals of approximated constraints on the performance. Hyperparameter settings of the compared algorithms follow the original papers [3,12,14,28]. For SAPO, we set to $N_{init} = 100$ for $D \in \{30, 50\}$ and 200 for $D = 100$ so that

$N_{init} > D$. We also use $N = 100$, $F = 0.5$, and $CR = 0.9$, following the original paper of DE [25], and $kernel = cubic$, one of the most popular RBFN kernels.

Following the original papers [3,12,14,28], the maximum number of FEs is set to $3,000$. The performance is evaluated with the average fitness values of feasible solutions obtained in 31 independent runs for each problem and average ranks over nine problems. We apply the Wilcoxon rank-sum test with a significance level of 0.05 to check statistical significance. When reporting statistical results, we use "+", "−", or "∼", indicating the compared algorithm significantly outperformed SAPO, significantly underperformed SAPO, or we cannot decide that there is a significant difference, respectively. In case no feasible solution is obtained within the observed number of FEs, a certain enough large fitness value (1E+20) is assigned to the corresponding runs when computing statistical test results and average ranks.

4.2 Result

Table 1 summarizes the average fitness values obtained at $3,000$ FEs for $D \in \{30, 50, 100\}$. SAPO derived six, six, and four best performances out of nine problems in the order of $D = 30$, 50, and 100 and no worst performance. This shows the robustness of SAPO over different problems, except for those on which all algorithms failed in finding feasible solutions as shown in gray. This means the partial optimization methodology proposed in this work contributed to steadily improving the objective value while satisfying constraints on many types of problems. The average ranks shown at the bottom of Table 1 also demonstrate the superiority of SAPO. Although the average rank slightly degrades as D increases, each average rank is the best among the five algorithms for all D. Thus, the performance of SAPO scales to the increase of problem dimension.

From a statistical point of view, the number of "+" indicating the superiority of the compared algorithms is zero or one in all comparison pairs for all D. The number of "−" indicating the inferiority of the compared algorithms is four and seven at least and at most, respectively. Accordingly, the number of "−" is much larger than that of "+" in all comparisons, where the maximum difference is seven out of nine problems. The total results of 108 comparisons across the three types of dimensions and the four compared algorithms are $+/-/\sim = 3/67/38$. These results clearly indicate the outstanding performance of SAPO.

It is worth noting that SAPO derived a much larger number of successful runs. For example, SAPO succeeded in finding feasible solutions in all runs on F12 ($D \in \{30, 50\}$) and F21 ($D = 30$) although some or all of the other algorithms failed. Even if D increased and the difficulty of finding feasible solutions became higher, SAPO obtained multiple successful runs on the same problems, i.e., F12 ($D = 100$) and F21 ($D = 50$), while the other algorithms could not succeed in any run. MPMLS is the best algorithm except SAPO in terms of finding feasible solutions with better objective values, e.g., on F12 and F20. This may be because MPMLS employs a decomposition strategy of $\hat{G}(\boldsymbol{x})$, unlike other SAEAs. However, SAPO contributed to obtaining more feasible solutions with better objective value on problems not only with one constraint but also with

Table 1. The average fitness values at 3,000 function evaluations for $D \in \{30, 50, 100\}$. The integers next to the problem name in brackets indicate the number of constraints. The best and worst results among the five algorithms are highlighted in green bold and pink italic, respectively. When some of the 31 runs failed in obtaining feasible solutions, the number of successful runs, i.e., cases where an algorithm found at least one feasible solution, is noted in brackets. When all algorithms could not find feasible solutions, corresponding cells are highlighted in gray. Statistical test result "+", "−", or "∼" indicates the compared algorithm significantly outperformed SAPO, significantly underperformed SAPO, or we cannot see a significant difference, respectively.

a) $D = 30$

Problem (# cons)	GLoSADE	FMSADE	MPMLS	SA-TSDE	SAPO
F1 (1)	9.557e+03 −	*4.164e+04* −	2.499e+04 −	7.389e+03 −	**5.400e+03**
F2 (1)	4.032e+03 −	*(1)* −	5.468e+03 −	3.070e+03 −	**2.110e+03**
F4 (2)	3.883e+02 −	*4.060e+02* −	1.973e+02 ∼	2.148e+02 −	**1.888e+02**
F5 (2)	4.178e+01 −	*3.183e+02* −	**2.895e+01** ∼	5.707e+01 −	3.251e+01
F12 (2)	1.517e+02 −	*(0)* −	1.559e+01 −	(30) −	**1.261e+01**
F13 (3)	*(0)* −	*(0)* −	*(0)* −	**(7)** ∼	**(7)**
F20 (2)	9.801e+00 ∼	*1.002e+01* ∼	**9.589e+00** +	9.920e+00 ∼	9.877e+00
F21 (2)	*(0)* −	*(0)* −	(29) −	*(0)* −	**1.075e+01**
F22 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
+/−/∼	0/**7**/2	0/**7**/2	1/**5**/3	0/**6**/3	–
Ave. Rank	3.222	*4.556*	2.556	3.056	**1.611**

b) $D = 50$

Problem (# cons)	GLoSADE	FMSADE	MPMLS	SA-TSDE	SAPO
F1 (1)	3.038e+04 −	*1.064e+05* −	6.341e+04 −	3.852e+04 −	**2.547e+04**
F2 (1)	1.707e+04 −	*(0)* −	2.087e+04 −	1.487e+04 −	**1.056e+04**
F4 (2)	*6.819e+02* −	6.686e+02 −	4.574e+02 −	3.507e+02 −	**3.100e+02**
F5 (2)	2.444e+03 −	*(27)* −	1.210e+02 −	1.879e+03 −	**9.113e+01**
F12 (2)	*(0)* −	*(0)* −	1.052e+02 −	*(0)* −	**1.429e+01**
F13 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
F20 (2)	1.824e+01 ∼	*1.882e+01* ∼	**1.803e+01** ∼	1.860e+01 ∼	1.838e+01
F21 (2)	*(0)* −	*(0)* −	*(0)* −	*(0)* −	**(30)**
F22 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
+/−/∼	0/**6**/3	0/**6**/3	0/**6**/3	0/**6**/3	–
Ave. Rank	3.278	*4.167*	2.833	3.056	**1.667**

c) $D = 100$

Problem (# cons)	GLoSADE	FMSADE	MPMLS	SA-TSDE	SAPO
F1 (1)	1.644e+05 ∼	*4.964e+05* −	2.352e+05 −	**1.597e+05** ∼	1.624e+05
F2 (1)	1.314e+05 −	*(0)* −	9.185e+04 −	9.066e+04 −	**7.763e+04**
F4 (2)	1.491e+03 −	*1.523e+03* −	1.257e+03 −	8.659e+02 −	**7.750e+02**
F5 (2)	4.271e+04 −	*(25)* −	2.094e+03 −	2.385e+04 −	**1.384e+03**
F12 (2)	*(0)* −	*(0)* −	*(0)* −	*(0)* −	**(19)**
F13 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
F20 (2)	4.007e+01 +	4.067e+01 ∼	**3.969e+01** +	*4.113e+01* ∼	4.093e+01
F21 (2)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
F22 (3)	(0) ∼	(0) ∼	(0) ∼	(0) ∼	(0)
+/−/∼	1/**4**/4	0/**5**/4	1/**5**/3	0/**4**/5	–
Ave. Rank	3.278	*3.944*	2.833	2.833	**2.111**

Table 2. Significant differences regarding findings for "+/ − / ∼" between SAPO and state-of-the-art SAEAs. Statistical test result "+", "−", or "∼" indicates the compared algorithm significantly outperformed SAPO, significantly underperformed SAPO, or we cannot see a significant difference, respectively. In the comparisons between the numbers of "+" and "−", the larger numbers are highlighted in bold.

D	FE	GLoSADE	FMSADE	MPMLS	SA-TSDE	D	FE	GLoSADE	FMSADE	MPMLS	SA-TSDE
30	300	0/**3**/6	0/**4**/5	0/**3**/6	**1**/**1**/7	100	300	0/ **2**/7	0/ **3**/6	0/ **1**/ 8	**2**/ **2**/ 5
	500	0/**5**/4	0/**5**/4	0/**3**/6	**2**/1/6		500	1/ **2**/6	0/ **3**/6	1/ **2**/ 6	**2**/ **2**/ 5
	1,000	0/**6**/3	0/**6**/3	0/**5**/4	1/**3**/5		1,000	1/ **3**/5	0/ **4**/5	1/ **2**/ 6	**2**/ **2**/ 5
	2,000	0/**7**/2	0/**7**/2	1/**7**/1	0/**5**/4		2,000	0/ **3**/6	0/ **4**/5	1/ **3**/ 5	1/ **2**/ 6
	3,000	0/**7**/2	0/**7**/2	1/**5**/3	0/**6**/3		3,000	1/ **4**/4	0/ **5**/4	1/ **5**/ 3	0/ **4**/ 5
50	300	0/**3**/6	1/**3**/5	0/**3**/6	**2**/**2**/5	Total	300	0/ **8**/19	1/**10**/16	0/ **7**/20	**5**/ **5**/17
	500	0/**3**/6	0/**4**/5	0/**3**/6	**2**/1/6		500	1/**10**/16	0/**12**/15	1/ **8**/18	**6**/ 4/17
	1,000	0/**5**/4	0/**5**/4	0/**5**/4	**2**/**2**/5		1,000	1/**14**/12	0/**15**/12	1/**12**/14	5/ **7**/15
	2,000	0/**6**/3	0/**7**/2	0/**6**/3	0/**5**/4		2,000	0/**16**/11	0/**18**/ 9	2/**16**/ 9	1/**12**/14
	3,000	0/**6**/3	0/**6**/3	0/**6**/3	0/**6**/3		3,000	1/**17**/ 9	0/**18**/ 9	2/**16**/ 9	0/**16**/11

multiple constraints. These results demonstrate the effectiveness of the partial optimization methodology of SAPO, where each objective and constraint were optimized directly and thus improved effectively. This methodology was also useful for non-separable and rotated constraints like F2 and F5 as SAPO derived good performances on them. Even if decision variables cannot be separated, the objective and constraints can be optimized independently.

Furthermore, we show the results of Wilcoxon rank-sum tests at 300, 500, 1,000, and 2,000 FEs in addition to 3,000 (baseline) FEs for $D \in \{30, 50, 100\}$ and their total number in Table 2 to evaluate the convergence performance of SAPO. SAPO outperformed GLoSADE, FMSADE, and MPMLS on every number of FEs in the table, where the number of "+" indicating the superiority of the compared SAEAs is at most one. Although SAPO was competitive with SA-TSDE before 1,000 FEs, "−" outnumbers "+" after 2,000 FEs, indicating that SAPO kept improving the objective/constraint towards 3,000 FEs. This tendency can be observed in every D and their total. Therefore, we can confirm the scalability of the performance of SAPO to the increase in the number of FEs.

5 Discussion

5.1 Impact of the Partial Optimization

Unlike existing SAEAs where constraints are treated as only an aggregation of constraints, SAPO partially optimizes each objective/constraint. This subsection investigates the effectiveness of the partial optimization proposed in this work. Here, we prepared three variants of SAPO.

1. **Variant using Aggregation (VUA)** This variant is the most similar to the mechanism of existing SAEAs, which employs the feasibility rule [5] using aggregation of constraints, instead of partial optimization. The feasibility

rule is the representative constraint handling technique [12]. Specifically, this variant selects and sorts solutions for parent solutions and the training dataset of an RSS by Eq. (2) every time. Feasible solutions are preferentially selected and sorted in ascending order of $f(\boldsymbol{x})$. Then, infeasible solutions are sorted in ascending order of $G(\boldsymbol{x})$. DE offspring solutions and an RSS are generated in the same manner as those of Sect. 3.2. Offspring solutions are prescreened as Lines 2–4 in Algorithm 3. Comparison with this variant reveals the impact of partial optimization.

2. **Variant Targeting Objective (VTO)** This variant sets only the objective function as the target. Thus, solutions for DE parent solutions and the training dataset are selected and sorted from good constraints. Specifically, only Lines 1–6 and Lines 1–4 are conducted in Algorithms 2 and 3, respectively.
3. **Variant Targeting Constraints (VTC)** This variant sets only the constraints as the target. Hence, M constraints are partially optimized, but solutions are not screened to improve $f(\boldsymbol{x})$. Only Lines 9–15 and Lines 5–7 are conducted in Algorithms 2 and 3, respectively. VTO and VTC are prepared to evaluate whether both the objective and constraints are needed or not as the targets of the partial optimization.

The experimental design is similar to Sect. 4.1. Table 3 summarizes the results of the Wilcoxon rank-sum test. Statistical sign "+", "−", or "∼" indicates the variant significantly outperformed SAPO, significantly underperformed SAPO, or we cannot say that there is a significant difference, respectively. In comparison with VUA, SAPO outperformed VUA at $D = 30$. Specifically, the number of "+", indicating the superiority of VUA, is larger than that of "−", indicating the inferiority of VUA, with 300 FEs. As the feasibility rule devotes many resources to feasible solutions [12], the objective value is improved in the very early phase of the search once feasible solutions are obtained. However, in problems where obtaining feasible solutions is difficult, the use of the constraint aggregation $G(\boldsymbol{x})$ or its approximation $\hat{G}(\boldsymbol{x})$ requires more FEs to improve constraint violation in the feasibility rule. Thus, the performance of VUA stagnated and the number of "−" outnumbers that of "+" after $1,000$ FEs. The same tendency is observed for $D \in \{50, 100\}$. Although the difference in the number of "−" and "+" decreases as D increases, SAPO outperformed VUA in total of all D. Thus, the effectiveness of the partial optimization is confirmed.

In the comparison between SAPO and VTO, the performance of SAPO is slightly better than that of VTO. A similar trend to the comparison between SAPO and VUA is detected; the performance of VTO stagnated. Thus, partial optimization of constraints is needed to keep improving the performance. On the other hand, SAPO clearly outperformed VTC for all D. This indicates the improvement of the objective values is necessary although the partial optimization of constraints contributes to the improvement of each constraint. From these two comparisons, we identified that both the objective and constraints should be dealt with as the targets of the partial optimization.

5.2 Impact of the Parallel Use of DE Mutation Strategies

SAPO uses both *rand/1* and *best/1* mutation strategies to produce a variety of offspring solutions. This subsection evaluates the effectiveness of the parallel use of two DE mutation strategies. We prepared two variants of SAPO; one uses only *rand/1* and the other uses only *best/1*. Note that these variants generate $2N$ offspring for fair comparison. Again, the experimental design is similar to Sect. 4.1. Table 4 summarizes the results of the Wilcoxon rank-sum test. Statistical test sign "+", "−", or "∼" denotes the variant significantly outperformed SAPO, significantly underperformed SAPO, or we cannot determine that there is a significant difference, respectively. From the table, SAPO outperformed the variant with *rand/1* as no "+". indicating the superiority of the variant, are observed. The single-use of *rand/1* lacks the exploitation ability. The variant with *best/1* derived slightly better performance than SAPO in the early stage of optimization while SAPO became slightly better at the end of the search. This indicates the strong exploitation ability of *best/1* but the diversity of offspring solution should be maintained by adding *rand/1*.

Table 3. Significant differences regarding findings for "+/ − / ∼" between SAPO and its variants for an ablation study on partial optimization.

	$D = 30$			$D = 50$			$D = 100$			Total		
FE	VUA	VTO	VTC	VUA	VTO	VTC	VUA	VTO	VTC	VUA	VTO	VTC
300	**3**/1/5	**3**/0/6	1/**4**/4	**2**/0/7	**3**/0/6	0/**3**/6	**2**/0/7	**4**/0/5	0/**4**/5	**7**/1/19	**10**/0/17	1/**11**/15
500	1/**3**/5	**3**/0/6	0/**8**/1	**5**/0/4	**5**/0/4	1/**4**/4	**4**/0/5	**4**/0/5	0/**4**/5	**10**/3/14	**12**/0/15	1/**16**/10
1,000	0/**5**/4	0/**5**/4	0/**8**/1	**4**/0/5	**4**/0/5	0/**6**/3	**6**/0/3	**4**/0/5	0/**6**/3	**10**/5/12	**8**/5/14	0/**20**/7
2,000	0/**5**/4	0/**4**/5	0/**8**/1	**2**/1/6	**2**/**2**/5	0/**7**/2	**2**/1/6	**2**/1/6	0/**7**/2	4/**7**/16	4/**7**/16	0/**22**/5
3,000	0/**5**/4	0/**3**/6	0/**8**/1	0/**3**/6	0/**3**/6	0/**6**/3	1/**3**/5	1/**2**/6	0/**6**/3	1/**11**/15	1/**8**/18	0/**20**/7

Table 4. Significant differences regarding findings for "+/ − / ∼" between SAPO and its variants for an ablation study on the DE mutation strategies.

	$D = 30$		$D = 50$		$D = 100$		Total	
FE	*rand/1*	*best/1*	*rand/1*	*best/1*	*rand/1*	*best/1*	*rand/1*	*best/1*
300	0/**4**/5	0/**1**/8	0/**3**/6	**2**/0/7	0/**2**/7	**1**/0/8	0/**9**/18	**3**/1/23
500	0/**5**/4	0/**2**/7	0/**3**/6	**2**/1/6	0/**3**/6	**0**/**0**/9	0/**11**/16	**2**/3/22
1,000	0/**6**/3	**1**/**1**/7	0/**5**/4	**2**/1/6	0/**4**/5	0/**2**/7	0/**15**/12	**3**/4/20
2,000	0/**7**/2	0/**1**/8	0/**6**/3	**1**/**1**/7	0/**4**/5	0/**2**/7	0/**17**/10	1/**4**/22
3,000	0/**6**/3	0/**3**/6	0/**6**/3	1/**3**/5	0/**5**/4	1/**3**/5	0/**17**/10	2/**9**/16

6 Conclusion

This work proposed an SAEA named surrogate-assisted partial optimization (SAPO). SAPO selects and sorts solutions with good objective/constraint values to form parent solutions and then independently optimizes each objective/constraint one by one. In the experiment, SAPO derived significantly better performance than existing SAEAs as SAPO dealt with each constraint effectively. In the discussion, we showed our partial optimization methodology can find feasible solutions with better objective values within a smaller number of FEs than using only an approximation of constraint violation made by aggregation of constraint approximations, which is commonly used in existing SAEAs.

Future work includes an adaptive selection of the objective/constraint to be optimized to improve the optimization efficiency. We are motivated to solve the entire problem set of the CEC 2017 benchmark suite, including problems with equality constraints. We will also extend SAPO for multi-objective ECOPs, where surrogate models are constructed for independent objectives/constraints or an aggregated function of all objectives/constraints [6].

Acknowledgments. This work was supported by JSPS KAKENHI under Grant No. 22KJ1409.

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

References

1. Bagheri, S., Konen, W., Bäck, T.: Online selection of surrogate models for constrained black-box optimization. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI) pp. 1–8. IEEE (2016). https://doi.org/10.1109/SSCI.2016.7850206
2. Bagheri, S., Konen, W., Emmerich, M., Bäck, T.: Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. Appl. Soft Comput. **61**, 377–393 (2017). https://doi.org/10.1016/j.asoc.2017.07.060
3. Chu, S., Yang, Z., Xiao, M., Qiu, H., Gao, K., Gao, L.: Explicit topology optimization of novel polyline-based core sandwich structures using surrogate-assisted evolutionary algorithm. Comput. Methods Appl. Mech. Eng. **369**, 113215 (2020). https://doi.org/10.1016/j.cma.2020.113215
4. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967). https://doi.org/10.1109/TIT.1967.1053964
5. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**(2), 311–338 (2000). https://doi.org/10.1016/S0045-7825(99)00389-8
6. Deb, K., Roy, P.C., Hussein, R.: Surrogate Modeling Approaches for Multiobjective Optimization: Methods, Taxonomy, and Results. Math. Comput. Appl. **26**(1), 5 (2020). https://doi.org/10.3390/mca26010005
7. Díaz-Manríquez, A., Toscano, G., Coello Coello, C.A.: Comparison of metamodeling techniques in evolutionary algorithms. Soft. Comput. **21**(19), 5647–5663 (2017). https://doi.org/10.1007/s00500-016-2140-z

8. Evans, L.C.: Partial Differential Equations. American Mathematical Society (Mar 2022)
9. Fix, E., Hodges, J.L.: Discriminatory analysis - nonparametric discrimination: small sample performance. Tech. Rep. ADA800391, University of California, Berkeley (1952)
10. He, C., Zhang, Y., Gong, D., Ji, X.: A review of surrogate-assisted evolutionary algorithms for expensive optimization problems. Expert Syst. Appl. **217**, 119495 (2023). https://doi.org/10.1016/j.eswa.2022.119495
11. Jin, Y.: Surrogate-assisted evolutionary computation: recent advances and future challenges. Swarm Evol. Comput. **1**(2), 61–70 (2011). https://doi.org/10.1016/j.swevo.2011.05.001
12. Li, G., Zhang, Q.: Multiple penalties and multiple local surrogates for expensive constrained optimization. IEEE Trans. Evol. Comput. **25**(4), 769–778 (2021). https://doi.org/10.1109/TEVC.2021.3066606
13. Liu, R., Bianco, M.J., Gerstoft, P.: Automated partial differential equation identification. J. Acoust. Soc. Am. **150**(4), 2364 (2021). https://doi.org/10.1121/10.0006444
14. Liu, Y., Liu, J., Jin, Y., Li, F., Zheng, T.: A surrogate-assisted two-stage differential evolution for expensive constrained optimization. IEEE Trans. Emerg. Topics Comput. **7**(3), 715–730 (2023). https://doi.org/10.1109/TETCI.2023.3240221
15. Lophaven, S.N., Nielsen, H.B., Søndergaard, J.: DACE: a MATLAB kriging toolbox. Tech. Rep. IMM-REP-2002-12, Informatics and Mathematical Modelling, DTU (2002)
16. Miranda-Varela, M.E., Mezura-Montes, E.: Constraint-handling techniques in surrogate-assisted evolutionary optimization. An empirical study. Appl. Soft Comput. **73**, 215–229 (2018). https://doi.org/10.1016/j.asoc.2018.08.016
17. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA J. **41**(4), 687–696 (2003). https://doi.org/10.2514/2.1999
18. Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural Comput. **3**(2), 246–257 (1991). https://doi.org/10.1162/neco.1991.3.2.246
19. Preen, R.J., Bull, L.: Toward the coevolution of novel vertical-axis wind turbines. IEEE Trans. Evol. Comput. **19**(2), 284–294 (2015). https://doi.org/10.1109/TEVC.2014.2316199
20. Regis, R.G.: Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. Comput. Oper. Res. (2011). https://doi.org/10.1016/j.cor.2010.09.013
21. Regis, R.G.: Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions. IEEE Trans. Evol. Comput. **18**(3), 326–347 (2014). https://doi.org/10.1109/TEVC.2013.2262111
22. Regis, R.G.: A survey of surrogate approaches for expensive constrained black-box optimization. In: Le Thi, H.A., Le, H.M., Pham Dinh, T. (eds.) WCGO 2019. AISC, vol. 991, pp. 37–47. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-21803-4_4
23. Regis, R.G., Shoemaker, C.A.: Constrained global optimization of expensive black box functions using radial basis functions. J. Global Optimiz. **31**(1), 153–171 (2005). https://doi.org/10.1007/s10898-004-0570-0

24. Shi, L., Rasheed, K.: ASAGA: an adaptive surrogate-assisted genetic algorithm. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 1049–1056. GECCO 2008, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1389095.1389289
25. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimiz. **11**, 341–359 (1997). https://doi.org/10.1023/A:1008202821328
26. Wang, W., Liu, H.L., Tan, K.C.: A surrogate-assisted differential evolution algorithm for high-dimensional expensive optimization problems. IEEE Trans. Cybern. **53**(4), 2685–2697 (2023). https://doi.org/10.1109/TCYB.2022.3175533
27. Wang, Y., Li, J.P., Xue, X., Wang, B.C.: Utilizing the correlation between constraints and objective function for constrained evolutionary optimization. IEEE Trans. Evol. Comput. **24**(1), 29–43 (2020). https://doi.org/10.1109/TEVC.2019.2904900
28. Wang, Y., Yin, D.Q., Yang, S., Sun, G.: Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints. IEEE Trans. Cybern. **49**(5), 1642–1656 (2019). https://doi.org/10.1109/TCYB.2018.2809430
29. Wu, G., Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. Tech. rep., National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report, Singapore (2017)
30. Wu, Y., Yin, Q., Jie, H., Wang, B., Zhao, J.: A RBF-based constrained global optimization algorithm for problems with computationally expensive objective and constraints. Struct. Multidiscip. Optim. **58**(4), 1633–1655 (2018). https://doi.org/10.1007/s00158-018-1987-2
31. Yang, Z., Qiu, H., Gao, L., Cai, X., Jiang, C., Chen, L.: Surrogate-assisted classification-collaboration differential evolution for expensive constrained optimization problems. Inf. Sci. **508**, 50–63 (2020). https://doi.org/10.1016/j.ins.2019.08.054

Author Index

M. Affenzeller et al. (Eds.): PPSN 2024, LNCS 15149, pp. 409–410, 2024.
https://doi.org/10.1007/978-3-031-70068-2