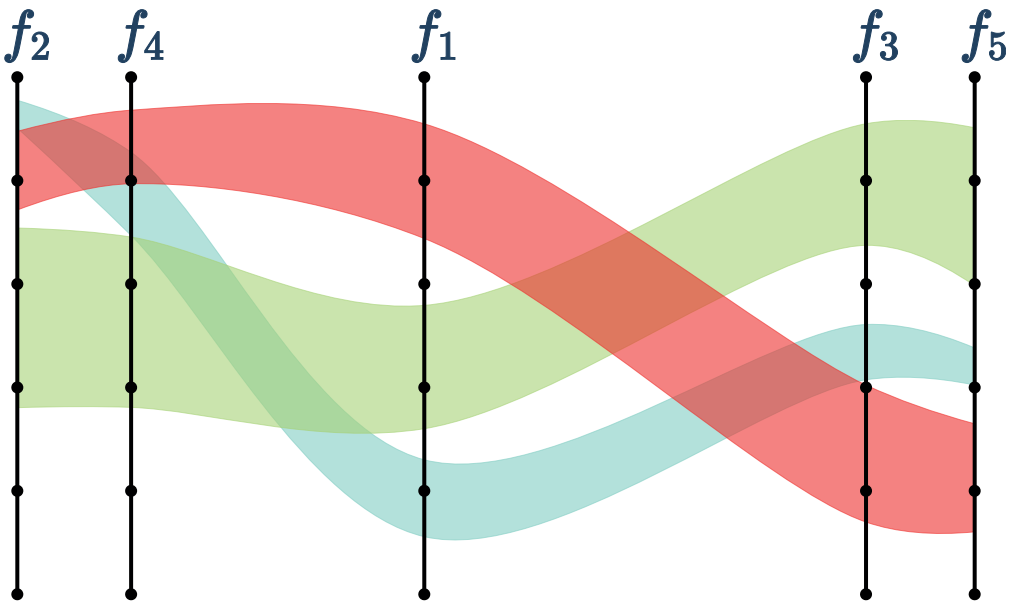
JYU DISSERTATIONS 556

Bhupinder Singh Saini

Pioneering Techniques to Tackle Challenges of Interactive Multiobjective Optimization



UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY

JYU DISSERTATIONS 556

Bhupinder Singh Saini

Pioneering Techniques to Tackle Challenges of Interactive Multiobjective Optimization

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston vanhassa juhlasalissa S212
syyskuun 9. päivänä 2022 kello 13.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Seminarium, auditorium S212, on September 9, 2022 at 13 o'clock.

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2022

Editors
Marja-Leena Rantalainen
Faculty of Information Technology, University of Jyväskylä
Ville Korkiakangas
Open Science Centre, University of Jyväskylä

Cover picture by Bhupinder Singh Saini.

ISBN 978-951-39-9196-8 (PDF)
URN:ISBN:978-951-39-9196-8
ISSN 2489-9003

Permanent link to this publication: http://urn.fi/URN:ISBN:978-951-39-9196-8

ABSTRACT

Saini, Bhupinder Singh
Pioneering Techniques to Tackle Challenges of Interactive Multiobjective Optimization
Jyväskylä: University of Jyväskylä, 2022, 80 p. (+included articles)
(JYU Dissertations
ISSN 2489-9003; 556)
ISBN 978-951-39-9196-8 (PDF)

Decision-makers (DMs) must often consider several, potentially conflicting objective functions simultaneously before making a decision. Such problems do not usually have a single optimal solution. Instead, they typically have (even infinitely) many so-called Pareto optimal solutions representing different trade-offs between the objectives. One of the ways to solve these multiobjective optimization problems (MOPs) is to use interactive methods that incorporate the DM's preferences during the solution process.

Interactive multiobjective optimization has various challenges. The process of formulating an MOP can itself be challenging. How to decide which objectives to consider or which method to use to solve the MOP? The implementations of many published methods are not openly available, which introduces additional challenges. In certain MOPs, objectives can only be evaluated by experimentation or conducting lengthy computer simulations introducing a need to replace the objectives with less costly machine learning models trained on data. However, this introduces further complications, including choosing the best models for the MOP and model management. Finally, there is also the issue of visualizing the solutions to the DM and enabling them to interact with the method intuitively.

This thesis tackles the aforementioned problems and more. We propose the so-called SMTS algorithm, which predicts the best machine learning model for MOPs. With the so-called IOPIS algorithm, we introduce a completely new paradigm for interactive multiobjective optimization, enabling modular creation of interactive methods and supporting various ways of incorporating preferences. We propose the O-NAUTILUS algorithm to tackle problems with costly function evaluations in a way that allows a DM to conduct targeted evaluations in their region of interest. We introduce a novel visualization technique, SCORE bands, which can simultaneously visualize thousands of solutions with up to a dozen objectives. The DESDEO framework provides free access to the algorithms mentioned above (and many others). The framework enables its users to utilize the implemented algorithms and easily combine parts of them to create whole new ones. Finally, we put the above into practice with a case study: solving a complex data-driven metallurgical problem using the tools provided by DESDEO.

Keywords: preference-based optimization, surrogate modelling, evolutionary algorithms, visualization, decision making, open-source software

TIIVISTELMÄ (ABSTRACT IN FINNISH)

Saini, Bhupinder Singh
Uraauurtavia menetelmiä vastaamaan interaktiivisen monitavoiteoptimoinnin haasteisiin
Jyväskylä: University of Jyväskylä, 2022, 80 s. (+artikkelit)
(JYU Dissertations
ISSN 2489-9003; 556)
ISBN 978-951-39-9196-8 (PDF)

Tehdessään päätöksiä päätöksentekijöiden tulee usein samanaikaisesti huomioida monia, tyypillisesti ristiriitaisia tavoitefunktioita. Näillä ns. monitavoiteoptimoinnin ongelmilla ei ole yhtä optimia vaan useita (jopa ääretön määrä) kompromissiratkaisuja, joita kutsutaan Pareto-optimaalisiksi. Ratkaisemiseen voidaan käyttää interaktiivisia menetelmiä, jotka huomioivat päätöksentekijän mieltymykset ratkaisuprosessin aikana. Interaktiivisten menetelmien käytössä on kuitenkin haasteita. Itse optimointiongelman muotoileminen voi olla vaativaa. Miten valitaan optimoitavat tavoitteet ja käytettävä menetelmä? Monilta julkaistuilta menetelmiltä puuttuvat avoimesti saatavissa olevat implementaatiot, mikä osaltaan vaikeuttaa niiden käyttöä. Joissakin ongelmissa tavoitearvojen laskeminen edellyttää käytännön kokeita tai kalliita tietokonesimulaatioita. Tällöin on mielekästä korvata tavoitteet vähemmän kalliilla koneopin malleilla, jotka sovitetaan ongelman dataan. Tässäkin on omat haasteensa, esimerkiksi miten mallit valitaan ja miten niitä hallitaan. Tärkeää on myös havainnollistaa ratkaisuja päätöksentekijälle ja varmistaa että interaktiivisten menetelmien käyttö on ymmärrettävää.

Tässä väitöskirjassa tarjotaan vastauksia edellä mainittuihin haasteisiin. Työssä esitellään ns. SMTS-menetelmä, joka ehdottaa parhaan koneopin mallin optimointitehtävän muotoiluun. IOPIS-menetelmä puolestaan tarjoaa täysin uuden paradigman interaktiiviseen monitavoiteoptimointiin. Sen avulla voidaan modulaarisesti luoda uusia interaktiivisia menetelmiä ja huomioida eri tavoin esitettyjä päätöksentekijän mieltymyksiä. Jos ongelmassa on kalliita tavoitteita, päätöksentekijä voi kohdentaa niiden arvojen laskemisen hänelle kiinnostaviin alueisiin O-NAUTILUS-menetelmällä. Työssä esitellään myös uusi visualisointimenetelmä SCORE bands, joka pystyy havainnollistamaan samanaikaisesti jopa tuhansia ratkaisuvaihtoehtoja. Työssä kuvataan myös Pythonilla tehty avoimen lähdekoodin modulaarinen DESDEO-ohjelmistokehikko, joka sisältää monia interaktiivisia menetelmiä. Lopuksi edellä esiteltyjen uusien menetelmien käyttökelpoisuutta havainnollistetaan ratkaisemalla interaktiivisesti DESDEOn avulla vaativa datapohjainen monitavoiteoptimointiongelma metallurgian alalta.

Avainsanat: preferenssipohjainen optimointi, monitavoiteoptimointimenetelmät, interaktiiviset menetelmät, sijaismallit, evoluutioalgoritmit, visualisointi, päätöksenteko, avoin lähdekoodi

Author

Bhupinder Singh Saini
Faculty of Information Technology
University of Jyväskylä
Finland

Supervisors

Professor Kaisa Miettinen
Faculty of Information Technology
University of Jyväskylä
Finland

Dr. Babooshka Shavazipour
Faculty of Information Technology
University of Jyväskylä
Finland

Reviewers

Professor Hisao Ishibuchi
Department of Computer Science and Engineering
Southern University of Science and Technology
China

Professor Mariano Luque Gallego
Faculty of Economics and Business Administration
University of Málaga
Spain

Opponent

Professor Jyrki Wallenius
Department of Information and Service Management
Aalto University
Finland

ACKNOWLEDGEMENTS

I want to thank my supervisors Prof. Kaisa Miettinen and Dr. Babooshka Shavazipour, for their continued support and advice during my doctoral studies. They always urged me to pursue my ideas, nudged me in the right direction, and helped me overcome many challenges. Their expertise in multiobjective optimization and their confidence in my ideas (sometimes even when I did not have the same confidence) encouraged me greatly. I especially want to thank Prof. Miettinen for her invaluable critiques of my works, pushing me to become a better researcher. Her enthusiasm regarding interactive multiobjective optimization was one of the reasons why working in this field has been so fulfilling. For this, I am eternally grateful.

I also want to thank Prof. Nirupam Chakraborti from Czech Technical University (former Professor at IIT Kharagpur), who first introduced me to multiobjective optimization and set me down the path that culminated in the writing of this thesis. In addition, I want to thank my international collaborators, especially Dr. Manuel López-Ibáñez (University of Málaga) and Dr. Michael Emmerich, for sharing their expertise. For reviewing this thesis, I want to thank Prof. Hisao Ishibuchi (Southern University of Science and Technology) and Prof. Mariano Luque Gallego (University of Málaga).

My doctoral studies were funded by the Academy of Finland (grant numbers 287496 and 322221). I am grateful for the generosity. The grants allowed me to make working on the DESDEO framework an essential part of my doctoral studies. Working with the DESDEO development team, especially Giovanni Misitano and Giomara Lárraga Maldonado, has been a great pleasure.

The current and former members of the Multiobjective Optimization Group at the University of Jyväskylä played a fundamental part in my growth as an academic researcher and development of this thesis. I would especially like to thank Dr. Atanu Mazumdar, Pouya Aghaei Pour, and Adhe Kania for being the first adopters of the DESDEO framework and for providing valuable comments. I would also like to thank Dr. Jussi Hakanen for his guidance.

I sincerely appreciate my friends and family, who have always supported and encouraged me. Lastly, I would like to thank my mother, Kiranjeet Kaur. Her love and support made me the person I am today.

LIST OF FIGURES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Bhupinder Singh Saini, Manuel López-Ibáñez, Kaisa Miettinen. Automatic Surrogate Modelling Technique Selection based on Features of Optimization Problems. *Proceedings of the Genetic and Evolutionary Computation Conference Companion, Edited by M. López-Ibáñez, ACM, NY, USA, 1765–1772*, 2019.

PII Bhupinder Singh Saini, Jussi Hakanen, Kaisa Miettinen. A New Paradigm in Interactive Evolutionary Multiobjective Optimization. *Parallel Problem Solving from Nature – PPSN XVI, Edited by T. Bäck, M. Preuss, A Deutz, H. Wang, C. Doerr, M. Emmerich, H. Trautmann, Springer, Cham, 243–256*, 2020.

PIII Bhupinder Singh Saini, Kaisa Miettinen, Kathrin Klamroth, Ralph E. Steuer, Kerstin Dächert. SCORE Band Visualizations: Supporting Decision Makers in Comparing High-Dimensional Objective Vectors in Multiobjective Optimization. *Submitted to a journal*.

PIV Bhupinder Singh Saini, Michael Emmerich, Atanu Mazumdar, Bekir Afsar, Babooshka Shavazipour, Kaisa Miettinen. Optimistic NAUTILUS Navigator for Multiobjective Optimization with Costly Function Evaluations. *Journal of Global Optimization, 83, 865–889*, 2022.

PV Giovanni Misitano, Bhupinder Singh Saini, Bekir Afsar, Babooshka Shavazipour, Kaisa Miettinen. DESDEO: The Modular and Open Source Framework for Interactive Multiobjective Optimization. *IEEE Access, 9, 148277–148295*, 2021.

PVI Bhupinder Singh Saini, Debalay Chakrabarti, Nirupam Chakraborti, Babooshka Shavazipour, Kaisa Miettinen. Interactive Data-driven Multiobjective Optimization of Metallurgical Properties of Microalloyed Steels using DESDEO. *Submitted to a journal*.

The author's contribution is described in Section 9.2.

1 INTRODUCTION

No man ever wetted clay and then left it, as if there would be bricks by chance and fortune.

Attributed to Plutarch

Decision makers (DMs) often have to consider a plethora of objectives before making a decision. A person choosing a car to buy may consider not only its fuel efficiency but also its power, acceleration, weight, and production year. A city planner placing a fishery may consider the effect of the placement on the production, profits and tax revenue, as well as the ecological, sociological and environmental impacts. A vehicle chassis designer may consider the cost of materials and production, the weight of the chassis, and various measures of crashworthiness. Often, the objectives of such problems are conflicting in nature, and the issue of choosing a solution is not trivial.

Different DMs need different methods to help them solve their problem. The person choosing a car can look up the reviews to gather the necessary data to start the decision making process. On the other hand, the city planner may not have such data. Instead, they may rely on some mathematical models that try to predict the impact of the placement on the various objectives. Such problems necessitate one or more optimization steps, where all the objectives are simultaneously optimized. Due to the conflicting nature of the objectives, such so-called multiobjective optimization problems (MOPs) typically have a set of optimal solutions (instead of a single optimal solution). This set of solutions, known as Pareto optimal solutions, is the only set that needs to be considered by DMs if all relevant objectives are included in the problem formulation.

Many optimization methods have been proposed to solve MOPs [52,70,104]. They can be categorized into three classes based on the involvement of the decision maker [70]:

– *a priori* methods: The optimization algorithm considers the preferences of the DM and returns Pareto optimal solutions that follow the preferences of the DM.

- *a posteriori* methods: The optimization algorithm does not utilize the preferences of a DM. Instead, it generates a representative set of Pareto optimal solutions.
- Interactive methods: The optimization process takes place in iterative steps. In each step, the DM provides their preferences to the optimization algorithm. The algorithm then returns a solution which may satisfy their preferences or lead them to change their preferences, resulting in a new iteration. Hence, the DM "interacts" with the optimization algorithm until a satisfactory solution is obtained.

Unlike *a priori* methods, interactive methods enable the DM to learn new information about the MOP, such as the trade-offs involved. The DM can change their preferences interactively to gain this information. The DM can also gain this information using the results from an *a posteriori* method. However, generating a representative set of Pareto optimal solutions may be challenging, especially in MOPs with many objectives. Moreover, these methods require the DM to analyze potentially thousands of solutions at once, which is cognitively challenging. Interactive methods overcome these challenges by focusing only on solutions that are preferable to the DM in the current iteration. This makes it easier to find Pareto optimal solutions and the DM only has to focus on a small number of solutions to give their preferences for the next iteration. This thesis focuses primarily on interactive methods.

Popular multiobjective optimization algorithms include scalarization-based methods [70] and multiobjective evolutionary algorithms (MOEAs) [52,104]. Scalarization-based methods use the preferences of the DM along with a scalarization function to convert a vector of objective values into a scalar value. This scalarization enables the usage of traditional single-objective optimizers for solving MOPs. When the scalarization function is chosen carefully, one can prove that the solution obtained is Pareto optimal to the original MOP [88]. MOEAs, on the other hand, emulate the process of evolution and work with a population of solutions. This population "evolves" over several generations to converge closer to the Pareto optimal set of solutions. Most MOEAs are *a posteriori* and perform especially well in generating a representative set of near-Pareto optimal solutions [52]. However, they also require a large number of objective function evaluations to converge [54].

The requirement of many objective function evaluations engenders many MOEAs to be inappropriate for some MOPs. The vehicle chassis designer, for example, may not have exhaustive data or simple mathematical models. Instead, they may be using computer simulations which can take minutes, hours, days, or even weeks to simulate a single solution. Alternatively, they may have to create and test the chassis corresponding to any solution to evaluate it. In either case, it is not possible to evaluate thousands of solutions. An alternative is to conduct surrogate-assisted data-driven optimization [20]. A small number of objective function evaluations are used to train surrogate models (predictive machine learning models or metamodels), which MOEAs can then use instead of the original objectives. If possible, new function evaluations can be used to update the

models every few generations, leading to increasingly better accuracy.

Solving an MOP requires the resolution of many challenges [100]. The problem formulation process itself may not be a trivial task. The choice of objectives to be optimized may evolve during the solution process. The choice and nature of objectives (mathematical models vs surrogate models, fast and cheap vs slow and expensive to evaluate) depend highly on the resources available, the stakeholders involved, and the familiarity and experience of the people solving the MOP with the various methodologies available to do so.

A large number of methods to solve MOPs are published every year. Even in the narrower field of surrogate-assisted optimization, many choices need to be made to train the surrogate models themselves (the choice of machine learning algorithm, the hyperparameters of the model, and strategies to update the model if more data is available). A DM, who is an expert in their domain but not in multiobjective optimization, cannot be expected to make all of these choices. Instead, an *analyst* with expertise in multiobjective optimization makes those choices and guides the DM through the decision making and problem-solving process. However, the issue of making those choices remains. The problem is further exacerbated by the fact that while many methods for solving MOPs have been published, their implementations may not be openly available [PV].

There is also the issue of presenting solutions obtained by the methods to the DM in a meaningful manner. One way to present solutions is via visualizations; however, different DMs may require very different visualizations to tackle the needs of different MOPs. The person choosing a car, for example, may prefer an interactive visualization which can present tens of solutions (alternative cars) in an easy to comprehend manner. The city planner placing the fishery may want the information about the objectives presented on a geographical map. This may limit the density of information presented (only one solution per map) but may make it easier for the city planner to provide their preferences for interactive optimization. The vehicle chassis designer solving the problem using surrogate-assisted optimization may want to visualize not just the objective values but also the uncertainty of prediction of the surrogate models used during optimization, as the amount of such uncertainty may affect their choice of solution.

This thesis is a collection of six articles [PI] – [PVI], published in scientific journals and conference proceedings, discussing and tackling challenges involved with various steps of interactive multiobjective optimization. The thesis examines challenges arising during problem formulation, method development and accessibility, visualization, and DM interaction. Specifically, we tackle the following challenges:

1. How do we choose the best surrogate modelling techniques to mimic the objectives of a data-driven MOP?
2. How do we reduce the number of function evaluations required by MOEAs to solve MOPs with expensive objective functions?
3. Problems with higher number of objectives are increasingly difficult to solve. Can we use scalarization functions and MOEAs together to make interactive methods that get around this issue?

4. How can we create interactive MOEAs that allow the DM to precisely control the optimization process?
5. How can we create intuitive and helpful visualizations to help a DM understand the trade-offs involved?
6. How can we create visualizations that display the uncertainity from predictions of surrogate models in a data-driven MOP to a DM?
7. How can we enable the DM to control how expensive function evaluations are spent?
8. How do we make interactive methods accessible to DMs, analysts, researchers, and students for solving MOPs, experimenting with interactive methods, or creating new interactive methods?
9. How do we tackle unexpected challenges that one may face while solving a real-life data-driven MOP?

Chapter 3 tackles the first challenge with the Surrogate Modelling Technique Selector, introduced in [PI]. The Selector is a machine learning model trained to predict the best surrogate modelling technique for data-driven multiobjective optimization problems based on the data's exploratory landscape features [69]. To achieve this, the Selector was trained on the performances of many popular surrogate modelling techniques on a large number of benchmark and real multiobjective optimization problems. Thanks to the trained Selector, an analyst with data of some real phenomenon can be supported in finding the most appropriate surrogate modelling technique to model objectives captured in the data.

In Chapter 4 we present a new paradigm for interactive multiobjective optimization to tackle the second, third, and fourth challenges. We propose the corresponding IOPIS algorithm in [PII]. The IOPIS algorithm combines scalarization-based methods and MOEAs in novel ways. The algorithm uses multiple scalarization functions and the preferences of a DM to create a new space that we call the preference incorporated space (PIS). Then, an MOEA is used to optimize in the PIS, resulting in convergence toward Pareto optimal solutions in the region of interest of the DM. The IOPIS algorithm enables easy and trivial conversion of popular *a posteriori* MOEAs into interactive methods. Moreover, the analyst can easily control the number of dimensions of the PIS. It is equal to the number of scalarization functions used and independent of the number of objectives. Lowering the number of dimensions can have many benefits, ranging from allowing the usage of MOEAs that perform better in a smaller number of dimensions to drastically reducing the number of objective function evaluations. We implemented two variants of the IOPIS algorithm and compared them against popular interactive and *a posteriori* MOEAs in a large number of scenarios.

We introduce the so-called SCORE bands visualization technique in Chapter 5 to support DMs in problems with many objectives and a large number of Pareto optimal solutions to be compared (fifth challenge). SCORE bands, proposed in [PIII], is an evolution of the parallel coordinates plot [41] that uses and displays information such as the presence of clusters in the Pareto optimal solutions and correlations between the various objectives in a single interactive vi-

sualization. The resulting visualization is rich in information and yet simpler in presentation than a parallel coordinate plot displaying the same data. Using clustering information allows the DM to show or hide the clusters in the interactive visualization, whereas using correlation information informs the DM about the trade-offs between the objectives visually. SCORE bands support a DM in understanding complicated dependencies and enables digesting large amounts of information by, for example, studying large trends first. We implemented an application that an analyst can use to create SCORE bands plots from raw data and presented case studies with various artificial, benchmark and real MOPs with up to dozen objectives and more than 1000 solutions.

In Chapter 6 we present the O-NAUTILUS algorithm for MOPs with expensive function evaluations. O-NAUTILUS, proposed in [PIV] combines a novel visualization technique as well as a new interactive method to tackle the second, sixth, and seventh challenges. Like IOPIS, the O-NAUTILUS algorithm also combines scalarization-based methods and MOEAs. It uses uncertainty predictions from surrogate modelling techniques such as Kriging [97] and Lipshitzian models [6] with MOEAs to create an "optimistic" representation of the Pareto optimal solutions. This optimistic representation informs the DM about solutions that may attain good objective values and can be discovered with one or a small number of function evaluations. The algorithm presents this information to the DM in an interface similar to NAUTILUS Navigator [84]. The interface helps the DM identify a region of interest based on the models' objective values and prediction uncertainty. Finally, the algorithm uses a scalarization function to identify the best solution to evaluate using the actual (but expensive) objective functions in the region of interest of the DM. The algorithm then retrains the models using this newly evaluated solution, recreates the optimistic solutions, and enables the DM to use the interface again, with more accurate solutions in the region of interest. The DM stops the algorithm once they find a satisfactory solution. Thus, the O-NAUTILUS algorithm tackles the issues of data-driven optimization with expensive functions, visualization of uncertainty information, interaction with the DM, and management of surrogate models and function evaluations. We implemented a web-based interface for O-NAUTILUS and compared the algorithm against other data-driven optimization algorithms using a case study.

Chapter 7 introduces the DESDEO framework and tackles the challenges involved with the creation and accessibility of interactive methods (the second and eighth challenges). The DESDEO framework, presented in [PV], is a collection of open-source Python and TypeScript packages which provide easy-to-use modular implementations of popular interactive MOEAs and scalarization-based methods. We introduce the structure of the core packages of the framework. The framework supports various use cases: formulating an MOP, solving MOPs using various interactive methods, and combining various methods to create complex optimization pipelines. The modular nature of the implementation enables researchers to easily create new methods by combining parts of previously implemented methods [PII, PIV, PVI, 3, 67]. The framework also provides analysts with quick access to open-source implementations of many methods, reducing

the barrier to conducting comparative studies or case studies [1,2,56].

Finally, in Chapter 8 we explore and tackle the challenges of solving a real data-driven MOP, a steel alloy composition problem (the ninth challenge). The major issues we tackled include:

- preprocessing of data to appropriately use it for data-driven optimization,
- training and validating of the surrogate models,
- choosing the objectives to formulate the MOP, and
- designing an interactive MOEA to handle preferences of two DMs who were previously unfamiliar with interactive methods.

We utilized most of the tools and methods developed for the other papers introduced in this thesis, alongside other popular open-source tools. The chapter and the corresponding paper [PVI] provide a detailed account of our measures to resolve the challenges, including measures that ultimately failed. The chapter thus functions as a general guideline to resolving challenges researchers, analysts, and DMs may encounter while solving real data-driven MOPs.

The rest of the thesis is structured as follows. Chapter 2 establishes the core concepts and terms used throughout the thesis. Chapters 3 – 8 introduce articles [PI] – [PVI], respectively. Each of those chapters begins by articulating the challenges tackled by the associated article, followed by a summary of the fundamental details of proposed pioneering techniques and a brief discussion of the results. Finally, we provide our conclusions in Chapter 9, put forward some directions for future research and describe the author's contributions in the included articles.

The process of solving an MOP is often portrayed in literature as a linear process: from problem formulation and optimization to visualization and decision making. The advent of interactive methods challenges that narrative, forcing DMs to carefully consider their perspective by simply making them aware of their control over the optimization process. However, the narrative has always been somewhat misleading. The previously mentioned challenges can sometimes lead to unresolvable failures, forcing DMs and analysts to start from the very beginning. Challenges faced in later steps of optimization may force them to reconsider the decisions made in the former steps. At the same time, the decisions made in the earlier steps limit the choices they can make in the later steps. Ultimately, the aim of this thesis is to discuss the possibilities provided by interactive methods; that these methods can not only resolve those challenges but also turn them into opportunities for DMs to learn more about their MOPs and lead them towards making better decisions.

2 BACKGROUND CONCEPTS

If you wish to make an apple pie from scratch, you must first invent the universe.

Carl Sagan, Cosmos

In this chapter, we introduce the main concepts used throughout this thesis. We first define the basic terms related to MOPs in Section 2.1. We then focus specifically on interactive optimization methods such as NAUTILUS Navigator, and scalarization functions in Section 2.2. Understanding the basics of this topic is necessary for Chapters 4, 6, 7, and 8. In Section 2.3, we give a brief introduction of multiobjective evolutionary algorithms, required for Chapters 4, 6, 7, and 8. We follow that by giving a brief description of some of the visualization tools used by interactive methods in Section 2.4, useful for Chapter 5. Finally, in Section 2.5, we establish the core metallurgical concepts used in the MOP we solve in Chapter 8.

2.1 Multiobjective optimization problems

We can define an MOP as:

$$\begin{aligned} &\text{minimize} && \{f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ &\text{subject to} && \mathbf{x} \in S, \end{aligned} \tag{1}$$

where $\mathbf{x}$ is a vector $(x_1, \ldots, x_n)^T$ of n decision variables. The decision variables can be confined to $S \subset \mathbb{R}^n$ by constraint functions, forming a so-called feasible set. These constraints can be upper and lower bounds on the decision variable values in the simplest case. The objective functions $\mathbf{f} = (f_1, \ldots, f_k)$ map vectors of S (the decision space) to objective function values (shortened to objective values) in $\mathbb{R}^k$, forming an objective space. For MOPs, k is greater than or equal to two, however MOPs with two objectives are generally called biobjective problems in the literature. MOPs in real life can have objectives that require maximization.

However, problem (1) can represent all such MOPs without loss of generality by simply multiplying the objectives to be maximized by -1.

To find optimal solutions of problem (1), we first need to be able to impart an ordering to vectors in the k-dimensional objective space. We can do so by using the concept of dominance [1]. A solution $\mathbf{x}^1 \in S$ (and the corresponding objective vector $\mathbf{f}(\mathbf{x}^1)$) is said to dominate another solution $\mathbf{x}^2 \in S$ (and its corresponding objective vector $\mathbf{f}(\mathbf{x}^2)$) if $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2)$ for all $i = 1, \ldots, k$ and $f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$ for at least one $j = 1, \ldots, k$. Using this relation, we can define the so-called Pareto set $PS \subseteq S$ as the set of all feasible solutions not dominated by any other feasible solution. By definition, the solutions in PS are mutually non-dominating and their image in the objective space, called a Pareto front, represent trade-offs between the different objectives.

Using the Pareto front, we can define a few points of significance in the objective space. The ideal point represents the best possible values attainable by each objective function independently. Generally, the ideal point is not feasible, except in cases where the objectives do not conflict with each other. In such cases, the Pareto front consists simply of the ideal point. Incidentally, we do not need to solve the MOP to find the ideal point; we only need to optimize the objective functions separately. A related point, the utopian point, is defined as a point very close to the ideal point but slightly dominating it. We can generate it by subtracting a small positive value from the components of the ideal point. The nadir point represents the worst possible objective values attained by Pareto optimal solutions. Unlike the ideal and utopian points, we need complete knowledge of the Pareto front to calculate the nadir point accurately, which is usually not possible. However, a few approximate methods exist, see, e.g. [25, 26, 70]. These points are of note because many methods which solve MOPs use them or their approximations.

MOPs can have thousands (or even an infinite number) of Pareto optimal solutions that do not dominate each other [70], i.e., are incomparable to each other without any additional information. However, in real-life MOPs, not all Pareto optimal solutions are equally important to a DM. Generally, a DM's ultimate goal in solving an MOP is to implement one or a few Pareto optimal solutions. A DM, who is usually a domain expert, may therefore already have preferences regarding the kind of solutions they want to implement. If they do not start with a well-defined preference (such as when, for example, a DM is solving a new MOP without the knowledge of what is feasible), they may still find some solutions preferable to others during the optimization process.

Interactive optimization methods exploit this fact and involve the DM deeply in the optimization process. As mentioned in the introduction, these methods conduct optimization iteratively [70, 104]. The DM provides their preferences to the method at the beginning of an iteration. The method then finds one of more Pareto optimal solutions that "satisfy" the DM's preference as well as possible. If the DM is truly satisfied, they can terminate the method. Otherwise, they can update their mental model of what solutions may be feasible, thereby learning more

[1] More specifically, the dominance relation is a weak partial order.

about the MOP. They learn what kinds of preferences are reachable and provide new preferences.

One way to incorporate the preferences of a DM is to use a *scalarization function* [70, 88]. Scalarization functions map the vectors in objective space to scalar values. The preferences can be incorporated using the parameters of such functions. The interactive method can then use a traditional single-objective optimization algorithms to solve the (now) single-objective problem to find a Pareto optimal solution that satisfies the DM's preferences. We discuss scalarization functions and ways of providing preferences in further detail in Section 2.2.

Multiobjective evolutionary algorithms (MOEAs) provide another way of solving MOPs. MOEAs are a family of nature-inspired methods that *evolve* a group of solutions, known as the population, towards optimality by mimicking the process of evolution [21, 23, 52]. Generally, the individual solutions in the population recombine their properties (decision variable values) to form new solutions: the offspring. The offspring are then evaluated and are compared against each other (and their parents) using a selection operator. The selection operator kills off solutions that are not "fit", making the population better on average. MOEAs repeat these operations over multiple generations, pushing the population closer toward the Pareto optimal solutions.

MOEAs are metaheuristic algorithms: they cannot guarantee Pareto optimality [21]. Most MOEAs are *a posteriori* and try to approximate a representative set of Pareto optimal solutions [21], though *a priori* and interactive MOEAs also exist [61,104]. The details of how MOEAs create new offspring and how selection operators judge the fitness of solutions differ among the various MOEAs. We discuss MOEAs further in Section 2.3.

We can define the objective functions of an MOP in many different ways. The values of the objective functions can be evaluated using simple analytical functions, complicated and time-consuming computer simulations, or even real-life experiments. In the latter two cases, we may have limited or no capacity to evaluate objective function values for new solutions. Instead, we can use data from past experiments (previously evaluated solutions) to enable us to conduct optimization using very few or no additional objective function evaluations. Such problems are called data-driven MOPs [55].

Surrogate-assisted optimization methods provide one way to solve data-driven MOPs [18,54,66]. These methods use the data to form "surrogate models" which mimic the behavior of the actual objective functions. The optimization algorithm then uses these surrogate models in place of the objective functions to solve the MOP. The accuracy of the solutions found thus depends on the accuracy of the surrogate models, which are usually regression algorithms such as Gaussian process regression[2] [33,65,97], neural networks [39], support vector machines [91], or ensemble methods [16, 37, 43, 55, 57]. MOPs with the possibility to evaluate the value of the true objective function for some solutions (for example, to increase the accuracy of the surrogate models) are termed "online" data-driven MOPs [18, 20, 55]. "Offline" data-driven MOPs are instead limited to the starting

[2] also known as Kriging.

dataset with no possibility of further true objective function evaluations [55, 66].

2.2 Interactive optimization methods and scalarization Functions

Different interactive methods allow a DM to provide their preferences in different fashions [70, 87, 104]. Some methods, such as reference point method [101, 102], allow the DM to provide a "reference point"[3], which is a vector in the objective space consisting of objective values that the DM finds satisfactory. The method then tries to find solutions similar to the reference point or converge along its direction. Other methods can use preferences in the form of "reservation levels" which also form a vector in the objective space [103]. Reservation levels are objective values that a DM wants to avoid. These levels can be used to focus the search only in regions of the objective space better than the levels. Another way of providing preferences is through the use of the so-called navigation-based methods inspired by Pareto race [59]. These methods [34,49] allow a DM to navigate along a Pareto front: the DM chooses the speed and direction of navigation and the methods display the changing Pareto optimal solutions in real-time.

Yet more ways of providing preferences include choosing a preferred solution from among a few alternatives, for example, in E-NAUTILUS [86]. The method then provides solutions similar to the chosen one. Alternatively, the DM can choose solutions that they do not like [46]. A method then ignores the areas in the objective space around such non-preferred solutions in future iterations. The NIMBUS method [74, 76] asks a DM to choose a solution and then provide preferences in the form of classification of the objectives. The DM may classify the objectives as "make better than", "allow to worsen", or "keep the same as" the chosen solution. If the DM classifies a particular objective to change in value, they can provide a value they wish to achieve or a bound they do not want to cross.

As mentioned earlier, scalarization functions can be used to incorporate DM preferences. These functions use the preference information to impart a total order in the objective space by mapping the objective vectors to scalar values. The preference can be incorporated using the parameters of such functions. However, not all scalar-valued functions are appropriate for interactive multiobjective optimization as scalarization functions. Some desirable properties for such functions, as identified in [88] are:

1. The scalarization functions should cover all Pareto optimal solutions, i.e., the DM should be able to discover any Pareto optimal solution by changing their preference that are parameters of the scalarization function.
2. The optimal solution of the single-objective problem formed by the scalarization function should be a Pareto optimal solution of the original MOP.

[3] also known as aspiration levels.

3. The optimal solution of the single-objective problem formed by the scalarization function should satisfy the DM (if their preferences are feasible).

No single scalarization function can satisfy all three criteria at once. For example, a weighted sum of the objectives (where the weights represent the preferences) will not find all Pareto optimal solutions of an MOP with a nonconvex Pareto front, breaking the first criteria [88]. The achievement scalarizing function [101] can be formulated as:

$$s(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}) = \max_{i=1,\ldots,k} \left[\frac{f_i(\mathbf{x}) - \bar{z}_i}{\mu_i} \right] + \rho \sum_{i=1}^{k} \left(\frac{f_i(\mathbf{x}) - \bar{z}_i}{\mu_i} \right), \tag{2}$$

where ρ is a small positive scalar, $\boldsymbol{\mu}$ and $\bar{\mathbf{z}}$ are vectors in the objective space. The vector $\bar{\mathbf{z}}$ is the preference provided by the DM in the form of a reference point. By minimizing (2), we can find a Pareto optimal solution of problem (1) along a vector passing through the reference point, in a direction parallel to $\boldsymbol{\mu}$ (a vector going from the nadir point to the ideal point in the original formulation). The achievement scalarizing function satisfy the second and third criteria [70]. Moreover, the function can cover arbitrarily large subsets of the Pareto optimal solutions (related to the first criteria) by setting smaller values for ρ and changing DM preferences. However, the function breaks the second criteria if ρ is set to zero [4].

Note that we do not expect a DM to be well informed about the intricacies of various interactive methods. Instead, we assume that an analyst, who is an expert in such methods, guides the DM through the optimization process. An analyst, among other tasks, chooses the interactive optimization method to solve the MOP. The GLIDE-II formulation [87] enables an analyst to create scalarization functions used in many different interactive optimization methods (and use different kinds of preferences) by simply changing the parameters of a formula.

Methods such as NIMBUS and the reference point method provide a few (preferable) Pareto optimal solutions to the DM at the end of each iteration. The DM must then either choose one of those solutions as the final solution, or provide new preferences, trading-off one or more objectives to improve others. On the other hand, NAUTILUS Navigator [84] takes a novel approach to interaction with the DM and does not provide Pareto optimal solutions to the DM for most of the interaction process [5]. Instead, it conducts the interaction process by putting the DM's point of view on a "step point"[6], a point in the feasible part of the objective space.

The method begins by putting the step point at the nadir point. The method requires as input a representative set of Pareto optimal solutions to operate. By

[4] More specifically, when ρ is set to zero, the optimal solution of the scalarized problem may be weakly Pareto optimal in the original MOP. A solution is weakly Pareto optimal if no other solution is better than it in *all* objectives. However, there may exist some solutions that are better than it in some objectives.

[5] It should be noted that NAUTILUS Navigator is not a navigation-based method as defined earlier. NAUTILUS Navigator does not allow the DM to navigate along the Pareto front.

[6] The step point is called the current point in the original article. We use the term step point as that is the term we use in the later chapters.

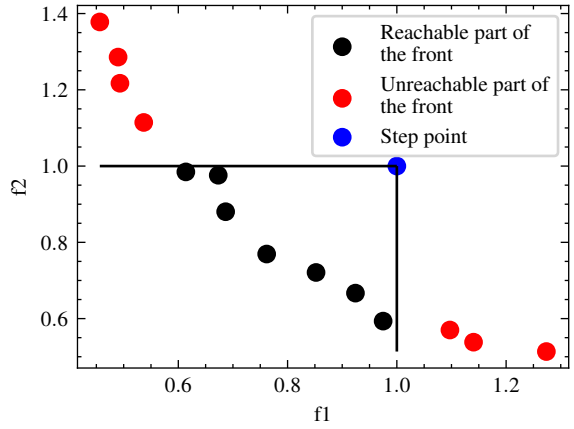


FIGURE 1 Visual representation for the concept of reachability for a biobjective problem.

starting at the nadir point, the NAUTILUS Navigator method asks the DM to choose a direction of improvement to improve all objective values without any trade-offs. The direction of improvement can come from a reference point (called aspiration levels in the method) that the DM wishes to achieve. Then the step point is moved closer to the Pareto front in the direction the DM chooses. The aspiration levels must always dominate the step point to progress further. Thus areas of the Pareto front are rendered "unreachable" as the step point moves closer to the Pareto optimal solutions.

We show the concept of reachability for an MOP with two objectives visually in Figure 1. The blue point is the step point. The method calculated the reachable part of the front as the subset of Pareto optimal solutions (from the previously mentioned representative set) that dominate the step point, shows as black points. The DM can reach any of those points from the current step point by setting the aspiration levels appropriately. The unreachable part is shown as red points. The method ends when the step point reaches one of the Pareto optimal solutions.

NAUTILUS Navigator simplifies making decisions and choosing aspiration levels by only visualizing the "reachable" region of the Pareto front for each objective independently. We show a snapshot of the "navigator" plot for one of the objectives of an MOP in Figure 2 [7]. The x-axis of the plot conveys the progress of the step point. The black vertical line move from left to right as the step point moves from the nadir point to one of the solutions in the representative set of Pareto optimal solutions. The y-axis represents the objective values. The span of the green shaded region, which is drawn as the black vertical line moves forward, represents the range of objective values that are still reachable at any step. As expected, the green region spans the entire vertical height at the left edge (when the step point was at the nadir point). The reachable range shrinks as the black vertical bar sweeps through the plot. At the current step, the reachable range for the objective lie between 4 and 6 units.

[7] Note that the author reimplemented the navigator plot in DESDEO based on inspiration

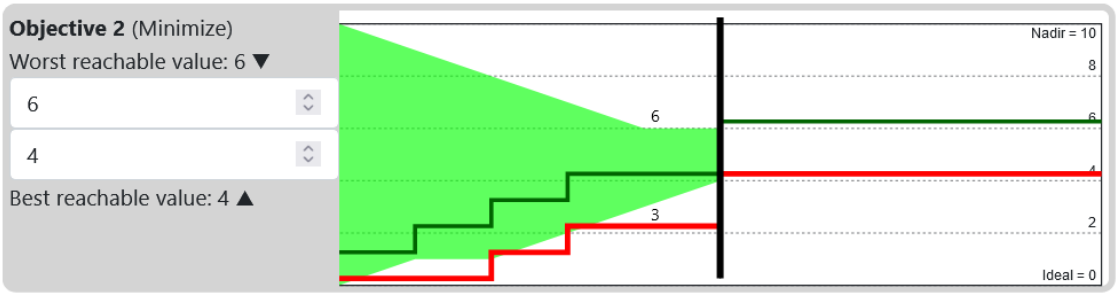


FIGURE 2 NAUTILUS Navigator interface showing the evolution reachable ranges as the step point moves closer to the Pareto front for one of the objectives.

The black line moves forward at a constant rate (set by the analyst). The DM can update their aspiration levels at any point by raising or lowering the green horizontal line. Similarly, they can provide strict bounds for objectives by setting the red horizontal line at the desired level. By setting the aspiration levels and the bounds, the DM can control which solution from the representative set of Pareto optimal solutions is chosen as the final one. They can also go back in time and put the step point at an earlier position by dragging the black vertical bar backward. This allows them to change their aspiration levels if they do not like how the reachable ranges evolved. As the black bar reaches the right edge at the end of the navigation, the reachable ranges for every objective shrink down to single values. These values correspond to the values attained by the Pareto optimal solution preferred by the DM.

2.3 Multiobjective evolutionary algorithms

We can broadly categorize MOEAs into three types based on their selection mechanism: dominance-based MOEAs [21], indicator-based MOEAs [35], and decomposition-based MOEAs [96]. Dominance-based MOEAs use the concept of dominance to select offspring. However, it becomes increasingly difficult to compare solutions using the dominance relation in MOPs with a large number of objectives [52], which impairs the optimization process. Some popular dominance-based MOEAs are NSGA-II [27] and SPEA2 [110]. Indicator-based MOEAs such as IBEA [109] use indicators such as hypervolume [111] and additive ϵ-indicator [112] to calculate the fitness of individual solutions for the selection operation. The calculation of these indicators also becomes computationally expensive with increasing number of objectives.

Decompositions-based MOEAs use the concept of reference vectors to divide the objective space into smaller regions. Each reference vector acts as a direction of improvement (usually relative to the ideal point) for a local subpopulation. The selection operator uses these reference vectors to generate scalar valued functions for each subpopulation which are then used to calculate the fit-

from previous implementations.

ness of the solutions. This enables the MOEA to effectively convert the MOP to a group of single-objective optimization problems, resulting in solutions close to the Pareto optimal front in the directions of the reference vectors. For *a posteriori* decomposition-based MOEAs, the number of reference vectors required to adequately represent the Pareto front grows exponentially with the number of objectives [52], thus incurring high computational costs. Popular MOEAs include RVEA [17], MOEA/D [106], and NSGA-III [8] [24]. A surrogate-assisted version of RVEA, K-RVEA [19], solves online data-driven MOPs effectively by using additional objective function evaluations to strategically make the surrogate models more accurate near the Pareto front.

Interactive MOEAs reduce the computational cost of solving MOPs with a high number of objectives by focusing on the DM's region of interest. Decomposition-based MOEAs provide a simple way of using DM preferences in an interactive fashion by limiting the spread of reference vectors towards a DM's region of interest [24, 46]. Unlike many interactive methods that only accept DM preferences in one format, interactive RVEA [46] enables the DM to provide their preferences in four different ways: a reference point, specifying preferred solutions, specifying non-preferred solutions, and specifying preferred ranges for the objectives. Indicator-based MOEAs also provide a straightforward way of creating interactive methods. MOEAs such as PBEA [93], which is an interactive version of IBEA, use a scalarization function which incorporates DM preferences in place of the indicator to calculate the fitness of individual solutions.

2.4 Visualizations to support DMs

To enable DMs to make decisions, we need to provide them with information regarding different Pareto optimal solutions of the MOP and the possible trade-offs. For *a posteriori* methods such as RVEA, the process can involve the visualization of thousands of solutions. The solutions are generally visualized in the objective space. We can use scatter plots for visualization in MOPs with two or three objectives. However, for MOPs with more objectives, we need to use visualization techniques such as a parallel coordinates plot. Parallel coordinates plots become messy and difficult to interpret with a large number of solutions. We expand upon this topic further in Chapter 5.

Even in simpler visualizations, however, comparing a large number of solutions is not a trivial task for any DM. Some visualization tools [47,95] support the DM by allowing them to interact directly with the plots to, for example, highlight or hide specific solutions and zoom in and out. These are called interactive or dynamic plots. We will use the latter term in this thesis to avoid confusion with the similarly named interactive methods.

Many interactive methods, such as NIMBUS, E-NAUTILUS, the reference point method, and Pareto race, avoid the issue of comparing thousands of solu-

[8] NSGA-III also the dominance relation in a way similar to NSGA-II.

tions by only presenting the DM with a small, manageable number (as low as one) of alternatives. The DM can then compare those solutions and provide their preference, leading to another set of alternative solutions. NAUTILUS Navigator avoids the same issue by visualizing reachable ranges of the objectives rather than individual solutions. Dynamic plots can help the DM even in methods where the number of alternative solutions is very small. In interactive methods specifically, we can use dynamic plots not just to explore alternative solutions, but also as interfaces that the DM can use to input their preferences. For example, the navigator plot shown in Figure 2 is a dynamic plot that allows the DM to provide aspiration levels and bounds, and go back to a previous step by dragging the green, red, and black lines in the plot respectively. The navigator plot also provides text boxes to accept input from the DM, if the DM wants to set an exact value for the aspiration level or the bound. Dynamic plots can make the process of providing preferences easier, compared to, for example, providing a reference point by typing out the values. This encourages DMs to experiment with the method and test out different preferences.

2.5 Properties of microalloyed steels

Structural materials, such as steels, can undergo temporary or permanent deformation under force [15]. If the structure springs back to its original shape and size after the removal of the external force, the deformation is termed elastic. Beyond a specific threshold force, the material can undergo plastic deformation, which permanently changes the structure's shape even after removing the force. Yield strength (YS) measures tensile stress[9] that a material can sustain while still being in the regime of elastic deformation. If the internal stress of the material exceeds YS, it undergoes plastic deformation. The ultimate tensile strength (UTS) is a similar metric. It measures the maximum tensile stress that a material can tolerate before undergoing a catastrophic fracture. In most applications, the structures are designed such that the material experiences stresses below YS.

YS and UTS depend on the chemical composition and the material's microscopic structure. They are usually measured by performing tension tests on specimens of a standard shape. The fractional increase in the length of the specimen after it is fractured (i.e., the internal stresses cross UTS) is termed percentage elongation. Another critical criterion that should be measured for many structural applications is the behaviour of materials at cold temperatures. Many materials, such as steel, which are very ductile at room temperature, can become brittle at low temperatures, increasing a structure's probability of failing and fracturing. This behaviour can be measured using the Charpy impact test, which measures the energy required to fracture a specimen of standard shape. The Charpy energy can be measured at various temperatures depending on the application's needs.

[9] Stress refers to force per unit area. Tensile stress arises from a force that "pulls" a material apart, as opposed to compression.

Microalloyed steels are alloys of steel which contain alloying elements like vanadium, niobium, and titanium in small amounts. The presence of these elements changes the microscopic structure of the steel and can lead to better values for the properties mentioned earlier. As they have a low amount of alloying elements, they also have a lower carbon equivalent value, which is measured as a weighted sum of the alloying elements present in the steel. The carbon equivalent value can be used to predict a material's easability to be welded.

3 AUTOMATIC SELECTION OF SURROGATE MODELLING TECHNIQUES

Cueball II: This is your machine learning system?
Cueball Prime: Yup! You pour the data into this big pile of linear algebra, then collect the answers on the other side.
Cueball II: What if the answers are wrong?
Cueball Prime: Just stir the pile until they start looking right.

Randall Munroe
Machine Learning

Solving data-driven MOPs, whether online or offline, often requires the usage of surrogate models. The models are generally used in place of objectives or constraints that may be difficult (time or resource expensive) or impossible (for offline data-driven problems) to evaluate during the solution process. An analyst can choose one or more surrogate modelling techniques for their MOP from a pool of hundreds of published methods [55]. Different surrogate modelling techniques can have vastly different performances on MOPs from different fields, different MOPs from the same field, or even different objectives or constraints within the same MOP. Therefore, the choice of surrogate modelling technique plays a crucial role in surrogate-assisted data-driven optimization. The choice can impact the performance of the optimization algorithm and the validity of the results themselves.

Therefore, it is discouraging that many studies investigating data-driven MOPs do not provide ample justification for the choice of surrogate modelling techniques [20]. In most cases, analysts use the techniques they are familiar with or are popular in the MOP's domain. In some cases, the analysts choose the surrogate modelling technique by conducting cross-validation testing of a small number of models using the limited data from the MOP they want to solve. This process requires training the various models multiple times and becomes increas-

ingly time-consuming with larger datasets and number of surrogate modelling techniques being considered. This issue is exacerbated in online data-driven problems, where the inclusion of newly sampled points in the dataset can warrant retesting of the surrogate modelling techniques.

The Surrogate Modelling Technique Selection (SMTS) algorithm, proposed in [PI], provides an efficient way to predict the best or near-best performing surrogate modelling technique for any given dataset. The SMTS algorithm does this without training any new surrogate models (except linear or quadratic models that are very quick to train). The field of automatic algorithm selection [58,82,90], which tackles a similar problem (choosing the best optimization algorithm), inspired the SMTS algorithm. The core idea behind the algorithm is the creation of a "selector" that can predict the optimal surrogate modelling technique for any MOP based on a standard set of features calculated from the associated dataset. In short, it is a machine learning model that predicts surrogate modelling techniques.

3.1 The SMTS algorithm

Figure 3 shows the general structure of the SMTS algorithm. The algorithm is divided into two phases: the training phase and the application phase. The training phase is a compute- and time-intensive step that trains the selector mentioned above. Once trained, we can make the selector publicly accessible for use by analysts to predict the best surrogate modelling techniques for their data-driven MOP. Using the selector (in the application phase) is much faster than conducting cross-validation because it involves a single prediction from the selector. To achieve this, we train the selector by (a) identifying patterns in large number of datasets and (b) correlating those patterns to the performance of surrogate modelling techniques.

Identifying the patterns in datasets

Data-driven MOPs can arise in a variety of fields. The datasets can be very small or large, sparse or dense, noisy or clean, uniformly distributed or skewed. The associated underlying objective functions can be linear and simple, complex but deterministic, or even stochastic. The selector must be able to distinguish between different datasets and identify such properties. One way to achieve this is to expose the selector to diverse datasets during the training phase. We suggest a technique to generate thousands of datasets with a diverse set of characteristics using well-known benchmarking MOPs in [PI].

Most machine learning techniques (i.e., selector candidates) cannot accept datasets of arbitrary sizes as their input. Therefore, we first calculate a fixed number of features from each dataset. We use the exploratory landscape analysis features [69], which have been successfully used in the literature to select

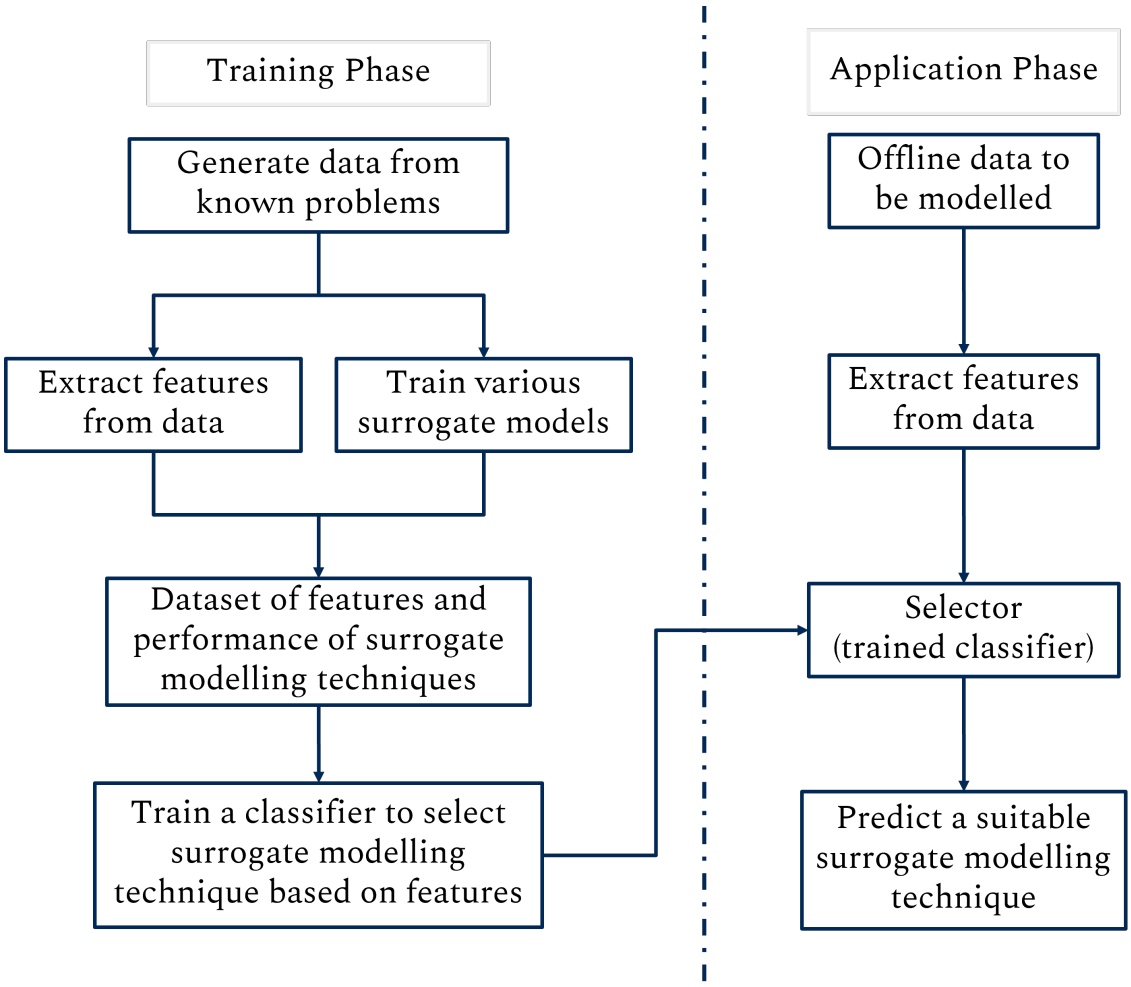


FIGURE 3 SMTS algorithm to train the selector for automatic selection of surrogate modelling techniques.

optimization algorithms for MOPs automatically [58, 90]. The selector can use these features to identify and distinguish between datasets.

Identifying the best surrogate models

We must first create a shortlist of the best surrogate modelling techniques for the selector's consideration. This list should be diverse and represent popular techniques from various domains. In [PI], we consider techniques such as neural networks, support vector machines, Kriging, and ensemble models as implemented in the Python package `scikit-learn` [80]. The next step is relatively simple but very time-consuming. We train models using every shortlisted surrogate modelling technique on every dataset generated in the previous step. We need to define what the "best surrogate modelling technique" means. In [PI], we use the R^2 accuracy metric as a measure of goodness. However, many other metrics can be used, including the combined performance of a surrogate modelling technique and an MOEA [111, 112]. Using the latter metric will significantly increase the time consumption of the algorithm, however.

Training the selector

We have features and surrogate modelling technique performances associated with thousands of datasets by the end of the previous step. We can now use a classification algorithm to train a selector to predict the best performing techniques based on the features. However, no classification algorithm can predict the best algorithm a hundred per cent of the time. In the worst-case scenario, the selector may predict the worst technique for the MOP. To counteract this, we propose a custom cost function in [PI] which ensures that the selector is trained to predict techniques in the order of their performance. If the selector fails to predict the best performing surrogate modelling technique, it will likely predict the second or third best technique rather than the worst technique.

We implemented the SMTS algorithm in Python using `NumPy`, `pandas`, and `scikit-learn` packages. The training phase took close to twenty hours to complete on a machine with AMD Ryzen 5 2600 six-core CPU and 8 GBs of RAM. Once the model is trained, however, an analyst can calculate the features of their MOP's dataset and use it with the selector to predict the (close to) best performing surrogate modelling technique instantaneously.

3.2 Discussion about the SMTS Algorithm

In [PI], we present multiple candidate selectors using different classification algorithms to predict the best surrogate modelling techniques for datasets of MOPs. We trained the selectors using thousands of benchmarking MOP datasets, tested them on hundreds of benchmarking MOP datasets (unseen by the selector during

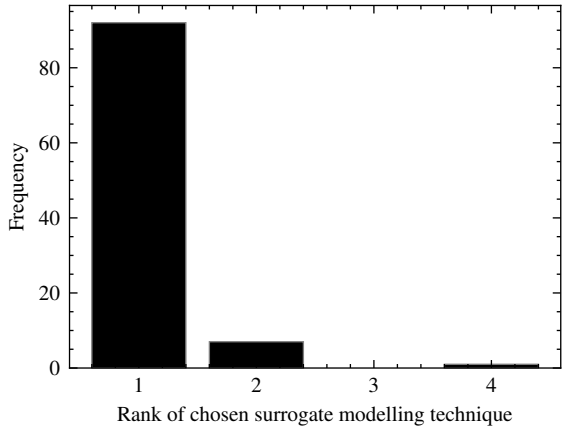


FIGURE 4 Frequency of ranks achieved surrogate modelling techniques chosen by the selector for the testing datasets.

the training phase), and validated them using datasets from engineering MOPs entirely unrelated to the datasets used during the training phase. We compared nine different classification algorithms as candidates for the selector, and each candidate selector had the option to choose from ten different surrogate modelling techniques. In our tests, the "extra-trees classifier" as implemented in the `scikit-learn` Python package had the best overall performance as the selector. As a by-product of training ten different surrogate modelling techniques on thousands of datasets, we also discovered that the "gradient boosted regressor" as implemented in the `scikit-learn` package was the best performing surrogate modelling technique on most datasets.

In this section, we conduct a simpler version of the experiment done in [PI] and create a selector using the "extra-trees classifier". We only consider four (popular) surrogate modelling techniques for the selector: "gradient boosted regressor", "neural networks", "gaussian process regression", and "support vector machines", as implemented in the `scikit-learn` package with default hyperparameter values. We use 500 randomly selected datasets from the datasets created for [PI]. We use 400 datasets to train the selector and the remaining 100 to test it.

Figure 4 displays the results of the experiment. Out of the 100 test datasets, the trained selector predicted the best surrogate modelling technique for 92 datasets. It predicted the second-best surrogate modelling technique for seven datasets and chose the worst out of four surrogate modelling techniques for one of the datasets. The selector is very successful at predicting the best surrogate modelling techniques. This selector, however, is inappropriate for real-life MOPs as it was trained on a minimal set of datasets, with very few options for surrogate modelling techniques. In [PI], however, we provide a much more general selector which performs well over a variety of datasets. The selector provides analysts unfamiliar with various surrogate modelling techniques a quick way to discover good modelling techniques for their MOP, leading to better solutions.

4 INTERACTIVE OPTIMIZATION IN THE PREFERENCE INCORPORATED SPACE

We've had one, yes. But what about second breakfast?

Peter Jackson
The Fellowship of the Ring

Calculating a representative set of solutions to approximate the Pareto front for an MOP with a *a posteriori* MOEA becomes exponentially more challenging with an increasing number of objectives [28,52]. This leads to increased time consumption and worse results; the MOEA can even fail to discover entire sections of the Pareto front [PVI]. Interactive MOEAs [7] can circumvent the issue by focusing on a small region of the objective space, the region of interest of a DM. This allows interactive MOEAs to focus computational resources on quickly converging in the region of interest, saving time and function evaluations (which may be limited in some MOPs). As the methods are interactive, they also reduce the cognitive load on the DM by not requiring them to analyze the entire approximated Pareto front at once, as is the case with *a posteriori* MOEAs.

However, interactive MOEAs suffer from many problems. Most interactive MOEAs are modified versions of their *a posteriori* counterparts [7]. This adds additional parameters to the MOEA to control how the preferences of a DM are utilized within the method. However, this utilization may not be straightforward, making it difficult for a DM to control the optimization process. Additionally, by allowing the MOEA to focus primarily or entirely on the region of interest, we dramatically reduce the diversity of the population (compared to a population in a *a posteriori* MOEA). This may lead to issues during the optimization process, especially when the DM changes their preferences [PII, 7, 29].

An alternative way to solve MOPs interactively is to use scalarization-based methods [70, 73, 75]. However, as these methods generally use only a single scalarization function at a time, they return one solution at the end of each iter-

ation[1]. The interpretation of preference information differs considerably among various scalarization functions. Thus, the choice of scalarization function can significantly impact the optimization process and the final chosen solution. But choosing the best scalarization function for an MOP and a DM is not a simple task. Some scalarization-based methods solve this issue by utilizing multiple scalarization functions at each iteration [76]. However, these still only return a few solutions at a time. Having a variety of solutions within the region of interest can benefit the optimization process. It gives the DM more information about the trade-offs among the objectives in the region of interest and presents a higher chance of finding an acceptable solution.

The WASF-GA MOEA [83,85] tackles the issue by using a reference point to identify a region of interest. The algorithm then uses several achievement scalarizing functions with the reference point as the preference information and a set of uniformly-distributed weight vectors ($\boldsymbol{\mu}$) to find solutions in the region of interest. WASF-GA conducts evolution in a manner similar to the decomposition-based algorithms described in Chapter 2, and the algorithm's performance suffers with an increasing number of objectives.

It is, therefore, desirable to create a method that has the benefits of MOEAs and scalarization-based methods while avoiding their downsides. Thus, we see a need for an interactive method that can (a) find a diverse set of solutions in (b) a region of interest not limited to a singular, strict interpretation of preferences and (c) free from the issues related to the number of objectives. In [PII], we introduce not just such a method but a methodology for creating such methods, thereby introducing a new paradigm in interactive evolutionary multiobjective optimization. We achieve this by using multiple (as low as two) scalarization functions to create a new space, which we call a preference incorporated space (PIS). The different scalarization functions interpret the same preference information differently leading to different optimal solution for each individual scalarization function (with infrequent exceptions) [75]. This leads to a trade-off between the different scalarization functions, leading to a Pareto front in the PIS. We then use any *a posteriori* MOEA to optimize in the new space instead of the objective space.

4.1 Interactive Optimization using Preference Incorporated Space (IOPIS) algorithm

We formally define the PIS by extending the MOP (1) as:

$$\underbrace{\mathbf{x}}_{\substack{\text{Decision}\\ \text{space}}} \xrightarrow{\mathbf{f}} \underbrace{\mathbf{f}(\mathbf{x})}_{\substack{\text{Objective}\\ \text{space}}} \xrightarrow[\bar{\mathbf{z}}]{\mathbf{s}} \underbrace{\mathbf{s}(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}})}_{\substack{\text{Preference}\\ \text{incorporated}\\ \text{space}}} \tag{3}$$

[1] Some scalarization-based methods such as synchronous NIMBUS [76] use multiple scalarization functions and return multiple solutions. However, these scalarization functions are optimized independently of each other.

where $\mathbf{s}$ is a set of q scalarization functions, and $\bar{\mathbf{z}}$ is the preference information given by the DM. We first need to establish some properties that the PIS should have to ensure that optimizing in this new space is worthwhile. We formulated these properties by modifying the desirable properties of scalarization functions as stated in [88] and mentioned in Chapter 2. We state the desirable properties of the PIS in [PII] as:

1. **Optimality**: Pareto optimal solutions in the PIS remain Pareto optimal in the objective space.
2. **Preferability**: Pareto optimal solutions in the PIS follow the preferences given by the DM in the objective space.
3. **Discoverability**: Any properly Pareto optimal solution of the MOP can be discovered by changing the preferences.

We prove in [PII] that if we construct the PIS using achievement scalarizing functions, all of the properties are satisfied. Optimizing in the PIS provides us with the following benefits compared to optimizing in the objective space:

1. Control of the number of dimensions: We can control q independently of k by simply changing the number of scalarization functions used to construct the PIS.
2. Reducing the number of function evaluations: The number of solutions needed to represent a Pareto front increases exponentially with the number of dimensions of the objective space (for other MOEAs) or the PIS (for IOPIS). By setting a low value for q, we can use IOPIS with a much smaller population size, independent of the number of original objectives.
3. Modular creation of interactive MOEAs: By using the PIS, we incorporate the preferences directly in the MOP. This allows us to use any *a posteriori* MOEA (or a biobjective EA if q is two) with the PIS, functionally making it an interactive MOEA.
4. Control over the interpretation of preferences: Trade-off between the Pareto optimal solutions in the PIS represents the trade-off between the interpretations of preferences by the scalarization functions that form the PIS. Hence, we can control the interpretation by adding or removing scalarization functions. This interpretation is independent of the choice of MOEA used for optimization. This has the added benefit of allowing easy comparison of different MOEAs in the context of interactive evolutionary optimization, an otherwise challenging task. For example, we can compare how effective different MOEAs are in optimizing in the same PIS.

We show the structure of the IOPIS algorithm in Figure 5. We assume that an analyst and a DM have worked together to construct an MOP. We also assume that the analyst has chosen a set of appropriate scalarization functions to construct the PIS. The IOPIS algorithm then begins with an analyst asking the DM to provide their preferences. The analyst can support the DM in this step by visualizing some sample solutions or providing information about the approximated ideal

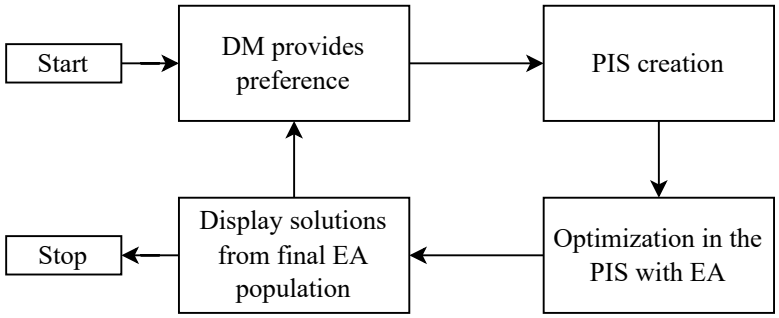


FIGURE 5 Illustration of the IOPIS algorithm

and nadir points. The algorithm then creates the PIS using the provided preferences and the set of scalarization functions. An appropriate MOEA is then used to solve the problem in the PIS, resulting in an approximation of Pareto optimal solutions in the region of interest of the DM. The analyst then visualizes these solutions[2] to the DM. If the DM is satisfied with one of these solutions, they can choose to terminate the solution process here. Otherwise, they can provide new preferences to the algorithm, starting a new iteration and continuing the process until they find a satisfactory solution.

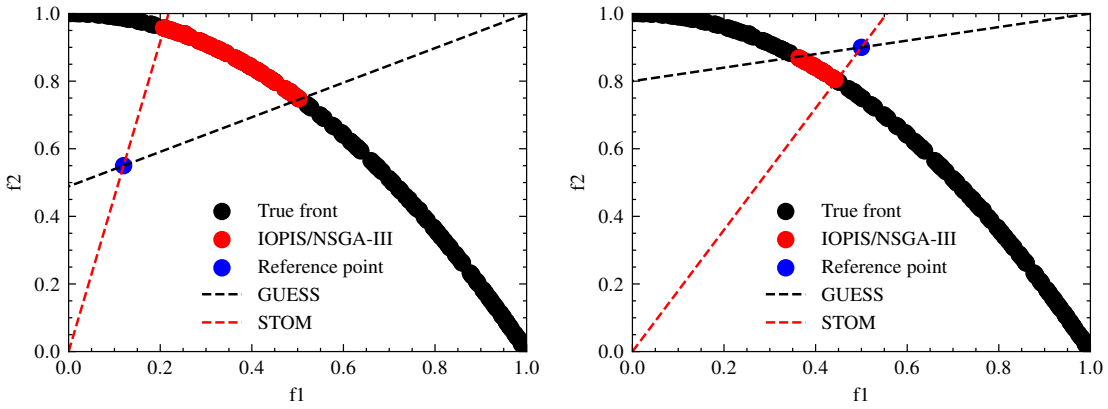
4.2 Discussion about the PIS and IOPIS

In [PII], we showcase two implementations of the IOPIS algorithm: IOPIS/RVEA and IOPIS/NSGA-III. Both algorithms use the STOM [79] and GUESS [13] achievement scalarizing functions to form a two-dimensional PIS. These accept the DM's preferences in the form of a reference point. IOPIS/RVEA uses RVEA to conduct the optimization in the PIS, whereas IOPIS/NSGA-III uses NSGA-III. We discuss the implications of optimizing in the PIS for the DM and show how a DM can control the interactive optimization process.

We show the effect of using STOM and GUESS functions to create the PIS to solve the ZDT2 [108] in Figure 6. The two scalarization functions are represented by dashed lines[3] in equation (2). The DM can "zoom out" and expand the region of interest by setting the reference point far from the Pareto front. This can happen naturally during the first few iterations when the DM does not have enough information about the Pareto optimal solutions. IOPIS provides a general overview of solutions spanning a large region. With the help of this new information, the DM can "zoom in" and shrink the region of interest by placing the reference point close to the Pareto front. On the other hand, one can say that the DM learns about how reachable the reference point is. If a wide range or solutions is given, the reference point is far from Pareto optimal solutions while few

[2] generally in the objective space, although certain domain-specific visualizations may also visualize decision variables

[3] More precisely, the dashed lines are the directions of improvement for the two scalarization functions as defined by $\boldsymbol{\mu}$ and $\bar{\mathbf{z}}$



(a) Reference point is far from the front.

(b) Reference point is close to the front.

FIGURE 6 Effect of a reference point on the solutions returned by IOPIS-NSGA-III for ZDT2.

solutions near the reference point are shown in the opposite case. In this way, the interpretation corresponds to the reasoning of the reference point method [101].

We test the performance of IOPIS/RVEA and IOPIS/NSGA-III against *a posteriori* RVEA and NSGA-III, and their interactive variants that we implemented as described in [46]. We conducted the tests using close to six hundred MOPs generated using the DTLZ and WFG benchmark problems [30,50]. We randomly generated reference points in the objective space to simulate a DM. The IOPIS variants of RVEA and NSGA-III generally performed better than their *a posteriori* or interactive but non-IOPIS variants for most of the considered MOPs. They achieved better convergence towards the Pareto optimal solutions in the region of interest of the (artificial) DM while consuming significantly fewer objective function evaluations (fewer by two orders of magnitude in some cases).

IOPIS successfully combines the benefits of MOEAs and scalarization-based methods. While in [PII], we only implement versions of IOPIS that accept a reference point from a single DM as the preference information, the algorithm can create interactive MOEAs to suit a variety of solutions. If the DM prefers to give their preference in a different format, an appropriate scalarization function can be incorporated into the PIS to handle that. For example, the NIMBUS scalarization function can enable expressing the preferences in the form of classification of objectives [76]. We will discuss another use of the IOPIS algorithm in Chapter 8, where we use IOPIS to create an interactive MOEA to support multiple DMs in group decision making.

5 VISUALLY APPEALING AND INFORMATIVE VISUALIZATIONS FOR DECISION MAKERS

People are not alarmed by the unusual so long as it is placed in an acceptable context.

Michael Moorcock, The Land Leviathan

Visualization of solutions of MOPs is not a trivial task [42, 60, 64, 71]. A scatter plot of objective values obtained by all solutions works well for biobjective problems. Scatter plots can be used for MOPs for a higher number of objectives by encoding information about some of the objectives as the markers' colour and size or by using multiple scatter plots displaying different combinations of objectives in a scatterplot matrix. However, these can be challenging to interpret with an increasing number of objectives. Dimensionality reduction techniques can enable visualization using a smaller number of dimensions while conserving the structure of the original data [11, 98]. However, such visualizations cannot visualize the actual objective values and thus can not act as the primary means of conveying information to the DM.

Parallel coordinate plots can create interpretable visualizations of high-dimensional data, as long as the number of alternative solutions is small [41]. Interactive MOEAs, which can accumulate thousands of solutions in a few iterations, can lead to messy and hard-to-interpret visualizations. "Dynamic" plots [1], which allow a DM to tweak the visualization in real-time, can help the DM interpret the information. Such plots can allow the DM to reorder the axes to check the trade-offs between different objective pairs. Moreover, dynamic plots allow the DM to filter solutions, helping them focus on a subset of solutions at a time. While these tools can greatly help in decision making, using only them for analysis can take a long time, especially if the MOP has thousands of solutions with ten or more objectives. Moreover, for these tools to be effective, a DM must be familiar with them, which is not always the case.

[1] https://d3js.org/

Thus additional tools to support a DM in analyzing such visualizations are needed. In [PIII], we introduce such a tool: the SCORE bands visualization technique. SCORE bands augments the traditional parallel coordinates plot by (a) ordering the objectives in an arrangement that brings out helpful information to a DM, (b) encoding information regarding the patterns in objectives (such as correlated objective groups and trade-offs) by varying the distances between objectives in the plot, (c) visualizing solution clusters as bands flowing through the visualization. By default, we do not visualize the individual solutions at the beginning. Instead, the bands give the DM a general overview of the trends in the solutions and the objectives, allowing them to identify regions of interest. The DM can then hide the other bands and display the individual solutions of their preferred clusters by interacting with the plot. SCORE bands visualizations provide simple yet information-dense visualizations and gives a DM head start in analyzing solutions. In other words, they support the DM in focusing at a smaller amount of information at a time and seeing bigger trends more conveniently.

5.1 Solution clustering and correlated objectives visualization via bands (SCORE Bands)

The SCORE bands algorithm augments a parallel coordinate plot in four steps as stated in [PIII]:

1. **Solution clustering**: We cluster the solutions to differentiate the clusters using different colours. Using clusters in visualization is not a novel idea [14,105]. However, it is a necessary step and works in tandem with the later steps to create the SCORE bands visualization. The clustering information can come from, for example, using clustering algorithms on objective values, decision variable values, or a combination or a subset of them. Alternatively, solutions from different iterations of an interactive methods can be put into different clusters.
2. **Axis ordering**: We identify the optimal ordering of the objectives for the parallel coordinates plot. Related ideas have been proposed [5,107] before, leading to varying degrees of success. We treat the problem of ordering the objectives as a simple travelling salesperson problem: objectives are the "cities" of the problem, and the correlations between objective pairs are used to calculate the "distance" which the salesperson uses to find the optimal order of travel. Obtaining the order in such a manner ensures that correlated objectives are placed together in the visualization, highlighting their patterns. This is very useful in static visualizations, such as plots in print media, as it immediately illustrates important information to DM. The step is still helpful for dynamic visualizations, providing the same advantage to the DM as the static plots while giving them the option to reorganize objectives for further analysis.
3. **Axis placement**: Traditional parallel coordinates plots encode much infor-

mation vertically (all axes are vertical; thus, the height of a trace signifies the objective value), but not horizontally. We increase the density of useful information provided to the DM by varying the distance between the objective axes in SCORE bands. This allows us to place highly correlated objectives close together while increasing the space between otherwise uncorrelated neighbouring objectives. This simple yet novel idea enhances a DM's analysis of solutions from interactive methods by providing extra information that can be visually interpreted at a glance.

4. **Solution visualization**: The major source of difficulty in understanding parallel coordinate plots comes from the number of solutions (which can be in the thousands, as shown in Figure 7). We alleviate the difficulty by not displaying the individual solutions by default. Instead, we visualize the clusters as bands, finally forming the SCORE bands visualization. The height of a band at each objective axis represents the median objective value of the associated cluster. The width represents a statistical measure of spread. We use the interquartile range as the default way to calculate the width. We use the clustering information to colour the bands using translucent colours. This allows the DM to discern and notice the trade-offs between the different clusters.

SCORE bands visualizations are dynamic, so they allow the DM to show or hide the cluster bands and the solutions belonging to those clusters. By not visualizing the solutions initially, we introduce a natural break in the decision making process. The DM first notices the bands and the groups of objectives (in case some of them are correlated) and immediately begins to absorb information about the trade-offs. We keep the initial plot visually simple and allow the DM to introduce complexity consciously by enabling them to show or hide solutions. This prevents the DM from being overwhelmed by information and allows them much more control over the visualization than traditional parallel coordinates plot.

5.2 Discussion about SCORE Bands

In [PIII], we implemented the SCORE bands visualization using the Plotly Python package [81]. The implementation provides multiple options for each of the four steps mentioned earlier, allowing an advanced user (perhaps an analyst) to fine-tune the visualization. We also implemented a graphical user interface using the Dash Python package [89], which makes those options available in a user-friendly application. The interface allows users to import their datasets and experiment with the visualization. Besides SCORE bands, it provides additional supporting visualization, such as scatter plots created using dimensionality reduction techniques [11,98]. However, we recommend that DMs and analysts work together to create MOP-specific supporting visualizations. For example, for a vehicle chassis design problem, a supporting visualization can show the chassis corresponding to a solution selected by the DM in the primary SCORE bands visualization.

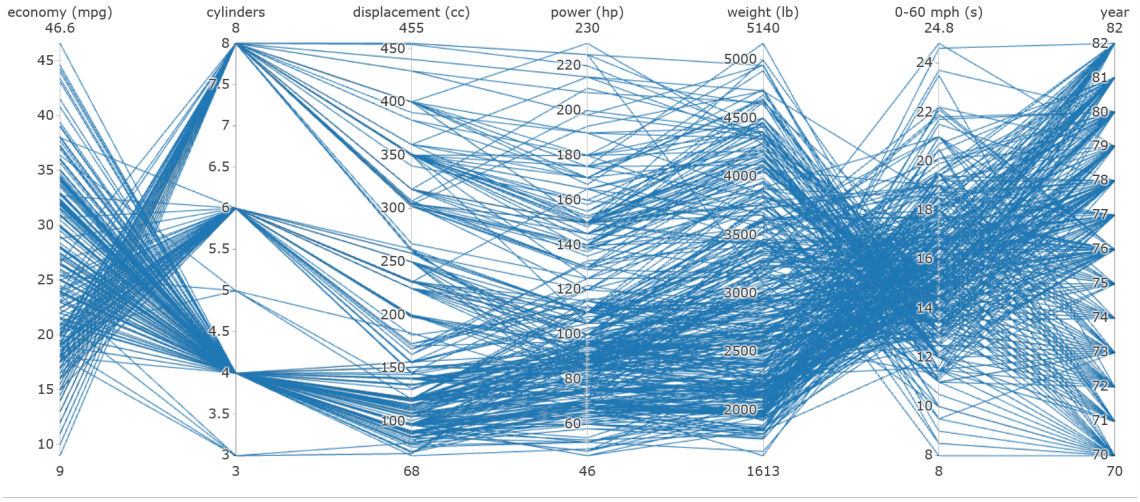


FIGURE 7 The auto mpg data set visualized using a traditional parallel coordinates plot.

We showcase the utility of each of the steps of the SCORE bands visualization using datasets from benchmark problems in [PIII]. We also present multiple case studies using artificial datasets (which highlight the usefulness of SCORE bands), and data from a real-life MOP. Here, we demonstrate the SCORE bands plot using a dataset about cars, the auto mpg dataset, obtained from UCI machine learning repository [31]. The dataset contains details like seven numeric columns detailing attributes (such as fuel efficiency, horsepower, and weight of the car) of about four hundred cars. The person choosing a car mentioned in Chapter 1 would perhaps use a similar dataset. We skip the optimization details (which attributes are objectives, whether they are maximized or minimized, and so on) as they are not necessary to showcase SCORE bands. However, we assume that the dataset to be visualized contains non-dominated solutions only.

Figure 7 visualizes the auto mpg data set using a parallel coordinates plot. We can immediately see that the "cylinders" objective has a small number of discrete values represented in the solutions. The "cylinders" objective and the "economy (mpg)" objective are inversely correlated, represented by the crossing over of the solution traces. However, the static visualization does not immediately clarify the further relationship between the various objectives. Extracting this information from a dynamic visualization would require thorough analysis.

Figure 8 presents the same dataset as a SCORE bands visualization. The plot is significantly less cluttered than the parallel coordinate plot. A DM can see that the solutions are clustered according to their "cylinders" value. SCORE bands rearranges the objectives and puts "power (hp)", "cylinders", "displacement (cc)", and "weight (lb)" together in a tightly packed group. This visual cue signifies to the DM that those objectives are positively correlated. The DM can confirm this by looking at the bands in those four objectives. The bands rarely cross over one another, signifying a positive correlation. The other objectives are places farther apart and thus have a very low or negative correlation.

The DM can intuit the correlations between any two objectives, even those

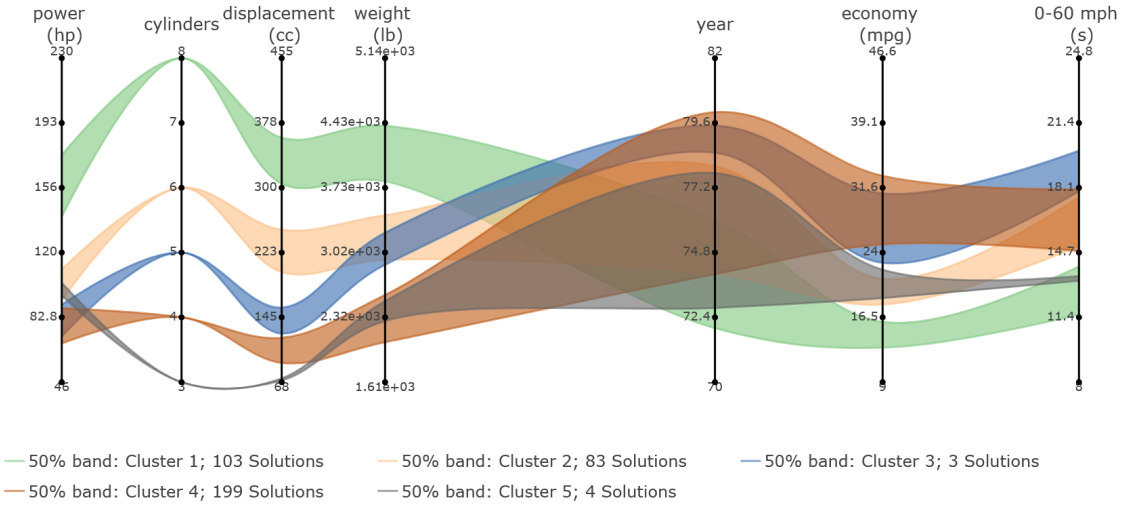


FIGURE 8 The auto mpg data set visualized using SCORE bands.

that are not neighbouring, by looking at the vertical order of the bands in those objectives. For example, the green, light brown, and dark brown bands switch order between the "cylinders" and "economy (mpg)" objectives, signifying a negative correlation. The blue and grey bands only contain three and four solutions, respectively (as mentioned below the visualization), and therefore do not contribute to the correlation significantly. The DM can gather this information by a simple visual inspection of the static visualization. Dynamic SCORE bands enable the DM to show the solutions or hide specific clusters, allowing further step-wise and intuitive analysis.

SCORE bands successfully tackle the challenge of supporting DMs in interactive multiobjective optimization methods by giving them helpful insight into the solutions. It simplifies and streamlines the decision making process and gives the DM control over the level of complexity of the visualization. It has advantages over the traditional parallel coordinates plot in both static and dynamic formats. The SCORE bands algorithm, our implementation, and the associated graphical user interface are open-source, giving DMs and analysts alike easy access for experimentation and further development.

6 HANDLING COSTLY FUNCTION EVALUATIONS WITH INTERACTIVE MULTIOBJECTIVE OPTIMIZATION

I am doing 1000 calculations per second and they are all wrong.

@shenanigansen, Shen Comix

Online data-driven MOPs raise unique challenges for interactive methods. These function evaluations may take a long time (and be financially expensive), introducing breaks in the decision making process. Surrogate-assisted MOEAs, such as K-RVEA [19] tackle online data-driven MOPs by conducting additional objective function evaluations strategically during the optimization process to make the surrogate models more accurate. In surrogate-assisted interactive methods, the surrogate models may not be very accurate during the first few iterations, especially close to the Pareto front. Naively constructed interactive MOEA exhibiting such behaviour may hinder the process of interactive optimization and discourage the DM. The DM may find "good" and preferred solutions when the interactive MOEA uses (inaccurate) surrogate models. They will then have to wait for the validation of those solutions by new objective function evaluations. In the end, they may discover that those solutions are either suboptimal or not in the region of interest or both, wasting the costly function evaluation.

One way to circumvent the issue is to involve the DM for interaction after ensuring that the surrogate models are accurate near the Pareto optimal front. However, this process misuses function evaluations trying to make surrogate models more accurate in regions where the DM may not be interested. The scalarization-based interactive method NAUTILUS Navigator [84] ultimately works with this assumption. It assumes the availability of a pre-generated representative set of Pareto optimal solutions. The set allows a DM to conduct swift interactive optimization even in MOPs with costly functions since no function evaluations are conducted when the DM is involved. However, creating such a

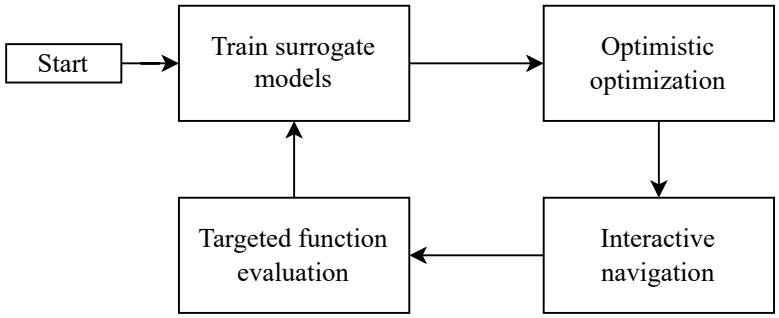


FIGURE 9 General flow of the O-NAUTILUS method

set, for example, with K-RVEA, still consumes many more function evaluations than necessary.

To resolve the issues of MOPs with costly function evaluations and enable functional interaction to solve such MOPs, we introduce the Optimistic NAUTILUS Navigator (O-NAUTILUS) method in [PIV]. The method combines MOEAs, surrogate models that can relay uncertainty information regarding their predictions, and scalarization functions to achieve its goal. Unlike the NAUTILUS Navigator, this method does not require a representative set of Pareto optimal solutions to begin interaction. Instead, it can use any data (even if it is suboptimal) that a DM may already have from past experiments to create surrogate models. The method creates "Optimistic" representations of the Pareto optimal front using the models and provides the uncertainty information to the DM in an interface similar to NAUTILUS Navigator. The DM can interact in confidence and find a preferred approximate solution with this additional knowledge. The method enables the DM to conduct function evaluations targeted in their region of interest. If the DM finds a satisfactory solution, they may end the process here. Alternatively, the method uses the targeted evaluation to make the models much more accurate in the region of interest, leading to better interaction and a higher chance of finding preferred solutions. Function evaluations are conducted sparingly, only when they are needed.

6.1 The O-NAUTILUS method

We show the major components of the O-NAUTILUS method in Figure 9. In [PIV], we provide extensive details about the algorithms involved in those components, as well as implementation details such as algorithm settings and hyperparameter values. In the following text, we provide a brief description of the components and deliver a basic understanding of the O-NAUTILUS method.

Training surrogate models

We begin the O-NAUTILUS method by acquiring data corresponding to the MOP. This data should include some solutions (not necessarily Pareto optimal), i.e.

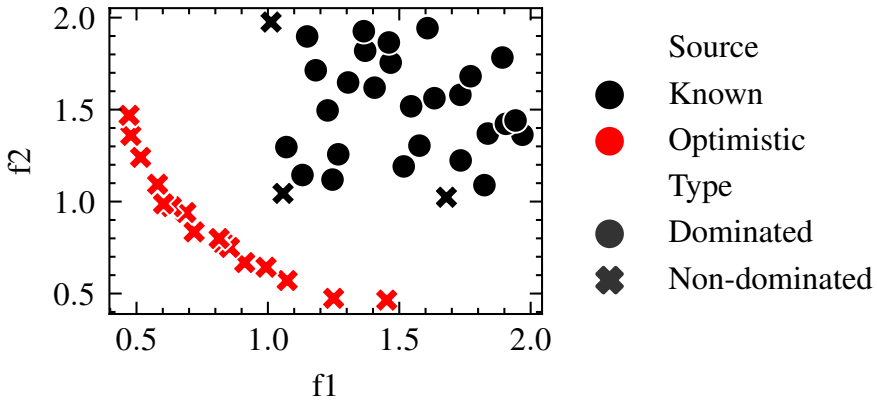


FIGURE 10 The known set of solutions, known front, and optimistic front for a biobjective optimization problem.

pairs of decision and objective vectors, of the MOP. The DM may already have access to such data. If the DM does not have access to the necessary data, the analyst and DM can discuss an appropriate number of function evaluations to spend for its generation. As the data does not need to be Pareto optimal, this data can be sampled randomly or uniformly, for example, using Latin hypercube sampling, thus requiring significantly fewer function evaluations than optimization. The case study described in [PIV] uses 100 initial samples. This data forms the known set of solutions. The non-dominated solutions from this set form the known front.

O-NAUTILUS requires surrogate modelling techniques that can give an uncertainty quantification or a bound on the predicted objective values. In [PIV], we use Gaussian process regression (which can give uncertainty quantification in terms of standard deviation from the predicted mean value) and Lipschitzian models (which can give exact bounds on the predicted mean value, assuming that the objective function is Lipschitz continuous). Other surrogate modelling techniques which provide similar information, such as random forest regression models [99], can also be used. We train the surrogate models using the known set of solutions.

Optimistic optimization

We use the trained models with MOEAs[1] to get a so-called optimistic approximation of the Pareto optimal front, or optimistic front. We obtain this by not using the predicted objective values from the surrogate models directly. Instead, we first subtract (or add, in case of maximization) the standard deviation/Lipschitz bound from the predictions. This optimistic prediction is then optimized, generating the optimistic front. To a DM, this front, when compared to known set, represents what can be achieved with function evaluations. In [PIV], we use RVEA to conduct optimization as it works well for MOPs with a high number of objectives. At the end of this step, we get two sets of fronts, as shown in Figure 10

[1] In principle, any *a posteriori* method can be used.

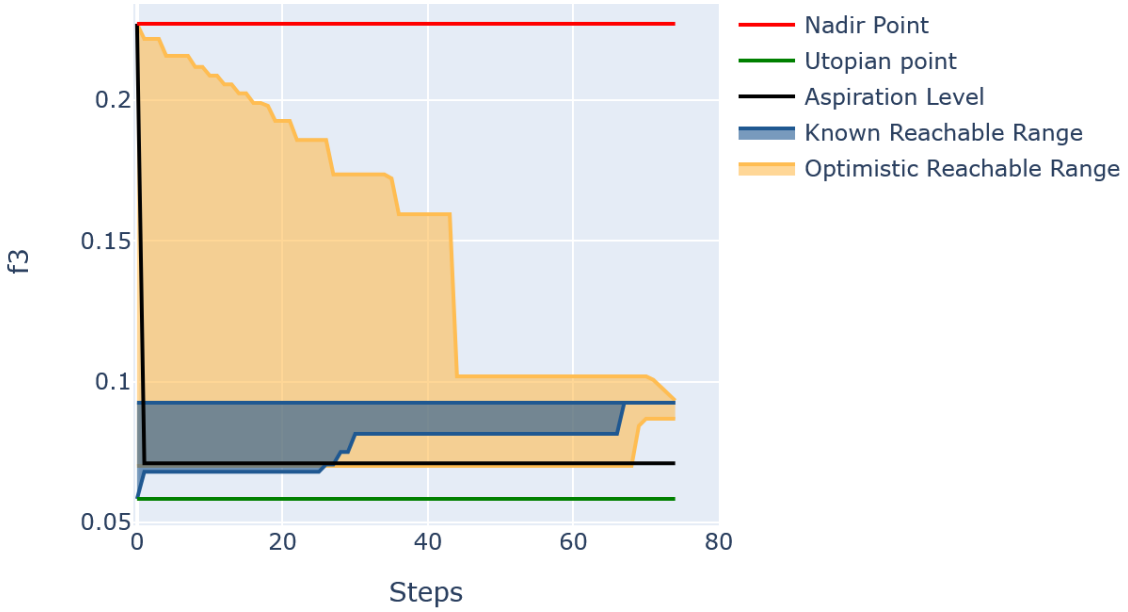


FIGURE 11 A part of O-NAUTILUS graphical user interface showing the known and optimistic reachable ranges for one of the objectives of an MOP.

for a bi-objective MOP. The black points in the figure are the known set of solutions. The black crosses are the known front. We show the optimistic front as red crosses.

Interactive navigation

After obtaining the two fronts, the O-NAUTILUS method enables the DM to interact using a navigation interface similar to that of the NAUTILUS Navigator. A portion of the O-NAUTILUS interface showing the visualization of one of the objectives for an MOP is shown in Figure 11. O-NAUTILUS begins navigation by putting a "step point" at the infinimum of the combined set of known and optimistic fronts. We call this the combined set nadir point, or, simply, the nadir point in Figure 11. The step point moves closer to the two fronts in a direction calculated using the preferences given by the DM in the form of aspiration levels. At each step, the reachable ranges are calculated independently for the known and optimistic fronts. We calculate the value by finding the minimum and maximum objective values attained by the solutions that still dominate the step point. The reachable range for the known front is represented by a blue shaded region in Figure 11, whereas an orange shaded region represents the reachable range for the optimistic front.

The step point moves closer to the known front at a constant rate (set by an analyst, though we provide a default value). The DM can pause the navigation to update their aspiration levels based on the changes in the known and optimistic reachable ranges. The DM can also go back to a previous step and change their aspiration levels to navigate in a new direction. The navigation ends once the step point reaches the known front and the known reachable range collapses to a single value. This means that a nondominated solution has been reached and

no improvement is possible any more in all objectives simultaneously. In Figure 11, this happens near step number 70[2]. The optimistic reachable range has not collapsed, signifying the DM that they can possibly reach better objective values in this region by conducting an additional function evaluation targeted toward their aspiration levels.

Targeted function evaluation

We use the achievement scalarizing function (ASF) described in Equation (2) to determine the best decision variable vector to evaluate such that the resulting solution is likely to be close to the Pareto front and follows the preferences of the DM. The ASF is a scalarizing function that can take aspiration levels as the preference information. We optimize the expected value of the ASF while using surrogate models to find the best candidate solution. As the ASF is a scalarizing function, the problem is a single-objective optimization problem. We use CMA-ES [48], which is a state-of-the-art solver for such problems, to find the best candidate solution. The solution is then evaluated using the costly objective functions. Note that this is the only step that involves a true function evaluation, and only a single true function evaluation is used.

If the newly evaluated solution is acceptable to the DM, they can stop the optimization process here. Alternatively, they can continue the process, in which case O-NAUTILUS will retrain the surrogate models with the newly evaluated point included in the training data. This retraining increases the accuracy of the surrogate models in the region of interest of the DM. The optimistic front is then calculated again, after which the DM can continue interacting with the method and navigate. At the end of the navigation, if the DM concludes that the remaining optimistic reachable range is not promising enough to spend function evaluations, they can stop the optimization process. If they are satisfied with the solution at the end of navigation, they can choose it as the final solution of the MOP. They can also push the step point backwards and give new preferences to get new reachable ranges. The process continues until the DM finds an acceptable solution.

6.2 Discussion about the O-NAUTILUS method

We implemented the O-NAUTILUS method using the DESDEO framework and a graphical user interface using the Dash Python package [89]. In [PIV], we used the O-NAUTILUS method to solve a real-world design optimization problem named the crash-worthiness design of vehicles problem [63]. It is an MOP with three objectives, optimizing the frontal structure of a car to improve the safety of passengers. We also used a combination of K-RVEA (to generate a representative

[2] The step number at which the step point reaches the known front can vary between different MOPs. This number is not relevant to a DM.

Pareto front) and NAUTILUS Navigator (to conduct interactive optimization) to solve the same MOP. We chose K-RVEA as it uses Kriging, same as O-NAUTILUS, and NAUTILUS Navigator because it provides the DM with a similar interface for interactive navigation. We gave the same total function evaluation budget to O-NAUTILUS and K-RVEA/NAUTILUS Navigator.

Because of the ability of O-NAUTILUS to conduct targeted function evaluations, the DM was able to find better solutions for the MOP than the K-RVEA/NAUTILUS Navigator combination. The DM was also able to make more informed decisions with the help of the uncertainty information provided by the visualization of the optimistic reachable ranges. The DM was also in direct control of when or whether to conduct a function evaluation.

However, this ability to make informed decisions to efficiently solve data-driven MOPs comes at the cost of the DM's time. Given a large enough total function evaluation budget, K-RVEA/NAUTILUS Navigator can presumably find solutions of quality similar to O-NAUTILUS. Conducting so many function evaluations may be very expensive and may take a very long time. However, K-RVEA/NAUTILUS Navigator only requires the attention of the DM during the navigation phase, which is a very quick process. O-NAUTILUS on the other hand requires much fewer function evaluations, and thus less time overall, but the DM is required to wait for time-consuming function evaluations to see the updated optimistic front and resume interaction with the algorithm. Thus, between the two sets of algorithms, there is an inherent trade-off involving the number of function evaluations and the DM's wait time.

O-NAUTILUS method provides an efficient way to tackle data-driven MOPs with costly objective functions. Note that while we provide the details about the algorithms used in our implementation, an analyst can swap these with alternatives which perform a similar purpose while still maintaining the general structure of the algorithm. These choices can include, for example, using random forest regression for surrogate modelling, NSGA-III for obtaining the optimistic front, and differential evolution [92] for optimizing the expected value of the ASF.

7 THE DESDEO FRAMEWORK: AN OPEN-SOURCE COLLECTION OF INTERACTIVE MULTIOBJECTIVE OPTIMIZATION TOOLS

All good things come in open-source packages.

Source unknown

As we established in the previous chapters, interactive multiobjective optimization methods can support a DM to tackle MOPs efficiently. The IOPIS algorithm presented in [PII] enables a DM to control the scope of optimization by putting the reference point closer or farther from the Pareto front. IOPIS can also reduce the number of dimensions of the MOP with the creation of the PIS, which helps reduce the number of function evaluations. O-NAUTILUS goes a step further and significantly reduces the number of (costly) function evaluations using surrogate-assisted optimization and facilitates a DM to make informed decisions about conducting function evaluations in a targeted fashion. Many other interactive methods, both evolutionary and scalarization-based, have been proposed [12, 51, 70, 73, 78, 104].

However, obtaining and applying these methods to solve MOPs can be challenging for both analysts and DMs. Many authors do not make implementations of their methods publicly available. Methods with available implementations can be in one of many popular programming languages such as C, C++, R, Julia, Matlab, Python, or Java. An analyst needs to be proficient in many programming languages to use these methods. Even methods implemented in the same programming language can require different ways of implementing the MOP to be solved. This makes experimenting with various interactive methods challenging. The methods may also not provide a user interface, making it challenging for a DM to use them. Many open-source optimization frameworks exist that bundle together many optimization algorithms and solve some of the earlier challenges [8–10, 22, 32, 36, 38, 40, 45, 53, 94]. In our survey of such tools in [PV], we found that none of these frameworks provided interactive methods for optimization and only provided either evolutionary or scalarization-based methods.

Therefore, there was a need for a framework that provided access to many interactive MOEAs and scalarization-based methods to enable quick experimentation by analysts as well as as a possibility to conveniently apply different methods in real applications. Such a framework should also implement these methods in a modular fashion. Many interactive methods use similar tools, and implementing them with modularity makes it trivial to reuse components for different methods. Such tools include scalarization functions, tools for collecting and interpreting DM's preferences, and the population's abstraction used in all MOEAs. The modularity also enables method developers (for example, researchers) to easily create new methods by mixing components of previously implemented methods. The framework should also have a good user interface to enable interactions with the DM. Finally, the framework should be open-source, allowing analysts, researchers, and DMs alike open access to all methods for experimentation. With DESDEO, described in [PV], we provide such a framework.

7.1 Design of the Framework

We designed the framework as a collection of open-source packages, each designed to fulfill a specific need of interactive multiobjective optimization. Some of these packages, for example, those implementing the interactive methods, form the framework's core. These packages are fully mature and are ready for use by researchers, analysts, DMs, and students. Other packages, such as packages that support users with a graphical user interface, are under active development. The framework also includes some packages which implement new methods using parts of the framework but have not been incorporated into the core packages yet. Note that while the core packages are mature, we still actively implement new methods and features into them.

We implement the packages mainly in Python and TypeScript. We provide thorough documentation on the packages, including descriptions of the packages, the methods within, their implementations, and how to use them. The documentation resources include docstrings in the code, documentation websites for individual packages, tutorials using Jupyter Notebook, as well as multiple video tutorials available via the YouTube website of the multiobjective optimization group [1]. All packages developed for the DESDEO framework are available via our website[2] and GitHub[3] pages. We now describe all the packages that compose the DESDEO framework.

`desdeo-tools`. This Python package forms the framework's base and implements essential tools and techniques used by both scalarization-based and evolutionary methods. These tools include tools for non-dominated sorting, validation of different kinds of preference information, and interconversion of differ-

[1] https://www.youtube.com/channel/UC6qFfAgD8_aa28pcBmKDXTw/
[2] https://desdeo.it.jyu.fi
[3] https://github.com/industrial-optimization-group

ent kinds of preference information [76]. This package also contains implementations of various scalarization functions [13, 74, 79, 101], as well as the GLIDE-II framework [87], which can be used to parametrically generate different scalarization functions. We also implemented a tool to connect DESDEO's methods with single-objective optimizers implemented outside the framework. Quality indicators that compare the quality of solutions for MOPs are also implemented in this package [111, 112]. Finally, we have implemented tools to enable the creation of the PIS by combining multiple scalarizing functions.

`desdeo-problem`. This Python package contains the abstractions for defining MOPs. We have implemented classes and functions to define the decision variables of an MOP individually or in batches. We have implemented different classes to handle objectives. For example, the `VectorObjective` class handles cases where a source (like a simulator) returns values of multiple objectives at once. Alternatively, the `ScalarObjective` class handles cases where each objective function can be calculated independently. While we currently do not utilize this feature in the framework, the ability to selectively calculate certain objectives is useful in MOPs with objectives of mixed complexities [4]. We have also implemented classes to handle the objectives of data-driven MOPs and support the creation of surrogate models from the popular Python package `scikit-learn` [80] and related packages. Finally, we have the `MOProblem` and `DataProblem` classes which aggregate the previously mentioned tools to implement MOPs and connect them to the interactive methods implemented in other packages.

`desdeo-mcdm`. This package, also implemented in Python, contains implementations of scalarization-based interactive methods. As of early 2022, this package implements the Pareto navigator [34], NAUTILUS [72], NAUTILUS navigator [84], E-NAUTILUS [86], NAUTILUS 2 [77], synchronous NIMBUS [76], and the reference point method [101].

`desdeo-emo`. This Python package implements various interactive and *a posteriori* MOEAs. We implement reusable components of MOEAs, such as the population, crossover, mutation, and selection operators and algorithms to create reference vectors (for decomposition-based MOEAs) as modular classes and functions. Even the basic algorithm of population evolution that some MOEAs follow (create offspring $\rightarrow$ select best individuals $\rightarrow$ repeat) are implemented base classes and can be inherited into implementations of MOEAs easily. The package contains implementations of RVEA [17], NSGA-III [24], MOEA/D [106], IBEA [109], PPGA [62], and tournament EA [44]. However, these algorithms can be used for interactive optimization using IOPIS. IOPIS is implemented as a unique problem class rather than an MOEA, thus enabling its usage with all algorithms in `desdeo-emo`. Additionally, some interactive versions of previously mentioned MOEAs have also been implemented [46]. These support providing preferences interactively in the form of a reference point, as bounds on the objective values, choosing one or more preferred solutions during evolutions, or choosing solutions that a DM does not like. The package also supports changing the way preferences are provided during the solution process. It also supports

changing the MOEA during the solution process without losing any progress.

`desdeo`. This is a user-facing package. Anyone (DMs, analysts, or students) who wants to use the implemented methods only needs to install this package using the command `'pip install desdeo'`. It automatically installs all other packages. This package does not implement any additional methods, but it contains the documentation of the entire framework.

`desdeo-components`, `desdeo-frontend`, and `desdeo-webapi`. These packages are not a core part of the framework yet. We implemented the first two packages in TypeScript. The `desdeo-components` package implements interactive visual components such as visualizations and forms using which a DM can see the results of interactive optimization and provide their preferences. The `desdeo-frontend` package combines these components to form graphical user interfaces for the various methods in the DESDEO framework. The `desdeo-webapi` package, implemented in Python, connects the graphical user interface provided by the other packages to the methods provided by the core packages of DESDEO.

Other packages. We have used the DESDEO framework to develop various new methods which have not been fully incorporated into the core packages yet. The `O-NAUTILUS` package [4] implements the O-NAUTILUS algorithm. It will become a part of the core packages when `desdeo-frontend` reimplements its graphical user interface. The selector (from the SMTS algorithm) depends on the R language as well. Thus we provide it as an optional install rather than a core component. Other packages include `desdeo-vis`, `desdeo-dash`, and `desdeo-adm`, as well as some repositories containing tutorials [5].

7.2 Discussion about the DESDEO Framework

We describe various ways of using the DESDEO framework in [PV]. We showcase different use cases, such as an MOP with objectives with mathematical functions and a data-driven MOP requiring the use of surrogate models. We showcase the process of using different evolutionary and scalarization-based methods and the process of combining or switching between those methods. The framework has enabled quick creation and experimentation with the IOPIS and O-NAUTILUS algorithms. Because of this, we could iterate and improve those algorithms easily, facilitating us to present those algorithms in [PII] and [PIV] must faster than it would have been possible otherwise.

The DESDEO framework has been and continues to be a massive collaboration with tens of contributors. The needs of methods developed by the author, IOPIS and O-NAUTILUS, and the needs of other researchers have shaped the design of the framework [1–3, 67]. Many courses have utilized the framework to teach topics related to interactive multiobjective optimization. The framework

[4] https://github.com/industrial-optimization-group/O-NAUTILUS

[5] Available from https://github.com/industrial-optimization-group

fills an otherwise empty niche and provides its users open access to interactive methods, enabling new research. Furthermore, its accessibility enables those unfamiliar with interactive methods to experiment with them, opening new possibilities.

8 SOLVING A REAL-LIFE DATA-DRIVEN MULTIOBJECTIVE OPTIMIZATION PROBLEM

Journey Before Destination.

Brandon Sanderson, The Way of Kings

Through Chapters 3-7, we introduced many challenges related to interactive multiobjective optimization. We designed new techniques to resolve those challenges successfully. However, real-life optimization problems can present unexpected issues that are not solved trivially. Such issues usually are specific to the MOP to be solved and require the combined expertise of the analyst and the DM to resolve. For example, the data available for a data-driven optimization problem may not elicit a straightforward MOP formulation with clear-cut decisions about which parts of the data form the decision variables and which parts form the objectives. Therefore, formulating an MOP can require in-depth deliberation and the combined efforts of a DM (who can identify domain-specific objectives directly or indirectly obtainable from the data), other domain experts, and an analyst (who can formulate a meaningful and solvable MOP based on the suggestions of the DM and available data). Such a process may happen over multiple discussions, each discussion introducing minor improvements to the formulation. The journey made by DMs and analysts to formulate and solve an MOP, the challenges they face during the process, and the choices they employ to overcome the challenges, therefore, can be as interesting as the MOP's solutions. From the undoubtedly biased view of an analyst (which the author is in the study described in this chapter), the process followed by the DM and analysts can even be more interesting.

In [PVI], we document such a process. We considered the problem of optimization of multiple metallurgical properties of microalloyed steels. These properties depend upon, among other factors, the composition of the alloy, i.e., the concentrations of the alloying elements present in the steel. Two domain experts were the DMs for the problem, aided by a single analyst. The problem is an offline data-driven MOP, i.e., we do not have access to new function evaluations.

We solved the problem using surrogate-assisted optimization, i.e., we trained surrogate models using the data to use as the objectives of the MOP. We developed a new interactive MOEA to support two DMs simultaneously. We faced challenges in all steps, from processing the data, formulating the MOP, choosing the surrogate models, and conducting interactive optimization with multiple DMs who did not have experience with interactive methods. In solving those challenges, we utilized all the techniques we introduced in Chapters 3-7 (except for O-NAUTILUS, which we designed for online data-driven problems) and more. In [PVI], we give a detailed account of the tools and techniques we used, including those that were unsuccessful in helping us find solutions. Someone wanting to formulate and solve an MOP using real-data may not face all of the issues discussed in [PVI]. However, the techniques and the knowledge of the process we followed to solve our MOP can still make valuable additions to the toolbox of such a person.

8.1 Overview

In [PVI], we provide a "linear narrative" of the solution process. By this, we mean that we provide a complete account of the first step of formulating and solving the MOP (data preprocessing) first. We provide complete accounts of the subsequent steps (surrogate modelling, MOP formulation, and interactive multiobjective optimization) in sequential order. This presentation allows the reader to replicate our results by following the order presented in the article. In reality, however, the solution process involved significant backtracking and reformulation of the MOP based on intermediate results and discussion with the DMs.

In this chapter, we instead discuss the solution process as we conducted it temporally, thus creating a supplementary to the aforementioned linear narrative. The solution process had four phases, demarked by four significant discussions that the author (as an analyst) had with the DMs. In the following four sections, we will describe the discussions conducted over those four meetings, the work we did to prepare for them, and the outcomes of the meetings. We discuss the results of the experiments we conducted only briefly and, instead, refer to [PVI] for a more detailed account.

8.2 First meeting with the DMs

The author initiated the study by contacting the two supervisors of his master's study, who are domain experts in metallurgy and materials engineering. The author aimed to solve a real-life MOP from his previous field of study (metallurgy and materials engineering) by utilizing the knowledge gained and the techniques he developed during his doctoral studies. The former supervisors were intrigued

by interactive multiobjective optimization and agreed to provide a dataset to formulate and solve a data-driven MOP. They took up the role of DMs for this study, while the author took up the analyst's role (with the support of his current supervisors).

During the first (virtual, as this study took place during the coronavirus pandemic) meeting, the DMs provided the dataset to the analyst and discussed its contents. This original "raw" dataset consisted of 736 rows and 51 columns. Each row in the dataset represented a particular steel composition and the values for some of its measured metallurgical properties. The first twenty columns denoted the concentrations of various alloying elements such as carbon, vanadium, and titanium. The concentrations of these elements controlled the values of the metallurgical properties denoted in the rest of the rows. These properties included the ultimate tensile strength (UTS), yield strength (YS), percentage elongation (ELON), and multiple columns for Charpy energy measured at various temperatures, among many others. We provided a brief description of these properties in Chapter 2.

The analyst noted that none of the rows contained values of all of the metallurgical properties (prospective objectives). Moreover, the rows were also missing values for some alloying element concentrations (prospective decision variables). The DMs explained that they created the dataset by collating data points (rows in the raw dataset) from various sources. Some of these sources contributed as few as a single point. These sources experimented with steels with slightly different ranges of alloy composition, and almost no source included all of the alloying elements observed in the combined raw dataset. Similarly, these sources did not conduct all the experiments necessary to measure all metallurgical properties present in the raw dataset.

The YS, UTS, and ELON metallurgical properties were the only ones that had a significant representation in the dataset. Therefore, including them in the MOP formulation was an obvious choice. The DMs expressed interest in including Charpy energy in the formulation as well. However, no single Charpy energy column had enough data to train a good surrogate model (which the analyst planned to use). The analyst noted that the 16 columns of Charpy energy could be combined into a single column by treating the temperature at which the Charpy energy was measured as an additional decision variable. Then, the temperature could be kept constant at a key temperature for the MOP formulation.

Choosing this key temperature led to the general discussion about interactive decision making. The DMs, after deliberation among themselves, proposed to use different standards and grades of steel (discussed in Chapter 2) as inspiration for their preferences. The DMs would be satisfied if the interactive optimization process could utilize the data to find alloy compositions that could satisfy strict steel standards. Based on the strict steel standards, we set the key temperature for the Charpy energy objective to $-80^{\circ}C$. Therefore, we formed the first version of the data-driven MOP with YS, UTS, ELON, and Charpy energy as the four objectives.

8.3 Second meeting with the DMs

To prepare for the second meeting, the analyst preprocessed the raw dataset to make it usable for the later steps. The preprocessing included dealing with empty cells in the dataset. Some cells had non-numeric contents even though the corresponding column was numeric. We describe all steps taken by the analyst to preprocess the data in [PVI]. The analyst also separated the raw dataset into four datasets, each representing an objective. Dividing the raw dataset into four clean datasets (which had no missing or incompatible values) made it easier for the analyst to conduct many tests with them.

First, the analyst used the selector from the SMTS algorithm [PI] on the four datasets to predict the best surrogate modelling technique for each of them. The selector predicted that the extra trees regression (ExTR in [PI]) was the best for two of the datasets, and the gradient boosted regression (GBR in [PI]) was the best for the rest. As the datasets were small (20 decision variables and less than 1000 rows), training surrogate models was not a time consuming task. Therefore, the analyst decided to test many surrogate models using cross-validation testing to confirm the results of the SMTS algorithm. The analyst also decided to include two state-of-the-art surrogate modelling techniques (XGBoost [1] and LightGBM[2]) that were similar to the extra trees regression technique predicted by the SMTS algorithm. Based on the results of this test (detailed further in [PVI]), the analyst chose to use extra trees regression for the YS and ELON objectives and XGBoost for the UTS objective.

The analyst removed the Charpy energy objective from the MOP as no surrogate modelling technique performed well on the corresponding dataset. To make the surrogate model perform better, the analyst tried to incorporate knowledge from other metallurgical literature into the surrogate models to especially control for the relationship between testing temperature and Charpy energy, as mentioned in Chapter 2. However, this did not make the model perform any better. There was simply too much noise in the experiments related to the Charpy energy compared to the other chosen objectives. Thus, the analyst removed the objective. Finally, the analyst computed the effect of each of the decision variables on the objectives as predicted by the chosen surrogate models.

During the second (virtual) meeting with the DMs, the analyst presented the results of the surrogate modelling. The DMs were happy with the results (except for the Charpy energy surrogate model). The primary point of discussion for this meeting was the computed effect of the decision variables (alloy composition). The DMs compared the computed effect against metallurgical literature. Most of the computed effects matched the DMs' expectations. They approved the surrogate models, and we decided to start the next phase of the study: optimization.

---

1 Implemented in the `xgboost` Python package.

2 Implemented in the `lightgbm` Python package.

8.4 Third meeting with the DMs

The analyst formulated the MOP using DESDEO [PV]. However, during the problem formulation, he noticed that the three datasets for the three objectives contained different ranges of values for the decision variables. Therefore, an MOP that combines the three datasets (using surrogate models) needs to be constrained within the region of overlap within the three datasets. However, the region of overlap was tiny, and the initial optimization results were not very promising.

Based on the discussion of the second meeting, the analyst concluded that not all decision variables were significant for all objectives. The analyst reasoned that removing unnecessary decision variables from some of the objectives could increase the region of overlap for the remaining objectives. The analyst confirmed the validity of this approach with the DMs over email. After receiving the DMs' approval, the analyst conducted a large number of varied tests to identify the importance of individual decision variables on each of the objectives. The analyst did not use computed effects discussed in the previous meeting for this task because they resulted from a single surrogate model (for each objective).

Based on the results of the new tests (described in detail in [PVI]), the analyst chose a subset of decision variables for each objective. As a result, 17 (out of the initial 20) decision variables (and three objectives) were considered in the latest version of MOP. Individual surrogate models considered a subset of the 17 decision variables and ignored the values of the other decision variables. This successfully expanded the region of overlap and led to much better results. The DMs approved the choice made by the analyst. The analyst conducted the previous tests again with this new problem formulation.

With the help of the analyst's current supervisors, the analyst chose a set of MOEAs to use for the project. The analyst used RVEA and NSGA-III to precompute an approximate representation of the Pareto front. The precomputed front would also allow the use of NAUTILUS Navigator [84]. The analyst also chose to use interactive RVEA [46] and IOPIS for the interactive optimization process. As the DMs were unfamiliar with interactive methods, the analyst provided short text and video introduction to those methods via email.

The third (virtual) meeting with the DMs began with the discussion of the precomputed approximation of the Pareto front of the MOP with three objectives. The analyst visualized the results for the DMs as an interactive 3D scatter plot. The DMs found the shape of the Pareto front familiar [3]. The analyst then showcased NAUTILUS Navigator and interactive RVEA to the DMs using example MOPs to familiarize them with the methods.

The DMs decided to try using interactive RVEA to solve the current version of the MOP. However, they found it challenging to provide preferences without including the Charpy energy objective. To have a meaningful first interaction ses-

[3] Biobjective optimization of UTS and ELON of steels is a well-studied phenomenon. The Pareto front for such problems has a characteristic shape which has earned the name "banana curve".

sion, the analyst decided to add a simple fourth objective to the problem: the cost of the materials used to form the alloy. The analyst formulated this objective as a linear combination of decision variable values weighted using the costs of individual elements. The fourth objective introduced trade-offs to the problem that the DMs found very interesting. Because they saw how quickly the analyst could implement a new objective using DESDEO, they asked the analyst to implement yet another objective: the carbon equivalent. The carbon equivalent predicts how the hardness of steel is affected by welding and can also be calculated as a linear combination of the alloying element concentrations.

The analyst implemented the new objective solved this new MOP by repeating the previous steps. The analyst then displayed the results of interactive optimization using a dynamic parallel coordinate plot. The DMs again concluded that the trade-offs were very interesting. Many steel grades also use carbon equivalent as one of the metrics that should be met (or, more specifically, not exceeded). While this enabled them to give preferences, the solutions found by the interactive RVEA still could not satisfy some of the stricter grades because of the lack of Charpy energy objective in the MOP. The DMs concluded that the Charpy energy surrogate model should be included in the MOP, even though it did not perform as well as the other surrogate models.

8.5 Fourth meeting with the DMs

The analyst noted that the two DMs often provided very different preferences in the third meeting. The solutions found by interactive RVEA often did not satisfy both DMs simultaneously. During the preceding weeks, the author was also working on extending the IOPIS algorithm. With inputs from his current supervisors, he had implemented a version of IOPIS that used a function from NIMBUS as one of the scalarization functions to form the PIS. This version of IOPIS (NIMBUS/IOPIS) could accept preferences in the form of a classification of objectives. The author also implemented a version of IOPIS that used the same scalarization function multiple times, but with different preferences, to form the PIS. In this case, each scalarization function could accept preferences from a different DM. The thus formed MultiDM/IOPIS algorithm returned solutions that satisfied the preference of all DMs involved and solutions that represented compromises between the desires of individual DMs. The IOPIS algorithm, which was originally created for a single DM was thus extended for group decision making. The author, as the analyst, decided to use these two new interactive MOEAs for the fourth meeting with the DMs.

The analyst implemented the Charpy energy objective into the MOP formulation to prepare for the fourth meeting. The analyst used GBR surrogate models for the Charpy energy objective as it was predicted by the SMTS algorithm and performed the best in the cross-validation testing. This (final) version of the MOP thus contained six objectives: YS, UTS, ELON, Charpy energy, carbon equivalent,

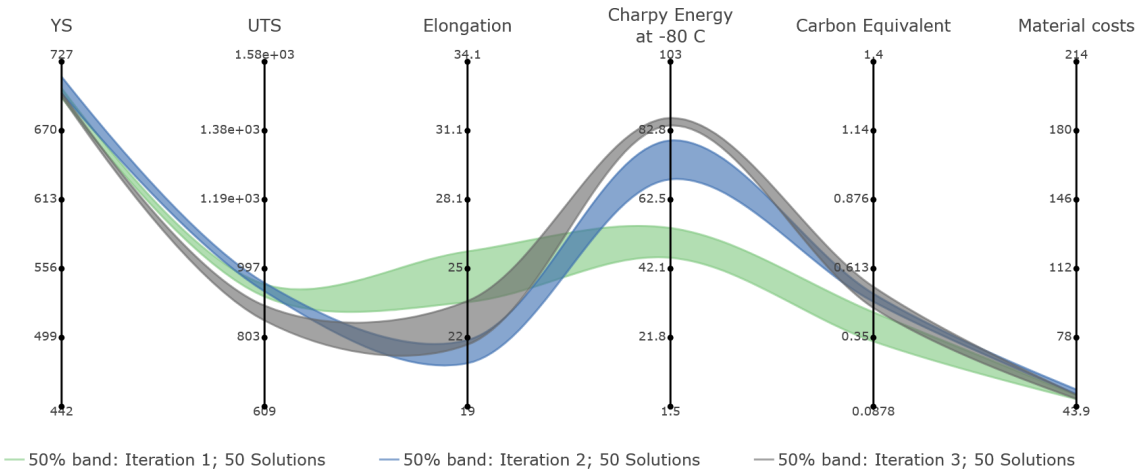


FIGURE 12 Solutions found by the DMs in three iterations with MultiDM/IOPIS visualized using SCORE bands.

and cost of materials. The analyst conducted all of the previously mentioned experiments again. The decision to break down the dataset into four datasets was very useful in making it easier to automate the process of repeating the experiments.

The fourth (final, and virtual) meeting began with the analyst presenting an approximate representation of the Pareto front to the DMs. This gave the DMs enough information about the feasibility of solutions to provide reasonable preferences. The DMs preferred interacting using the MultiDM/IOPIS method. Therefore, we discuss the details of MultiDM/IOPIS in [PVI]. The two DMs provided their preferences independently as reference points (one per DM) each iteration. The two DMs interacted with the method for three iterations. and found much better solutions than any solution in the representative Pareto front calculated using *a posteriori* RVEA and NSGA-III. We show the evolution of the solutions returned by MultiDM/IOPIS over the three interactions with the two DMs in Figure 12 as a SCORE bands plot (one band per iteration). We provide a detailed discussion of the preferences given by the DMs and the solutions obtained in [PVI].

8.6 Discussion

In this study, we successfully tackled the varied and unforeseen challenges of solving a real-life data-driven MOPs. We preprocessed a dataset to make it helpful in formulating an MOP. We determined and validated the best surrogate modelling techniques to model the potential objectives of the MOP. We iterated over the MOP formulation multiple times to arrive at the version presented in [PVI], requiring us to add or remove objectives based on the needs of the DMs and the limitations of the starting dataset. We designed a new interactive MOEA to enable two DMs to give their (different) preferences simultaneously. We also faced

and dealt with many minor challenges in the process.

In doing so, we utilized most of the methods previously developed by the author (and described in previous chapters), as well as well-known data analysis tools. We introduced the concept of interactive multiobjective optimization to two DMs, enabling them to notice novel trade-offs. It helped the analyst make a better MOP formulation for the DMs. Our proposed method, MultiDM/IOPIS extends the previously discussed IOPIS algorithm. We found better solutions than those obtained by MOP formulations with fewer objectives or *a posteriori* methods. Ultimately, we provided a detailed account of the tools and techniques we used and the protocols we followed to solve our MOP. We thus created a general guideline to solve data-driven MOPs interactively.

9 CONCLUSIONS AND AUTHOR'S CONTRIBUTIONS

How do you want to do this?

Matt Mercer, Critical Role

The field of interactive multiobjective optimization is a vast one. Interactive methods have the capacity to solve problems from innumerable domains. This capacity, however, also brings with it multifaceted challenges. Some of these challenges span the entire breadth of the field of interactive multiobjective optimization; others may only show up in specific application domains or even with individual DMs. The way we overcome these challenges (or fail to do so) can have monumental impacts on the decisions made by DMs and the solutions implemented in real life.

This thesis tackles several such challenges, spanning the entire process of interactive multiobjective optimization, from problem formulation to decision making. We do not claim to address all major challenges one may face while solving MOPs, but all challenges we do address are significant ones. We created tools and techniques to support DMs and analysts achieve good results in various demanding situations. Moreover, through this thesis and its included articles, we have provided a general framework to approach such challenges, whether expected or unforeseen.

With the final chapter of this thesis, we provide our concluding remarks. In Section 9.1, we draw conclusions for the methods developed and experiments conducted throughout this thesis. We also discuss avenues of future research in the section. We follow that by elaborating the author's contributions to the included articles in Section 9.2. We close this chapter by providing some final thoughts in Section 9.3.

9.1 Conclusions and Future Research

We began this thesis by addressing the challenge of choosing the best surrogate modelling techniques for data-driven MOPs to fit surrogate models to data available so that optimization methods can call these models. We trained a selector to do this task using the proposed SMTS algorithm. We trained this selector on thousands of datasets, with the ability to choose between ten different surrogate modelling techniques. With this selector, an analyst can get good candidate surrogate models for their MOPs. Alternatively, they can use the predictions to narrow down the list of surrogate modelling techniques they consider.

We then introduced a whole new paradigm in interactive multiobjective evolutionary optimization with the creation of the IOPIS algorithm. The IOPIS algorithm makes the creation of interactive MOEAs a trivial task by using the concept of the PIS. It offers a middle ground between MOEAs that work in the same number of dimensions as the number of objectives of an MOP and scalarization-based methods, which collapse the number of dimensions down to one. IOPIS allows analysts to create interactive MOEAs tailored to suit the needs of the DM (or DMs) by letting analysts choose the scalarization functions that form the PIS. Thus, DMs can provide their preference in ways that they are comfortable.

We then tackled the problem of presenting the results of optimization to DMs in a meaningful manner. To overcome this challenge, we created an evolution of the parallel coordinates plot, the SCORE bands plot. The SCORE bands visualization highlights the patterns in the solutions and the objectives simultaneously by using clustering algorithms and information about the correlation between objectives. We encode this information with the visualization in an appealing and visually striking fashion. The SCORE bands visualization enables DMs to gain insights about the MOP and the trade-offs involved quickly and help digesting even large amounts of information conveniently.

SCORE bands, however, does not address the issue of visualizing the uncertainty of prediction of surrogate models used in data-driven MOPs. Such information can be essential to a DM, especially if the problem is online and has costly objective functions. To support DMs in such cases, we designed the O-NAUTILUS algorithm. The proposed algorithm enables a DM to make informed decisions by visualizing known and optimistic fronts. The algorithm also allows a DM to target the costly function evaluations in their region of interest. The resulting algorithm, therefore, is very efficient and effective.

The new algorithms we proposed would only be helpful to others if they are accessible in the first place. DMs, analysts, researchers, and students sometimes face an additional challenge when experimenting with interactive methods. Authors of interactive methods published in the literature often do not make their implementations available. We solve this problem by creating the open-source Python framework for interactive multiobjective optimization: DESDEO. The DESDEO framework implements many popular interactive methods in a modular and easy-to-use manner. DMs and analysts can download the frame-

work and solve their MOPs with it. Researchers can experiment with many methods using a single framework and create new methods using the already implemented modular components in the framework. The framework has also been used to teach interactive multiobjective optimization to students. We have implemented our previously mentioned algorithms using the DESDEO framework and made the source code of all our methods publicly available.

We created SMTS, IOPIS, SCORE bands, O-NAUTILUS, and DESDEO to deal with known, specific challenges. However, real-life MOPs often raise unexpected challenges. To showcase the effectiveness of our methods and techniques, we used them to solve a real-life data-driven problem from the domain of metallurgy and materials engineering. In that process, we faced various major and minor challenges. We processed a raw, noisy dataset to make it suitable for use in surrogate-assisted optimization. We chose and validated the best surrogate modelling techniques for the potential objectives of the MOP. We interacted with the two DMs to formulate a functional MOP over multiple sessions. Interactive MOEAs were crucial in that process. We implemented a whole new method, MultiDM/IOPIS, to enable the two DMs to provide their preferences simultaneously and receive satisfactory solutions for both even if their preferences are different. Using interactive MOEAs to solve an MOP with six objectives, we discovered new trade-offs and better solutions.

The new techniques we proposed do much more than just solve the corresponding challenges. They open pathways to exciting new research directions. The SMTS algorithm can be extended (given enough computational resources) to predict not just the best surrogate modelling technique for a dataset but also the best MOEAs. We have only scratched the surface with the IOPIS algorithm. Studying the landscape in the PIS could lead to exciting discoveries. We limited the construction of the PIS to use achievement scalarizing functions only. Relaxing this condition opens up the variety of scalarizing functions that can be used, possibly leading to the creation of new methods with valuable properties.

Both SCORE bands and O-NAUTILUS can benefit from extensive case studies involving many real DMs. The input from the DMs will help us improve those algorithms to their total capacity. The DESDEO framework will be actively developed for a long time. In the short term, we plan to finish the initial development of its graphical user interface and add many test problems and popular interactive methods to the framework. In the long term, we plan to support additional optimization domains within the framework, such as scenario-based multiobjective optimization and optimization under deep uncertainty. The DESDEO team has even made progress on creating physical user interfaces [1] to make the DM's interaction experience more "tactile" [68].

[1] https://github.com/phoopies/DesdeoInterface

9.2 Author's Contributions

The choice of research topics considered in this thesis was influenced significantly by the author's membership in the Multiobjective Optimization Group (at the University of Jyväskylä) and the DESDEO development team. The author's supervisors had a significant role in setting the overall direction of this thesis toward interactive multiobjective optimization methods.

The idea of developing the surrogate modelling technique selector in [PI] originated with Dr. Manuel López-Ibáñez's (then, University of Manchester, UK) research visit to JYU in 2018. His summer school course on "Data analytics + Machine learning + Optimisation" introduced the idea of automatic algorithm selection to the author. We had further discussions on the topic during his collaboration with the Multiobjective Optimization Group. Once the author formulated the basic idea of the selector with the support of Dr. Manuel López-Ibáñez and Prof. Kaisa Miettinen, the author implemented the concept in MATLAB. The initial version of the selector was trained on a small number of features and datasets and did not perform very well. Dr. Manuel López-Ibáñez suggested the idea of using exploratory landscape analysis features. The author reimplemented the idea in Python, incorporated Dr. López-Ibáñez's idea, and created a workflow to automatically generate a large amount of data from various sources. The author then trained and tested the selector on the numerous datasets, leading to better results, as reported in [PI]. The author conducted all numerical experiments, compiled the results, and wrote most of the article [PI], which Prof. Manuel López-Ibáñez augmented and Prof. Kaisa Miettinen spearheaded the revision process.

The roots of the new paradigm in interactive multiobjective optimization (introduced in [PII]) have their origin in Dr. Tinkle Chugh's lecture to the Multiobjective Optimization Group on the topic "A study on using different scalarizing functions in Bayesian multiobjective optimization" and the ensuing discussions with the group. The author then studied the properties of pairs (and later, bigger groups) of scalarization functions. Prof. Kaisa Miettinen and Dr. Jussi Hakanen informed the author about the properties of a subset of scalarization functions: the achievement scalarizing functions. The author used these properties to mathematically formulate the desirable properties of the now-named Preference Incorporated Space. Convinced by these new properties, the author created and implemented the IOPIS algorithm within the DESDEO framework. The author compared the IOPIS algorithm with popular interactive and *a posteriori* evolutionary algorithms on hundreds of benchmark test cases. The author also designed a novel way to visualize the results to enable straightforward interpretation. The author wrote the initial draft of [PII], and Prof. Kaisa Miettinen and Dr. Jussi Hakanen were heavily involved in the revision process.

The visualization ideas that eventually evolved into SCORE bands in [PIII] were first developed by Prof. Kerstin Dächert, Prof. Kathrin Klamroth, Prof. Kaisa Miettinen, and Prof. Ralph E. Steuer at the Dagstuhl Seminar 20031 (Scalability in Multiobjective Optimization). The author was initially invited to the project

to implement the previously proposed ideas. The visualization did not perform as well as intended, which led the author to propose significant additions to the visualization. These include treating the problem of ordering the objectives as a travelling salesperson problem, developing new metrics to calculate the placement and distance between objectives, and representing clusters of solutions as bands. The author implemented these ideas in an easy to use interactive graphical user interface that allowed quick prototyping and testing of the effect of the different choices available in the algorithm (the choice of the distance metric, for example). The authors of [PIII] collectively selected the most optimal combination of choices to be presented in the paper. Prof. Ralph E. Steuer provided the artificial datasets for the case studies presented in [PIII]. The author generated the benchmark datasets, and Dr. Atanu Maxumdar provided the GAA dataset as an example of data from a real MOP. The author created all the results and figures, wrote the initial draft of Sections 3, 4 and 5 of [PIII], and was actively involved in revising the paper.

Numerous discussions with Dr. Michael Emmerich during his research visit to the Multiobjective Optimization Group led to the development of the O-NAUTILUS algorithm proposed in [PIV]. The development of the algorithm was a highly collaborative process with equal contributions from all co-authors. The major contributions of the author during the algorithm development were the details of evaluating the "Optimistic front" and visualization of the known and optimistic reachable ranges. The author implemented most of the O-NAUTILUS algorithm and the associated graphical user interface. The exception was the implementation of the algorithm to conduct targeted function evaluations, which Dr. Atanu Mazumdar contributed. The author made various algorithmic decisions during the implementation process, such as choosing hyperparameters and the general flowchart of the optimization process. The author also updated the user interface design over multiple iterations based on the co-authors' comments. The author implemented the problem used for the case study in [PIV], which Dr. Bekir Afsar then solved as the DM. The author wrote the initial draft of Section 3 (except Subsection 3.4 which Dr. Atanu Mazumdar contributed) of [PIV]. He also coordinated the writing of the paper, contributing to all Sections of the paper along with the co-authors, and lead the revision process.

The author initially joined the University of Jyväskylä and the Multiobjective Optimization Group as a project researcher in the DESDEO project. He was tasked with implementing evolutionary algorithms within the framework. The author found the original version of DESDEO challenging and unwieldy to augment and further develop. Therefore, after considerable discussion with his supervisors, he decided to implement the evolutionary algorithms in a new, unconnected Python package named pyRVEA. The DESDEO development team then concluded that DESDEO needed to be redesigned with modularity and extensibility as core principles. The abstractions introduced by the author in pyRVEA, the Problem class, for example, became the foundations of the new DESDEO framework introduced in [PV]. The pyRVEA package evolved into the desdeo-emo package as the author (and other contributors) implemented more interac-

tive and *a posteriori* evolutionary algorithms. The author also oversaw and guided contributors to the DESDEO framework. The author also has various contributions to the other packages of the DESDEO framework, including desdeo-tools (implementation of the GLIDE-II framework and fast non-domination sorting tools), desdeo-problem (various problem classes, test problems, and case studies), desdeo-frontend (graphical user interfaces for interactive evolutionary algorithms), and desdeo-webapi (the backend for interactive evolutionary algorithms). The author is thus the lead developer of desdeo-emo and one of the main developers of various other DESDEO packages alongside Giovanni Misitano. The author wrote Sections 3 and 4 of [PV] with Giovanni Misitano. The author focused especially on the parts of the paper that involved evolutionary algorithms and was involved in the revision process of the entire paper. The author also implemented and ran the case studies involving evolutionary algorithms discussed in Section 4.

Prof. Nirupam Chakraborti and Prof. Debalay Chakrabarti were the author's supervisors during his master's study in the Department of Metallurgy and Materials Engineering at the Indian Institute of Technology Kharagpur. Prof. Nirupam Chakraborti introduced the concept of evolutionary algorithms to the author, and Prof. Debalay Chakrabarti provided a metallurgical data-driven problem and expert guidance for the author's master's thesis. Therefore, collaborating with them to solve a challenging data-driven problem using DESDEO was a natural choice for the author. They provided the data for the MOP presented in [PV] and acted as DMs in the case study, though they were previously unfamiliar with interactive optimization methods. The author took up the analyst's role and conducted the various tests discussed in [PV] to maximize the utility of the provided data. The author's current supervisors guided him throughout the process of solving the MOP. They also counselled the author to ensure that interaction sessions with the DMs were meaningful and informative. With their support, the author (as an analyst) conducted multiple meetings with the DMs to:

1. Formulate a meaningful MOP
2. Design an interactive optimization method that would make it easier for them to provide their preferences
3. Conduct interactive optimization with the newly developed method

Thus, the author was responsible for the items above. The DMs provided domain expertise while judging the validity of the surrogate models and interpreting the results. The author wrote Sections 3-8 of [PV], with contributions from Prof. Debalay Chakrabarti in Section 5 to confirm the validity of trained surrogate models by comparing the predictions of the models to metallurgical literature. The author was also involved in writing Sections 1, 2, and 9, and the revision of the entire paper.

9.3 Final Thoughts

Collaborating with the Multiobjective Optimization Group and its visitors has been crucial in the academic growth of the author and the development of this thesis. The core idea of the thesis, discussing and solving the challenges of interactive multiobjective optimization, came about as a direct result of collaborating with so many different co-authors and team members.

As mentioned in Chapter 1, the aim of this thesis ultimately is to discuss the possibilities provided by interactive methods. The methods we presented in this thesis helped DMs solve MOPs efficiently and effectively and gave them more control over the interactive process. We pioneered entirely new methodologies to tackle the challenges we faced. We hope that this thesis will provide an excellent guide to anyone who wishes to venture into the rewarding field of interactive multiobjective optimization.

YHTEENVETO (SUMMARY IN FINNISH)

Monitavoiteoptimoinnin ongelmia, toisin sanoen ongelmia, joissa on useita ristiriitaisia tavoitefunktoita, voidaan ratkaista interaktiivisten menetelmien avulla. Niissä ns. päätöksentekijä ohjailee optimointiprosessia ja voi tehokkaasti löytää mieleisensä ratkaisun. Tähän liittyy kuitenkin erilaisia haasteita, kuten asianmukainen ongelmaan liittyvän tiedon käsittely, sopivan interaktiivisen menetelmän löytäminen ja interaktiivisen menetelmän käytön helpottaminen ymmärrettävien visualisointikeinojen avulla.

Tämän väitöskirjan ensisijainen tavoite on tunnistaa ja ratkoa interaktiivisiin menetelmiin ja niiden käyttöön liittyviä haasteita. Työhön sisältyy kuusi artikkelia [PI-PVI], jotka kukin vastaavat interaktiivisen monitavoiteoptimoinnin eri vaiheiden haasteisiin. Yksi käsitellyistä haasteista on uusien interaktiivisten menetelmien tuominen tutkijoiden, opiskelijoiden, analyytikkojen ja päätöksentekijöiden saataville. Ratkaisu tähän on avoimen lähdekoodin DESDEO-ohjelmistokehikko (desdeo.it.jyu.fi). Se sisältää monien interaktiivisten menetelmien modulaarisia implementaatiota, mukaan lukien tässä väitöskirjassa esitellyt uudet menetelmät.

Väitöskirjassa esitellään uudet interaktiiviset menetelmät O-NAUTILUS ja IOPIS. O-NAUTILUS on suunnattu laskennallisesti kalliille ongelmille, joissa tulee laskea optimoitavien funktioiden arvoja tehokkaasti. IOPIS puolestaan esittelee täysin uuden paradigman interaktiiviseen monitavoiteoptimointiin evoluutiopohjaisilla menetelmillä. Se yhdistää ideoita skalarisointipohjaisista menetelmistä evoluutiopohjaisiin. Täten mikä tahansa evoluutiopohjainen menetelmä voidaan muuttaa interaktiiviseksi modulaarisin keinoin sisällyttämällä päätöksentekijän mieltymyksiä uuteen avaruuteen, ns. preferenssejä liittävään avaruuteen.

Työssä myös esitellään SCORE bands -visualisaatiomenetelmä, jonka avulla voidaan päätöksentekijälle havainnollistaa ymmärrettävästi ja intuitiivisesti lukuisia ratkaisuja, joissa on monia tavoitteita. Tämä tukee päätöksentekijää ymmärtämään nopeasti eri ratkaisujen kirjoa ja tavoitteiden riippuvuussuhteita, jolloin päätöksenteko on helpompaa.

Väitöskirjassa esiteltyjä menetelmiä ja muita työkaluja sovelletaan todelliseen metallurgian alan optimointiongelmaan, jossa on 6 tavoitetta ja useita päätöksentekijöitä. Ratkaisuprosessin vaativuutta lisäsi tarve ratkoa matkan varrella ilmenneitä ennalta-arvaamattomia ongelmia, mitä on kuvattu väitöskirjassa. Ongelma tuo esiin interaktiivisten menetelmien hyötyjä. Täten väitöskirja toimii oppaana sille, miten käytännön monitavoiteoptimoinnin ongelmia voidaan ratkoa interaktiivisten menetelmien avulla muillakin sovellusaloilla. Uudet menetelmät vastaavat todellisiin tarpeisiin ja siten täyttävät aukkoja kirjallisuudessa esitettyjen menetelmien kirjossa.

REFERENCES

[1] AFSAR, B., MIETTINEN, K., AND RUIZ, A. B. An artificial decision maker for comparing reference point based interactive evolutionary multiobjective optimization methods. In *Evolutionary Multi-Criterion Optimization, 11th International Conference, EMO 2021, Proceedings* (Cham, 2021), H. Ishibuchi, Q. Zhang, R. Cheng, K. Li, H. Li, H. Wang, and A. Zhou, Eds., Springer International Publishing, pp. 619–631.

[2] AFSAR, B., RUIZ, A. B., AND MIETTINEN, K. Comparing interactive evolutionary multiobjective optimization methods with an artificial decision maker. *Complex & Intelligent Systems* (2021), to appear. 10.1007/s40747-021-00586-5.

[3] AGHAEI POUR, P., RODEMANN, T., HAKANEN, J., AND MIETTINEN, K. Surrogate assisted interactive multiobjective optimization in energy system design of buildings. *Optimization and Engineering 23* (2022), 303–327.

[4] ALLMENDINGER, R., HANDL, J., AND KNOWLES, J. Multiobjective optimization: When objectives exhibit non-uniform latencies. *European Journal of Operational Research 243*, 2 (2015), 497–513.

[5] ANKERST, M., BERCHTOLD, S., AND KEIM, D. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *Proceedings of the IEEE Symposium on Information Visualization* (1998), IEEE, pp. 52–60.

[6] AUDET, C. A survey on direct search methods for blackbox optimization and their applications. In *Mathematics without Boundaries*, P. Pardalos and T. Rassias, Eds. Springer, 2014, ch. 2, pp. 31–56.

[7] BECHIKH, S., KESSENTINI, M., SAID, L. B., AND GHÉDIRA, K. Chapter four - Preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. In *Advances in Computers*, A. R. Hurson, Ed. Elsevier, 2015, pp. 141–207.

[8] BENITEZ-HIDALGO, A., NEBRO, A. J., GARCIA-NIETO, J., OREGI, I., AND DEL SER, J. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation 51* (2019), article 100598.

[9] BISCANI, F., IZZO, D., AND YAM, C. H. A global optimisation toolbox for massively parallel engineering optimisation. *arXiv:1004.3824* (2010).

[10] BLANK, J., AND DEB, K. Pymoo: Multi-objective optimization in Python. *IEEE Access 8* (2020), 89497–89509.

[11] BORG, I., AND GROENEN, P. J. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[12] BRANKE, J., DEB, K., MIETTINEN, K., AND SLOWIŃSKI, R., Eds. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer, 2008.

[13] BUCHANAN, J. T. A naïve approach for solving MCDM problems: The GUESS method. *Journal of the Operational Research Society 48*, 2 (1997), 202–206.

[14] CAJOT, S., SCHÜLER, N., PETER, M., KOCH, A., AND MARÉCHAL, F. Interactive optimization with parallel coordinates: Exploring multidimensional spaces for decision support. *Frontiers in ICT 5* (2019), article 32.

[15] CALLISTER, W. D., AND RETHWISCH, D. G. *Materials Science and Engineering: An Introduction*. Wiley New York, 2018.

[16] CHEN, T., AND GUESTRIN, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (2016), pp. 785–794.

[17] CHENG, R., JIN, Y., OLHOFER, M., AND SENDHOFF, B. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation 20*, 5 (2016), 773–791.

[18] CHUGH, T. Handling expensive multiobjective optimization problems with evolutionary algorithms. *PhD Dissertation, Jyväskylä Studies in Computing, University of Jyväskylä*, 263 (2017).

[19] CHUGH, T., JIN, Y., MIETTINEN, K., HAKANEN, J., AND SINDHYA, K. A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. *IEEE Transactions on Evolutionary Computation 22*, 1 (2016), 129–142.

[20] CHUGH, T., SINDHYA, K., HAKANEN, J., AND MIETTINEN, K. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing 23*, 9 (2019), 3137–3166.

[21] COELLO, C. A. C., LAMONT, G. B., AND VAN VELDHUIZEN, D. A. *Evolutionary algorithms for solving multi-objective problems*. Springer New York, 2007.

[22] D., H. Platypus: Multiobjective optimization in Python. https://platypus.readthedocs.io. Accessed May 31, 2022.

[23] DEB, K. *Multi-objective optimization using evolutionary algorithms*. Wiley UK, Chichester, 2001.

[24] DEB, K., AND JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation 18*, 4 (2014), 577–601.

[25] DEB, K., AND MIETTINEN, K. Nadir point estimation using evolutionary approaches: Better accuracy and computational speed through focused search. In *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems* (2010), M. Ehrgott, B. Naujoks, T. J. Stewart, and J. Wallenius, Eds., Springer, Berlin, Heidelberg, pp. 339–354.

[26] DEB, K., MIETTINEN, K., AND CHAUDHURI, S. Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. *IEEE Transactions on Evolutionary Computation 14*, 6 (2010), 821–841.

[27] DEB, K., PRATAP, A., AGARWAL, S., AND MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 2 (2002), 182–197.

[28] DEB, K., AND SAXENA, D. Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In *Proceedings of the World Congress on Computational Intelligence (WCCI-2006)* (2006), pp. 3352–3360.

[29] DEB, K., AND SUNDAR, J. Reference point based multi-objective optimization using evolutionary algorithms. In *GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (New York, 2006), ACM, pp. 635–642.

[30] DEB, K., THIELE, L., LAUMANNS, M., AND ZITZLER, E. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002)* (2002), IEEE, pp. 825–830.

[31] DUA, D., AND GRAFF, C. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2017. Accessed May 31, 2022.

[32] DURILLO, J. J., AND NEBRO, A. J. jMetal: A java framework for multi-objective optimization. *Advances in Engineering Software 42*, 10 (2011), 760–771.

[33] EMMERICH, M. Single-and multi-objective evolutionary design optimization assisted by Gaussian random field metamodels. *PhD dissertation, University of Dortmund* (2005).

[34] ESKELINEN, P., MIETTINEN, K., KLAMROTH, K., AND HAKANEN, J. Pareto navigator for interactive nonlinear multiobjective optimization. *OR Spectrum 32*, 1 (2010), 211–227.

[35] FALCÓN-CARDONA, J. G., AND COELLO, C. A. C. Indicator-based multi-objective evolutionary algorithms: A comprehensive survey. *ACM Computing Surveys 53*, 2 (2020), 1–35.

[36] FORTIN, F.-A., DE RAINVILLE, F.-M., GARDNER, M.-A., PARIZEAU, M., AND GAGNÉ, C. DEAP: Evolutionary algorithms made easy. *The Journal of Machine Learning Research 13*, 1 (2012), 2171–2175.

[37] FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics 29*, 5 (2001), 1189 – 1232.

[38] GANDIBLEUX, X., SOLEILHAC, G., PRZYBYLSKI, A., AND S., R. vOptSolver: an open source software environment for multiobjective mathematical optimization. In *IFORS2017: 21st Conference of the International Federation of Operational Research Societies* (2017).

[39] GARDNER, M. W., AND DORLING, S. Artificial neural networks (the multilayer perceptron)- A review of applications in the atmospheric sciences. *Atmospheric Environment 32*, 14-15 (1998), 2627–2636.

[40] GARRETT, A. inspyred: Bio-inspired algorithms in Python. https://github.com/aarongarrett/inspyred. Accessed May 31, 2022.

[41] GEOFFRION, A., DYER, J., AND FEINBERG, A. An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management Science 19*, 4 (1972), 357–368.

[42] GETTINGER, J., KIESLING, E., STUMMER, C., AND VETSCHERA, R. A comparison of representations for discrete multi-criteria decision problems. *Decision Support Systems 54*, 2 (2013), 976–985.

[43] GEURTS, P., ERNST, D., AND WEHENKEL, L. Extremely randomized trees. *Machine Learning 63*, 1 (2006), 3–42.

[44] GOLDBERG, D. E., AND DEB, K. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. Elsevier, 1991, pp. 69–93.

[45] HADKA, D. MOEA framework: A free and open source java framework for multiobjective optimization. http://moeaframework.org/. Accessed May 31, 2022.

[46] HAKANEN, J., CHUGH, T., SINDHYA, K., JIN, Y., AND MIETTINEN, K. Connections of reference vectors and different types of preference information in interactive multiobjective evolutionary algorithms. In *Proceeding of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (2016), pp. 1–8.

[47] HAKANEN, J., RADOŠ, S., MISITANO, G., SAINI, B. S., MIETTINEN, K., AND MATKOVIĆ, K. Interactivized: Visual interaction for better decisions with interactive multiobjective optimization. *IEEE Access 10* (2022), 33661–33678.

[48] HANSEN, N., AND OSTERMEIER, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation 9*, 2 (2001), 159–195.

[49] HARTIKAINEN, M., MIETTINEN, K., AND KLAMROTH, K. Interactive Non-convex Pareto Navigator for multiobjective optimization. *European Journal of Operational Research 275*, 1 (2019), 238–251.

[50] HUBAND, S., BARONE, L., WHILE, L., AND HINGSTON, P. A scalable multi-objective test problem toolkit. In *Evolutionary Multi-Criterion Optimization, Third International Conference, Proceedings* (Berlin. Heidelberg, 2005), C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds., Springer, pp. 280–295.

[51] HWANG, C.-L., AND MASUD, A. S. M. *Multiple Objective Decision Making-Methods and Applications: A State-of-the-Art Survey*. Springer, 1979.

[52] ISHIBUCHI, H., TSUKAMOTO, N., AND NOJIMA, Y. Evolutionary many-objective optimization: A short review. In *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)* (2008), pp. 2419–2426.

[53] IZZO, D., AND BISCANI, F. PyGMO: Python parallel global multiobjective optimizer. https://esa.github.io/pygmo. Accessed May 31, 2022.

[54] JIN, Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation 1*, 2 (2011), 61–70.

[55] JIN, Y., WANG, H., AND SUN, C. *Data-Driven Evolutionary Optimization*. Springer, 2021.

[56] KANIA, A., SIPILÄ, J., AFSAR, B., AND MIETTINEN, K. Interactive multi-objective optimization in lot sizing with safety stock and safety lead time. In *Computational Logistics, 12th International Conference, ICCL 2021, Proceedings* (Cham, 2021), M. Mes, E. Lalla-Ruiz, and S. Voß, Eds., Springer, pp. 208–221.

[57] KE, G., MENG, Q., FINLEY, T., WANG, T., CHEN, W., MA, W., YE, Q., AND LIU, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[58] KERSCHKE, P., AND TRAUTMANN, H. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary Computation 27*, 1 (2019), 99–127.

[59] KORHONEN, P., AND WALLENIUS, J. A Pareto Race. *Naval Research Logistics 35*, 6 (1988), 615–623.

[60] KORHONEN, P., AND WALLENIUS, J. Visualization in the multiple objective decision-making framework. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds. Springer, Berlin, 2008, pp. 195–212.

[61] LI, K., WANG, R., ZHANG, T., AND ISHIBUCHI, H. Evolutionary many-objective optimization: A comparative study of the state-of-the-art. *IEEE Access 6* (2018), 26194–26214.

[62] LI, X. A real-coded predator-prey genetic algorithm for multiobjective optimization. In *Evolutionary Multi-Criterion Optimization, 2nd International Conference, EMO 2003, Proceedings* (Berlin, Heidelberg, 2003), C. M. Fonseca, P. J. Fleming, E. Zitzler, L. Thiele, and K. Deb, Eds., Springer, pp. 207–221.

[63] LIAO, X., LI, Q., YANG, X., ZHANG, W., AND LI, W. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization 35*, 6 (2008), 561–569.

[64] LOTOV, A. V., AND MIETTINEN, K. Visualizing the Pareto frontier. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds. Springer, Berlin, 2008, pp. 213–243.

[65] MATHERON, G. Principles of geostatistics. *Economic Geology 58*, 8 (1963), 1246–1266.

[66] MAZUMDAR, A. Novel approaches for offline data-driven evolutionary multiobjective optimization. *PhD Dissertation, JYU dissertations, University of Jyväskylä*, 456 (2021).

[67] MAZUMDAR, A., CHUGH, T., HAKANEN, J., AND MIETTINEN, K. Probabilistic selection approaches in decomposition-based evolutionary algorithms for offline data-driven multiobjective optimization. *IEEE Transactions on Evolutionary Computation* (2022), to appear. 10.1109/TEVC.2022.3154231.

[68] MAZUMDAR, A., OTAYAGICH, S., AND MIETTINEN, K. Interactive evolutionary multiobjective optimization with modular physical user interface. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, 2022), GECCO '22, ACM, p. 1835–1843.

[69] MERSMANN, O., BISCHL, B., TRAUTMANN, H., PREUSS, M., WEIHS, C., AND RUDOLPH, G. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, 2011), GECCO '11, ACM, p. 829–836.

[70] MIETTINEN, K. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1999.

[71] MIETTINEN, K. Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum 36*, 1 (2014), 3–37.

[72] MIETTINEN, K., ESKELINEN, P., RUIZ, F., AND LUQUE, M. Nautilus method: An interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research 206*, 2 (2010), 426–434.

[73] MIETTINEN, K., HAKANEN, J., AND PODKOPAEV, D. Interactive nonlinear multiobjective optimization methods. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, S. Greco, M. Ehrgott, and J. Figueira, Eds., 2 ed. Springer, 2016, pp. 927–976.

[74] MIETTINEN, K., AND MÄKELÄ, M. Interactive bundle-based method for nondifferentiable multiobjective optimization: NIMBUS. *Optimization 34*, 3 (1995), 231–246.

[75] MIETTINEN, K., AND MÄKELÄ, M. M. On scalarizing functions in multiobjective optimization. *OR Spectrum 24*, 2 (2002), 193–213.

[76] MIETTINEN, K., AND MÄKELÄ, M. M. Synchronous approach in interactive multiobjective optimization. *European Journal of Operational Research 170*, 3 (2006), 909–922.

[77] MIETTINEN, K., PODKOPAEV, D., RUIZ, F., AND LUQUE, M. A new preference handling technique for interactive multiobjective optimization without trading-off. *Journal of Global Optimization 63*, 4 (2015), 633–652.

[78] MIETTINEN, K., RUIZ, F., AND WIERZBICKI, A. P. Introduction to multiobjective optimization: Interactive approaches. In *Multiobjective Optimization: Interative and Evolutionary Approaches*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds. Springer, 2008, pp. 27–57.

[79] NAKAYAMA, H., AND SAWARAGI, Y. Satisficing trade-off method for multiobjective programming. In *Interactive Decision Analysis* (Berlin, Heidelberg, 1984), M. Grauer and A. P. Wierzbicki, Eds., Springer, pp. 113–122.

[80] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[81] PLOTLY TECHNOLOGIES INC. Collaborative data science, 2015.

[82] RICE, J. R. The algorithm selection problem. In *Advances in Computers*, M. Rubinoff and M. C. Yovits, Eds. Elsevier, 1976, pp. 65–118.

[83] RUIZ, A. B., LUQUE, M., MIETTINEN, K., AND SABORIDO, R. An interactive evolutionary multiobjective optimization method: Interactive WASF-GA. In *Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015, Proceedings* (Cham, 2015), A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello, Eds., Springer International Publishing, pp. 249–263.

[84] RUIZ, A. B., RUIZ, F., MIETTINEN, K., DELGADO-ANTEQUERA, L., AND OJALEHTO, V. NAUTILUS Navigator: free search interactive multiobjective optimization without trading-off. *Journal of Global Optimization 74*, 2 (2019), 213–231.

[85] RUIZ, A. B., SABORIDO, R., AND LUQUE, M. A preference-based evolutionary algorithm for multiobjective optimization: the weighting achievement scalarizing function genetic algorithm. *Journal of Global Optimization 62*, 1 (2015), 101–129.

[86] RUIZ, A. B., SINDHYA, K., MIETTINEN, K., RUIZ, F., AND LUQUE, M. E-NAUTILUS: a decision support system for complex multiobjective optimization problems based on the NAUTILUS method. *European Journal of Operational Research 246*, 1 (2015), 218–231.

[87] RUIZ, F., LUQUE, M., AND MIETTINEN, K. Improving the computational efficiency in a global formulation (GLIDE) for interactive multiobjective optimization. *Annals of Operations Research 197*, 1 (2012), 47–70.

[88] SAWARAGI, Y., NAKAYAMA, H., AND TANINO, T. *Theory of Multiobjective Optimization*. Elsevier, 1985.

[89] SHAMMAMAH HOSSAIN. Visualization of bioinformatics data with Dash Bio. In *Proceedings of the 18th Python in Science Conference* (2019), Chris Calloway, David Lippa, Dillon Niederhut, and David Shupe, Eds., pp. 126–133.

[90] SMITH-MILES, K. A. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys 41*, 1 (2009), 1–25.

[91] STEINWART, I., AND CHRISTMANN, A. *Support vector machines*. Springer, 2008.

[92] STORN, R., AND PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization 11*, 4 (1997), 341–359.

[93] THIELE, L., MIETTINEN, K., KORHONEN, P. J., AND MOLINA, J. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation 17*, 3 (2009), 411–436.

[94] TIAN, Y., CHENG, R., ZHANG, X., AND JIN, Y. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine 12*, 4 (2017), 73–87.

[95] TRINKAUS, H. L., AND HANNE, T. knowCube: A visual and interactive support for multicriteria decision making. *Computers and Operations Research 32*, 5 (2005), 1289–1309.

[96] TRIVEDI, A., SRINIVASAN, D., SANYAL, K., AND GHOSH, A. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation 21*, 3 (2016), 440–462.

[97] VAN BEERS, W. C. M., AND KLEIJNEN, J. P. C. Kriging for interpolation in random simulation. *Journal of the Operational Research Society 54*, 3 (2003), 255–262.

[98] VAN DER MAATEN, L., AND HINTON, G. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, 86 (2008), 2579–2605.

[99] WAGER, S., HASTIE, T., AND EFRON, B. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *The Journal of Machine Learning Research 15*, 1 (2014), 1625–1651.

[100] WANG, H., OLHOFER, M., AND JIN, Y. A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. *Complex & Intelligent Systems 3*, 4 (2017), 233–245.

[101] WIERZBICKI, A. P. The use of reference objectives in multiobjective optimization. In *Multiple Criteria Decision Making Theory and Application*, G. Fandel and T. Gal, Eds. Springer, 1980, pp. 468–486.

[102] WIERZBICKI, A. P. A mathematical basis for satisficing decision making. *Mathematical Modelling 3*, 5 (1982), 391–405.

[103] WIERZBICKI, A. P. Convergence of interactive procedures of multiobjective optimization and decision support. In *Essays In Decision Making: A Volume in Honour of Stanley Zionts*, M. H. Karwan, J. Spronk, and J. Wallenius, Eds. Springer, Berlin, Heidelberg, 1997, pp. 19–47.

[104] XIN, B., CHEN, L., CHEN, J., ISHIBUCHI, H., HIROTA, K., AND LIU, B. Interactive multiobjective optimization: A review of the state-of-the-art. *IEEE Access 6* (2018), 41256–41279.

[105] YANG, D., DI STEFANO, D., TURRIN, M., SARIYILDIZ, S., AND SUN, Y. Dynamic and interactive re-formulation of multi-objective optimization problems for conceptual architectural design exploration. *Automation in Construction 118* (2020), article 103251.

[106] ZHANG, Q., AND LI, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation 11*, 6 (2007), 712–731.

[107] ZHEN, L., LI, M., CHENG, R., PENG, D., AND YAO, X. Adjusting parallel coordinates for investigating multi-objective search. In *Simulated Evolution and Learning* (Cham, 2017), Y. Shi, K. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, and Y. Jin, Eds., Springer, pp. 224–235.

[108] ZITZLER, E., DEB, K., AND THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation 8*, 2 (2000), 173–195.

[109] ZITZLER, E., AND KÜNZLI, S. Indicator-based selection in multiobjective search. In *International Conference on Parallel Problem Solving from Nature - PPSN VIII* (Berlin, Heidelberg, 2004), X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, Eds., Springer, pp. 832–842.

[110] ZITZLER, E., LAUMANNS, M., AND THIELE, L. SPEA2: Improving the strength pareto evolutionary algorithm. In *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems* (Athens, Greece, 2001), K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., pp. 95–100.

[111] ZITZLER, E., AND THIELE, L. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation 3*, 4 (1999), 257–271.

[112] ZITZLER, E., THIELE, L., LAUMANNS, M., FONSECA, C. M., AND DA FONSECA, V. G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation 7*, 2 (2003), 117–132.

ORIGINAL PAPERS

PI

AUTOMATIC SURROGATE MODELLING TECHNIQUE SELECTION BASED ON FEATURES OF OPTIMIZATION PROBLEMS

by

Bhupinder Singh Saini, Manuel Lopez-Ibanez, Kaisa Miettinen 2019

Proceedings of the Genetic and Evolutionary Computation Conference Companion, Edited by M. Lopez-Ibanez, ACM, NY, USA, 1765 – 1772

https://doi.org/10.1145/3319619.3326890

Automatic Surrogate Modelling Technique Selection based on Features of Optimization Problems

BHUPINDER SINGH SAINI, University of Jyvaskyla
MANUEL LÓPEZ-IBÁÑEZ, Alliance Manchester Business School, University of Manchester, UK
KAISA MIETTINEN, University of Jyvaskyla

A typical scenario when solving industrial single or multiobjective optimization problems is that no explicit formulation of the problem is available. Instead, a dataset containing vectors of decision variables together with their objective function value(s) is given and a surrogate model (or metamodel) is build from the data and used for optimization and decision-making. This data-driven optimization process strongly depends on the ability of the surrogate model to predict the objective value of decision variables not present in the original dataset. Therefore, the choice of surrogate modelling technique is crucial. While many surrogate modelling techniques have been discussed in the literature, there is no standard procedure that will select the best technique for a given problem.

In this work, we propose the automatic selection of a surrogate modelling technique based on exploratory landscape features of the optimization problem that underlies the given dataset. The overall idea is to learn offline from a large pool of benchmark problems, on which we can evaluate a large number of surrogate modelling techniques. When given a new dataset, features are used to select the most appropriate surrogate modelling technique. The preliminary experiments reported here suggest that the proposed automatic selector is able to identify high-accuracy surrogate models as long as an appropriate classifier is used for selection.

CCS Concepts: • **Computing methodologies → Supervised learning by regression**;

Additional Key Words and Phrases: surrogate modelling, automatic algorithm selection, exploratory landscape analysis

ACM Reference Format:
Bhupinder Singh Saini, Manuel López-Ibáñez, and Kaisa Miettinen. 2019. Automatic Surrogate Modelling Technique Selection based on Features of Optimization Problems. 1, 1 (August 2019), 13 pages.
https://doi.org/10.1145/3319619.3326890

1 INTRODUCTION

Increasingly, data is being used as the starting point of analysis of problems and optimization. Alternatives, such as running simulations or conducting physical experiments, may be computationally expensive, financially expensive or both. Data, on the other hand, can be analyzed cheaply and efficiently, while leading to equally relevant insights. One of the ways to use data is to create metamodels or surrogate models that try to replicate the behaviour of the simulations or real-life phenomena. Creating these surrogate models is comparatively cheap and the models can be used for descriptive,

Authors' addresses: Bhupinder Singh Saini, University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora) FI-40014 University of Jyvaskyla, Finland, bhupinder.s.saini@jyu.fi; Manuel López-Ibáñez, Alliance Manchester Business School, University of Manchester, UK, manuel.lopez-ibanez@manchester.ac.uk; Kaisa Miettinen, University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora) FI-40014 University of Jyvaskyla, Finland, kaisa.miettinen@jyu.fi.

predictive or prescriptive analysis of the problem. Using surrogate models for prescriptive analysis, such as optimization of the problem, is particularly beneficial as optimization may require multiple calls to an expensive objective function, which can be replaced with a cheap surrogate model. However, accurate modelling of data is important for such analysis.

In many papers, there is no reasoning behind the selection of a surrogate modelling technique apart from the popularity of the technique in the research community and the researcher's familiarity with it. In the absence of an evidence-based guide to select a surrogate modelling technique, choices made based on experience or popularity may be far from optimal [2]. This may lead to inaccurate analysis, which at best is a waste of time and resources and, at worst, may lead to ill-informed decisions. A possible approach for selecting surrogate modelling techniques would be to train a large number of techniques in the given dataset, which may require considerable time and, in the worst case, overfit to the particular dataset. Cross-validation approaches may somewhat overcome this over-fitting. However, they require setting aside part of the available data for validation of the models. Not only the validation data could otherwise have been used for training, leading to a better modelling of the problem; but also a selection based on cross-validation only makes use of the available dataset when making a decision, without any knowledge about similar datasets or optimization problems. Hence, a better selection approach is desirable, which can help us choose a good surrogate modelling technique for a problem instance, without sacrificing useful data, while being computationally efficient.

A similar problem is faced when choosing an optimization algorithm to tackle an optimization problem, due to the large number of algorithms available and the lack of precise guidelines for choosing the most appropriate algorithm for a given problem. In the field of automatic algorithm selection [10], an optimization algorithm is selected for a given problem instance by computing features of the problem instance that help to predict the performance of the available algorithms. In particular, features generated using explorative landscape analysis (ELA) have been used to automatically select optimization algorithms for continuous black-box optimization problems with promising results [6]. Similar techniques have been applied to combinatorial optimization problems [13]. Similar to the assumptions made by automatic algorithm selection methods, we expect that there are classes of optimization problems for which certain surrogate modelling techniques perform better than others.

Instead of selecting an optimization algorithm for a given optimization problem, we propose here to select a surrogate modelling technique for a given dataset by using classification techniques from machine learning. Moreover, by assuming that the dataset originates from an underlying but unknown optimization problem, we propose that the classifier learns from ELA features, among others. Our working hypothesis is that, due to common properties of the optimization problems underlying the datasets, ELA features may give enough information to identify a good surrogate modelling technique, if we can collect enough training data to characterize new, i.e., unseen problems.

The classifier is trained on known optimization benchmark functions from which it is easy to sample datasets with diverse features and evaluate the accuracy of various surrogate modelling techniques. The computational cost of this training phase may be large, but it is incurred only once "offline". When the automatic selection system is applied to an unseen dataset, the system only has to compute dataset features in order to select the appropriate surrogate modelling technique to model the dataset.

In this paper, we evaluate the above proposal using a collection of datasets generated from benchmark and real-world engineering problems from the literature. We also compare the performance of several classifiers when used as the selector of the proposed system. Our preliminary results show that the proposed system is able to select good surrogate modelling techniques for almost all engineering problems.

The paper is structured as follows. Section 2 describes the general idea of our proposal for automatic selection of surrogate modelling techniques. In Section 3, we evaluate this proposal experimentally on benchmark and real-world

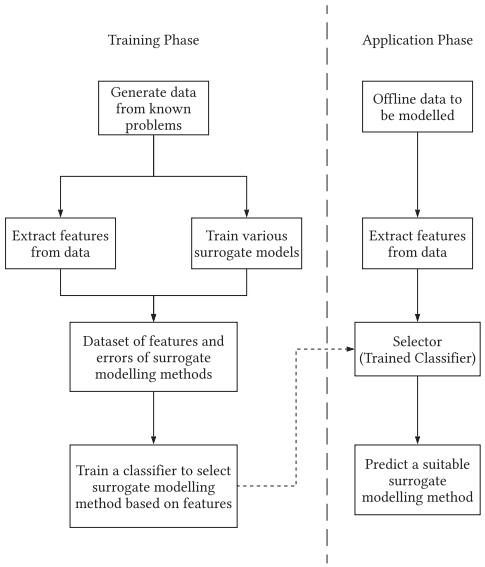


Fig. 1. Automatic selection of surrogate modelling techniques based on optimization features.

engineering problems by studying a proof-of-concept system that automatically chooses among ten surrogate modelling techniques. In addition, we compare the performance of nine classifiers as the selection method used by our automatic selection system. Finally, we summarise our conclusions in Section 4 and point out ongoing and future work.

2 AUTOMATIC SELECTION OF SURROGATE MODELLING TECHNIQUES

We propose here to automatically select the most appropriate technique for building a surrogate model of a given optimization problem based on features that have been traditionally used for landscape analysis of optimization algorithms. Surrogate modelling technique selection (SMTS) uses machine learning techniques to learn and predict which surrogate modelling techniques would perform best on given data based on certain "features" of the dataset rather than individual samples. In our SMTS framework, a classifier that we will call a "*selector*" is trained on landscape features of optimization problems and on the performance of various surrogate modelling techniques. Later, in an application phase, given a dataset to be modelled from an underlying optimization problem, the trained selector will select just one surrogate modelling technique based on features of the given data. By assuming a data-driven context, the application phase is constrained to the data already available and there is no possibility of generating new data from the same underlying problem to evaluate the performance of alternative surrogate modelling techniques.

An outline of the proposed automatic SMTS framework is shown in Figure 1. There are two clearly delimited training and application phases. The training phase proceeds as follows:

(1) **Data generation**: Training datasets are generated from well known optimization problems. Each training dataset corresponds to a sample of solutions and corresponding objective function values. To ensure the applicability of the SMTS to a wide range of real-life problems, the individual datasets should differ not only in the characteristics of the problems, e.g., number of decision variables, but also in the features of the sampling: different numbers

of samples, uniform vs. non-uniform distributions of samples in the decision space, presence of noise and/or missing data, etc.

(2) **Surrogate model training**: A previously chosen set of surrogate modelling techniques is trained on each dataset. These techniques will form the choices available to the selector. Then, the accuracy of each surrogate model is evaluated by using a sample of solutions and objectives values from the same problem but different from those used for training, and calculating its R^2 coefficient.

(3) **Feature calculation**: Independently of the training of each surrogate model, we also calculate a relatively large number of features for each training dataset that hopefully help to characterize the landscape of the optimization problem that underlies the dataset.

(4) **Training of the selector**: The features of each training dataset together with the R^2 value of each surrogate modelling technique on the problem forms the training data for a classifier that aims to predict the best surrogate modelling technique (i.e., with the highest R^2) for each dataset based on its features.

(5) **Custom cost function:** We have observed that there are large differences among various surrogate modelling techniques and, often, the second-best technique is almost as good as the best one, while both of them are significantly better than the worst one. In other words, when failing to select the best surrogate modelling technique, selecting instead the second-best is much better than selecting the worst one. Therefore, we use the following cost function to quantify the performance of the selector relative to the best-performing surrogate modelling technique:

$$Loss(Dataset_j) = \max_{k \in K}\{R^2_{k,j}\} - R^2_{\text{selected},j} \tag{1}$$

$$Cost = \frac{\sum_{j=1}^{J} Loss(Dataset_j)}{J}, \tag{2}$$

where

$Dataset_j$ = Dataset with index j
$R^2_{i,j}$ = R^2 value of surrogate modelling technique (SMT) i applied to dataset j
$R^2_{\text{selected},j}$ = R^2 value of the SMT selected by the selector for dataset j
K = Total number of SMTs available for selection
J = Total number of datasets included in cost evaluation.

The best-performing surrogate modelling technique on a particular dataset produces the maximum R^2 value and, if selected, it results in a loss value of zero. Selecting any other surrogate modelling technique will produce a higher loss value. The cost metric aggregates the loss values over multiple datasets.

Once the selector has been trained using the training datasets, it may be used to select a surrogate modelling technique on unseen data. In this application phase, we have access to some features of the sampling, such as the number of samples, presence of noise or missing data, etc. Since we assume that the unseen data arises from an unknown optimization problem, we can also calculate landscape features of the underlying optimization problem from the data available. However, we do not have the possibility of creating new data and we do not have access to the underlying optimization problem. Moreover, we do not attempt to train each surrogate modelling technique on the available data. Instead, we use the selector trained in the previous phase to select one surrogate modelling technique according to

the features that can be computed on the given sample data. The selected surrogate modelling technique will then be trained on the available data and used for analysis, prediction and, possibly, for data-driven optimization.

3 EXPERIMENTAL STUDY

We evaluate here a proof-of-concept of our proposed automatic selection of surrogate modelling techniques based on features of optimization problems. The first goal of our experimental analysis is to understand whether there are sufficient differences among standard surrogate modelling techniques to justify the computational cost of training a selector. To answer this question, we consider 10 popular surrogate modelling techniques (Table 2) and evaluate them on datasets generated from standard numerical optimization benchmark functions.

A second question is whether training a selector in the manner proposed above can identify a good surrogate modelling technique for a given dataset, which is at least better than a random selection. Of course, an important component of the proposed selector is the classification method used for selection. Hence, we evaluate here nine different classifiers (Table 3). Moreover, a key characteristic of our proposal is that, given an unseen dataset, we wish to select a surrogate modelling technique based on features of the underlying optimization problem extracted from the available dataset, without training any of the surrogate models, not even on a sub-sample of the dataset. The latter could in principle help to inform the selection, but it may obscure the contribution of the landscape features, which is what we are testing here.

Finally, we also want to assess whether a selector trained on data from benchmark functions is better than chance in identifying a good surrogate modelling technique for real-world engineering problems.

3.1 Experimental Setup

Benchmark Datasets (Training). For the training phase, we generate datasets from well-known benchmark problems. In particular, we consider separately single functions from several multiobjective benchmark test suites (DTLZ [3], WFG [4] and ZDT [14]). The number of decision variables, i.e., the dimensionality of the decision space, was varied between 10, 20 and 30 for the DTLZ and WFG sets, and the recommended dimensionality for ZDT (30 in most cases).

Table 1. Characteristics of datasets generated from benchmark problems.

Benchmark problem family:	DTLZ functions {1–5}, ZDT funcs. {1–9}, WFG funcs. {1–4, 6}
Decision variables:	{10, 20, 30} (ZDT: only recommended values)
Number of samples:	{100, 150, 200, 250, 300, 350, 400, 500, 600, 700, 800, 900, 1000, 1200, 1500, 2000}
Distribution of samples (decision space):	{Uniform, Normal with $\mu = 0.5$, sd $= 0.25$}
Missing data:	{None, Missing}

Data from real-life problems may have characteristics that are not usually found in uniformly distributed datasets generated from benchmark functions. These may include skewed distributions of samples in the data, presence of noise, and chunks of the decision space may be missing from the data. Datasets may also have too few samples for a surrogate modelling technique to work correctly, or some surrogate modelling techniques may benefit more than others from having a large number of samples. We account for these characteristics by sampling several training datasets from each benchmark problem. In particular, we sample datasets with various sizes ranging from 100 to 2 000 solutions. Samples are also generated either from a uniform random distribution in the decision space or they are normally distributed

Table 2. Surrogate modelling techniques available for automatic selection.

Surrogate modelling technique	Keyword
Support vector machine	SVR
Neural networks	NN
Adaptive boosted regression	Ada
Gaussian process regression	GPR
Stochastic gradient descent	SGD
K Nearest neighbour	KNR
Decision trees	DTR
Random forest	RFR
Extra trees regression	ExTR
Gradient boosted regression	GBR

with a mean equal to 0.5 and standard deviation equal to 0.25 for each decision variable. Some datasets have chunks of data missing, which is done by creating the dataset as previously described, and then removing data which lie within a rectangular hyperbox one-tenth of the size of the decision space. Table 1 summarizes all the variations of training datasets included in our training phase.

Engineering Datasets (Validation). For evaluating the performance in the application phase, we generated datasets from several box-constrained engineering problems from the literature:

(1) Kursawe test function [7]
(2) Four bar plan truss problem [1]
(3) Gear train design [11]
(4) Pressure vessel design [11]
(5) Speed reducer problem [12]
(6) Welded beam design problem [11]
(7) Unconstrained function problem 1 [8]

In the case of multiobjective optimization problems, each objective function was treated as a separate problem, leading to a total of 12 problems. For each problem, we sample several datasets of sizes {50, 100, 150, 200, 400, 700, 1000}, sampling from the decision space uniformly at random. This resulted in a total of 84 datasets created from the engineering problems. The number of decision variables ranges from 3 to 7 and there is no missing data.

Dataset Features. In order to calculate landscape features of each dataset that hopefully characterize the underlying optimization problem, we used the R package `FLACCO` [5]. The features considered here belong to the set of "simple" or ELA metamodel features, which are calculated by creating linear and quadratic models on the dataset. The parameters and accuracy of these models, such as the intercept of the linear model or the adjusted R-square of the models, form the features of the dataset, resulting in a total of 10 features. In addition to the landscape features, three other features were included per dataset: the dimensionality of the decision space, the number of samples, and a Boolean variable representing whether the sampling was uniformly distributed or not. Table 4 describes all features.

Surrogate Modelling Techniques. We consider 10 popular surrogate modelling techniques (Table 2), from which the selector must choose one. In particular, we use the standard implementations of these techniques available in the Python package `scikit-learn` [9] with their default hyper-parameters. Surrogate models are trained and evaluated on

Table 3. Classification methods evaluated for the selector.

Classifier method	Keyword
Bagging	BC
Support vector machine	SVC
K-nearest neighbor	KNC
Nearest centroid	NC
Gaussian process	GPC
Decision tree	DTC
Neural network	NNC
Extremely randomized tree	ExTC1
Extra-trees	ExTC2

Table 4. Dataset features used for selector training and application.

Feature Name	Description
ela_meta.lin_simple.adj_r2	Adjusted R^2 of a simple linear model
ela_meta.lin_simple.intercept	Intercept of a simple linear model
ela_meta.lin_simple.coef.min	Smallest non-intercept coefficient of the linear simple model
ela_meta.lin_simple.coef.max	Largest non-intercept coefficient of the linear simple model
ela_meta.lin_simple.coef.max_by_min	Ratio of the largest and smallest non-intercept coefficient of the linear simple model
ela_meta.lin_w_interact.adj_r2	Adjusted R^2 of a simple linear model with interactions
ela_meta.quad_simple.adj_r2	Adjusted R^2 of a simple quadratic model
ela_meta.quad_simple.cond	The ratio of the biggest and smallest cofficients of the simple quadratic model
ela_meta.quad_w_interact.adj_r2	Adjusted R^2 of a quadratic model with interactions
ela_meta.costs_runtime	Runtime for the computation of the features
numsamples	Number of samples in the problem instance
dimensionality	Dimensionality of the decision space of the problem instance
is_uniform	True if data is uniformly distributed in decision space, otherwise false

one dataset at a time by randomly splitting the dataset into 70% solutions used for training and 30% solutions used to compute an R^2 value to measure the performance of the surrogate model.

Selector. In our proposal, the selector that chooses a surrogate modelling technique is a classifier. Since we do not have any intuition about which classifier may perform best for this task, we evaluate here nine alternative classifiers (see Table 3). We use the standard implementation of these classifiers available from `scikit-learn` with default hyper-parameters. To compare the classifiers, we use the custom cost function described by Eq. 2. The dataset features explained above together with the R^2 values of each surrogate modelling technique on each dataset are the input data for the classifier during the training phase, while only the dataset features are available to the classifier on the application phase when selecting a surrogate model.

Source Code. The SMTS code can be found at https://github.com/industrial-optimization-group/SurrogateAgents2/releases/tag/GECCO2019

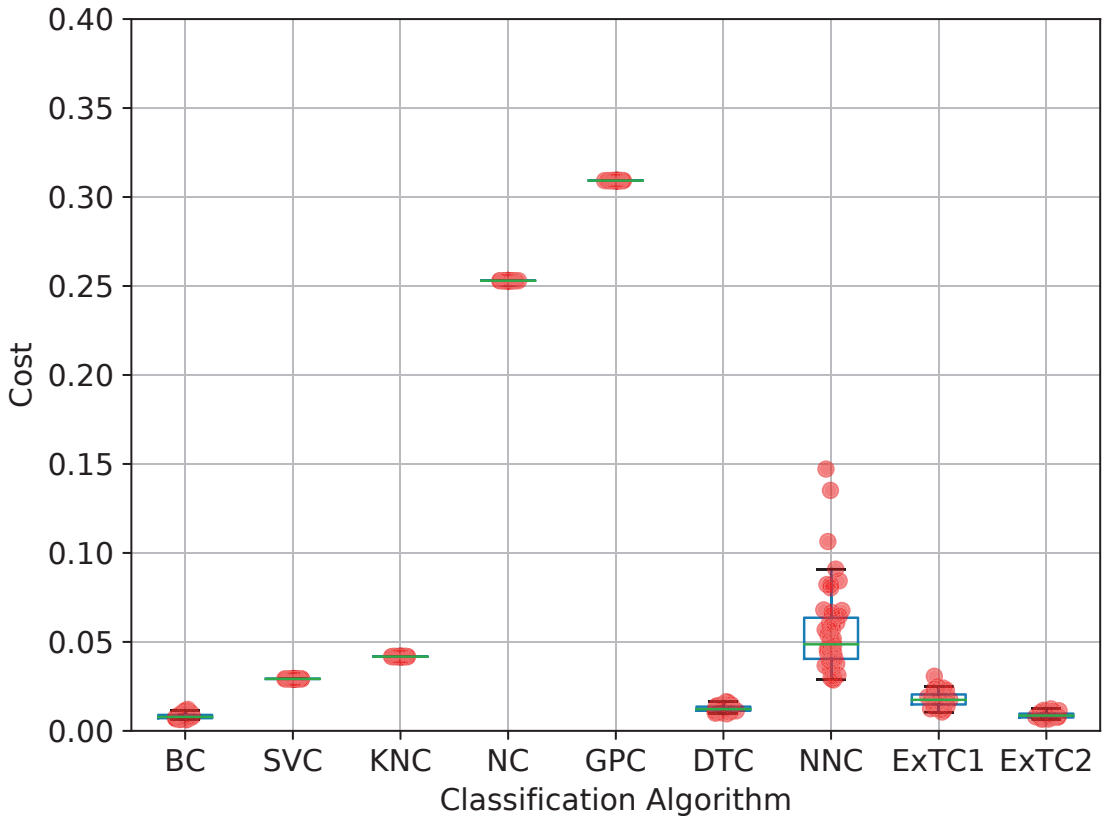


Fig. 2. Cost values of each classifier on the benchmark datasets. Each point is the cost value of the surrogate modelling technique selected by the corresponding classifier for one random 70%/30% split of the benchmark datasets. Each boxplot contains 50 cost values.

3.2 Experimental Results

As the first experiment, we train each of the classifiers in Table 3 on 70% of the benchmark datasets and perform automatic selection of a surrogate modelling technique on the remaining 30% datasets to calculate a cost value.[1] We repeat this step 50 times with different random splits of the benchmark datasets, thus obtaining 50 cost values per classifier, which are shown as boxplots in Figure 2.

The plot shows that most classifiers were able to select a high-performing surrogate model, with cost values very close to zero. Of particular note are the BC, SVC, KNC, DTC, ExTC1 and ExTC2 classifiers, all of which had a cost value below 0.05 consistently. Given the random 70%/30% split of the datasets, the same exact dataset is not available for training and selection. However, the same underlying optimization problem may appear in both phases. Nevertheless, some classifiers performed very poorly, which suggests that selecting the best-performing classifier is not trivial.

[1] As explained above, when training and evaluating each surrogate modelling technique on each dataset, the dataset itself is also randomly split into 70% points used for training and 30% points used for computing R^2 values.

In the second experiment, we focus on the best trained variant of each classifier from the first experiment and evaluate it on the engineering datasets. In other words, we choose as a selector for the application phase the trained version of each classifier with the lowest cost out of the 50 repetitions of the training phase performed above. We then use this selector to select surrogate modelling techniques for each engineering dataset. More specifically, a random (training) 70% subset of each engineering dataset is used to compute features, the classifier uses these features to select a surrogate model, the selected surrogate model is trained on the same training subset and its R^2 value is computed on the remainder 30% data of the dataset. We also calculate the R^2 value of all other surrogate modelling techniques so that we can identify the best-performing technique and compute the loss function (Eq. 1). Figure 3 shows the loss values of all the selectors for all 84 engineering datasets. In this case, obtaining a perfect score is much more difficult, as the selector was trained in samples of features generated from the benchmark datasets that are quite different from the feature samples generated from the engineering datasets. Nevertheless, the performance of the various classifiers follows a similar pattern as in the first experiment. In particular, BC, KNC, SVC, ExTC1 and ExTC2 again perform better than the rest.

Table 5 shows the means and sample standard deviations of the loss values of each classifier over all engineering datasets. By looking at the mean loss values we can say that the SVC selector performed the best with a mean value of 0.052 and a standard deviation of 0.096. This means that the classifier is selecting good surrogate modelling techniques, among the ones available, for most of the engineering problems. However, we noticed that the SVC selector always selected the same surrogate modelling technique (GBR) for all datasets. This technique belongs to the group of ensemble based techniques, all of which performed very well for most of the problems. By contrast, the second best selector, ExTC2, selected different surrogate models for different engineering datasets, and still achieved excellent loss values.

Table 5. Mean and standard deviation of loss values of selection on engineering data.

Classification method	Mean loss	Standard deviation of loss
BC	0.1072	0.2369
SVC	**0.0518**	**0.0963**
KNC	0.1798	0.3407
NC	0.3887	0.4153
GPC	0.7351	0.3723
DTC	0.2516	0.3972
NNC	0.1553	0.2730
ExTC1	0.0979	0.1895
ExTC2	**0.0853**	**0.1689**

Finally, we compare the actual R^2 values of the selected surrogate models versus the rest. As described above, all surrogate modelling techniques were trained on a random 70% subset of each engineering dataset and their R^2 value is calculated on the remainder 30%. Figure 4 shows the mean R^2 values (as lines, with a 95% confidence interval as a shaded area) of each surrogate modelling technique over all datasets generated from each engineering problem (x-axis). Good surrogate models have R^2 values close to 1 with a low variance. Interestingly, popular surrogate modelling techniques such as support vector machines (SVM), neural networks (NN), and Gaussian processes (GPR), had a significantly worse accuracy than ensemble methods such as extra-trees regression (ExTR) and gradient boosted regression (GBR).

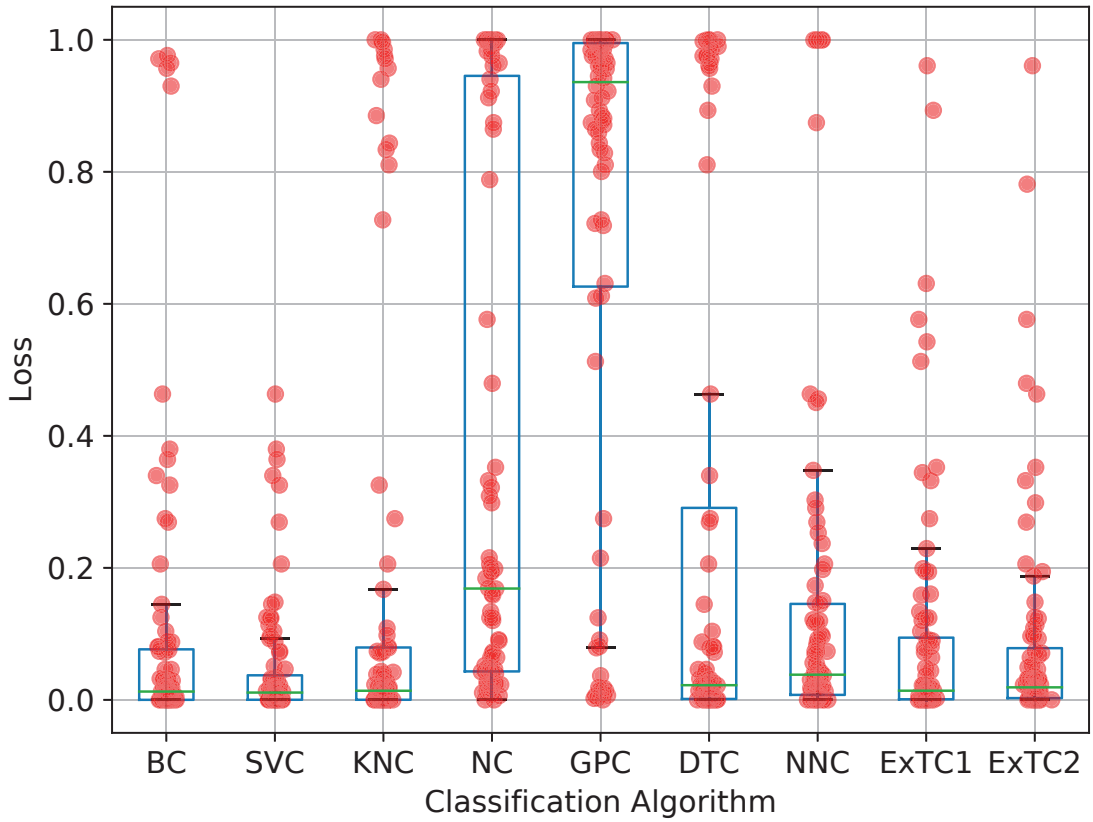


Fig. 3. Loss values of each classifier on the benchmark datasets. Each point is the loss value of the surrogate modelling technique selected by the corresponding classifier for one engineering dataset after training the classifier on the benchmark datasets. Each boxplot contains 84 loss values.

The black line in Figure 4 shows the mean R^2 values obtained when the SVC classifier selects the surrogate modelling technique for each dataset. Figure 5 shows the same data and, instead, the black line indicates the mean R^2 values obtained by the surrogate modelling technique selected by ExTC2 classifier. These plots show that the SVC classifier has a lower mean loss value than ExTC2 because it performs significantly better in the `pressure_f1` and the `unconstrained_f_f1` problem. Although neither classifier is able to always select the surrogate model with the highest R^2 value, both are always able to select surrogate models with R^2 larger than 0.8, except for problems `gear_train_f1` and `unconstrained_f_f1`. Given the behavior of the surrogate models on these two problems, it seems likely that either our training phase does not have benchmark problems with the appropriate features, or we are lacking the features required to characterize these problems.

4 CONCLUSIONS

In this paper, we have proposed the automatic selection of surrogate modelling techniques for a given dataset by using features that aim at characterizing the underlying optimization problem. We describe here a proof-of-concept system

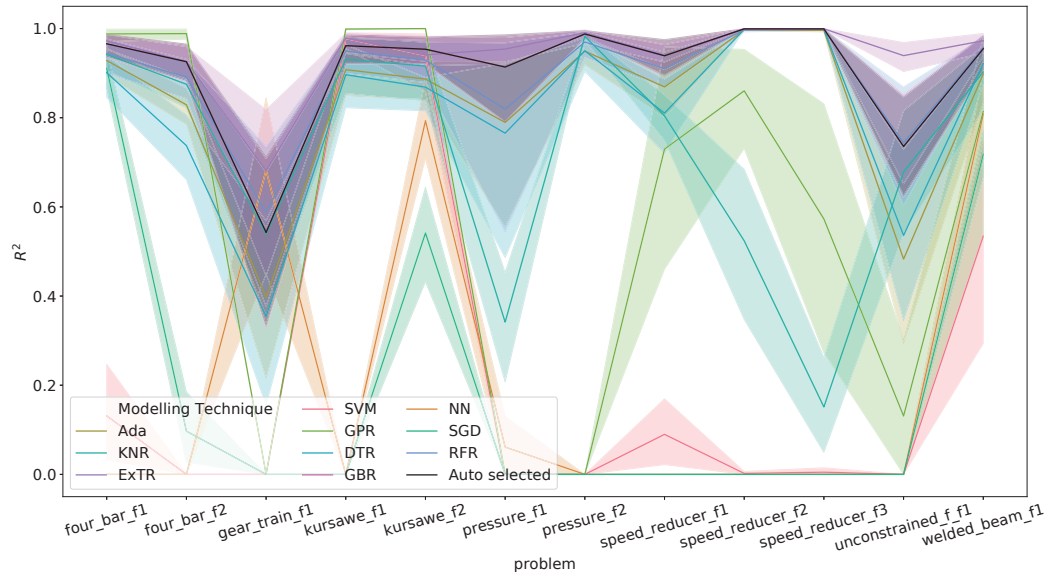


Fig. 4. Mean R^2 values (and 95% confidence interval as shaded area) of each surrogate modelling technique when applied to all datasets generated from each engineering problem (x-axis). The black line indicates the mean R^2 value when SVC is used to select a surrogate modelling technique for each dataset.

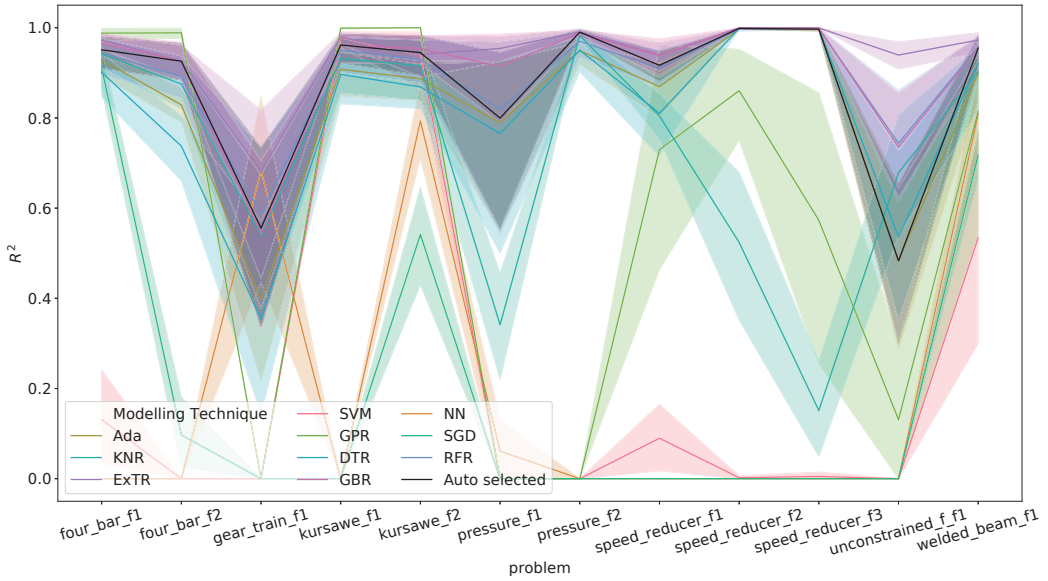


Fig. 5. Mean R^2 values (and 95% confidence interval as shaded area) of each surrogate modelling technique when applied to all datasets generated from each engineering problem (x-axis). The black line indicates the mean R^2 value when ExTC2 is used to select a surrogate modelling technique for each dataset.

that uses exploratory landscape features provided by the FLACCO package in addition to the dimensionality of the data, the number of points in the dataset and whether the dataset is a uniform sample or not. These features are used to select among ten available surrogate modelling techniques. For evaluating our proposal, we have generated a diverse set of datasets from popular benchmark functions as well as real-world engineering problems. In addition, we have compared nine different classifiers to be used as the selector of the proposed automatic surrogate model selection technique.

The preliminary experimental results presented in this work show that the choice of a classifier to be used as a selector is crucial, with significant differences in performance between classifiers. In addition, the best classifiers for benchmark datasets turned out to also perform well on engineering datasets. Another interesting result was that, despite the fact that no single surrogate modelling technique was always the best for all datasets, the automatic selection method was able to select surrogate modelling techniques with R^2 values higher than 0.8 in almost all datasets, when using the best or second-best classifier as a selector.

Although this research is at a very preliminary stage and there is much that could be improved, we believe that our results already show the potential of automatically selecting surrogate models for unseen data by training on datasets generated from known optimization problems.

Among the obvious improvements, we plan to simplify the creation and use of cross-validation sets for training and validating the surrogate models as well as the selector to make better use of the available data while keeping the benefits of separating training and validation data. In addition, we evaluated the various classifiers using a custom loss function. However, the training of the classifier could be itself cost-sensitive. Moreover, as surrogate modelling techniques are frequently used in conjunction with optimization algorithms, the quality of the optimal solutions obtained can also be used as a metric for training the Selector. The Selector, thus, is trained to select surrogate modelling techniques that perform well when used along with the considered optimization algorithms. We also considered "only" ten surrogate modelling techniques with default hyperparameter settings. Although this is a much larger number than what is usually considered in the data-driven surrogate modelling literature, it should be possible to select not only from an even larger pool, but also from various sets of hyperparameter settings. Moreover, we plan to analyze the importance of the features used by the selector. This will give us a better understanding of which characteristics of the problem instances are desirable as features, and will enable us to explore a much larger landscape of features productively. Finally, we plan to include more diverse training datasets, as our results suggest that our current training phase does not capture the characteristics of some real-world engineering problems.

ACKNOWLEDGMENTS

This research was partly supported by the Academy of Finland (grant number 287496, project DESDEO). This related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization, jyu.fi/demo) of the University of Jyvaskyla.

REFERENCES

[1] F. Y. Cheng and X. S. Li. 1999. Generalized center method for multiobjective engineering optimization. *Engineering Optimization* 31, 5 (1999), 641–661.

[2] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen. 2019. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing* 23, 9 (2019), 3137–3166.

[3] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. 2005. Scalable Test Problems for Evolutionary Multiobjective Optimization. In *Evolutionary Multiobjective Optimization*, A. Abraham et al. (Eds.). Springer, London, UK, 105–145.

[4] S. Huband, P. Hingston, L. Barone, and L. While. 2006. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation* 10, 5 (2006), 477–506.

[5] P. Kerschke and H. Trautmann. 2016. The R-package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In *Proceedings of the 2016 Congress on Evolutionary Computation (CEC 2016)*. IEEE Press, Piscataway, NJ, 5262–5269.

[6] P. Kerschke and H. Trautmann. 2019. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation* 27, 1 (2019), 99–127.

[7] F. Kursawe. 1991. A variant of evolution strategies for vector optimization. In *Proceedings of PPSN-I, First International Conference on Parallel Problem Solving from Nature*, H.-P. Schwefel and R. Männer (Eds.). Springer, Berlin, Heidelberg, 193–197.

[8] M. Mahdavi, M. Fesanghary, and E. Damangir. 2007. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* 188, 2 (2007), 1567–1579.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[10] J. R. Rice. 1976. The Algorithm Selection Problem. *Advances in Computers* 15 (1976), 65–118.

[11] E. Sandgren. 1990. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design* 112, 2 (1990), 223–229.

[12] J. N. Siddall. 1982. *Optimal Engineering Design: Principles and Applications*. Marcel Dekker Inc., New York.

[13] K. Smith-Miles. 2008. Cross-disciplinary Perspectives on Meta-learning for Algorithm Selection. *Comput. Surveys* 41, 1 (2008), 1–25.

[14] E. Zitzler, L. Thiele, and K. Deb. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation* 8, 2 (2000), 173–195.

PII

A NEW PARADIGM IN INTERACTIVE EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATION

by

Aaltola, Kirsi, 2021

Bhupinder Singh Saini, Jussi Hakanen, Kaisa Miettinen 2020

Parallel Problem Solving from Nature – PPSN XVI, Edited by T. Back, M. Preuss, A Deutz, H.Wang, C. Doerr, M. Emmerich, H. Trautmann, Springer, Cham, 243 – 256

https://doi.org/10.1007/978-3-030-58115-2_17

A New Paradigm in Interactive Evolutionary Multiobjective Optimization

Bhupinder Singh Saini[1][0000−0003−2455−3008], Jussi Hakanen[1][0000−0001−9579−8657], and Kaisa Miettinen[1][0000−0003−1013−4689]

University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland
{bhupinder.s.saini,jussi.hakanen,kaisa.miettinen}@jyu.fi

Abstract. Over the years, scalarization functions have been used to solve multiobjective optimization problems by converting them to one or more single objective optimization problem(s). This study proposes a novel idea of solving multiobjective optimization problems in an interactive manner by using multiple scalarization functions to map vectors in the objective space to a new, so-called preference incorporated space (PIS). In this way, the original problem is converted into a new multiobjective optimization problem with typically fewer objectives in the PIS. This mapping enables a modular incorporation of decision maker's preferences to convert any evolutionary algorithm to an interactive one, where preference information is directing the solution process. Advantages of optimizing in this new space are discussed and the idea is demonstrated with two interactive evolutionary algorithms: IOPIS/RVEA and IOPIS/NSGA-III. According to the experiments conducted, the new algorithms provide solutions that are better in quality as compared to those of state-of-the-art evolutionary algorithms and their variants where preference information is incorporated in the original objective space. Furthermore, the promising results require fewer function evaluations.

Keywords: Interactive methods · Achievement scalarizing functions · Evolutionary algorithms · Preference information · Decision maker

1 Introduction

Many multiobjective optimization problems (MOPs) are encountered in real life applications. Due to the conflicting nature of objectives in these problems, often there does not exist a single optimal solution. Instead, there exists a set of *Pareto optimal solutions* which represent trade-offs among the various objectives.

One family of methods, known as *a posteriori* methods, solve MOPs by finding a set of solutions which adequately represents the entire set of Pareto optimal solutions [18]. Evolutionary algorithms (EAs) have been employed for this with a varying degree of success. An example of such methods is NSGA-II [6], which works well for solving problems with a low number of objectives, but the performance degrades as the number of objectives increases [9]. Recent a posteriori EAs have tackled this problem in various ways [14].

However, increasing the number of objectives brings forth new challenges. As the number of objectives increases, the number of solutions required to adequately represent the set of Pareto optimal solutions (which may have an infinite number of solutions) increases exponentially [9, 14]. Regardless of the number of objectives, only one or few of these solutions are useful to a *decision maker* (DM) who wants to find and implement the desirable solution. Hence, when using a posteriori methods, computational resources are wasted on finding solutions that are not relevant. If objective function evaluations require time-consuming simulations or physical experiments, this problem is compounded and may lead to a waste of monetary resources as well. Moreover, these algorithms leave the task of choosing the final solution to the DM. As each solution is a vector in a high-dimensional objective space, comparing potentially thousands of solutions is a difficult task. This process can set a high cognitive load on the DM.

As DMs are experts in their domain, they usually have opinions or *preferences* regarding which solutions are desirable or undesirable to them. The preference information may be elucidated in the form of desirable objective function values, ranking of importances of objectives, pair-wise comparison of solutions and many other techniques [16]. Recent advances in EAs try to incorporate this information to limit the scope of search of the EA. As the DM may learn new information about the problem during the solution process, allowing them to change their preferences during the solution process is desirable [11]. Methods which allow such change are known as *interactive* methods [17–19, 30]. Ideally, this leads to less waste of resources as only solutions that are preferable to the DM are focused upon. Moreover, as only a small subset of the Pareto optimal solutions is to be represented at a time, the number of solutions to be shown to the DM is smaller, hence reducing the cognitive load. However, many interactive EAs have problems ranging from addition of hyperparameters to lack of diversity in the population, which can impair the optimization process [1, 10].

The concept of utilizing the preferences of a DM in the solution process of an MOP is very popular in the field of multiple criteria decision making [18,19]. One of the methods adopted is to use scalarization functions [18,20]. These functions utilize the preferences of the DM to map the objective function values of solutions to scalar values, hence converting the MOP to one or more single objective optimization problems. Different scalarization functions interpret the same preference information differently, and may lead to different results [20]. Different solutions can hence be obtained by solving multiple scalarization functions with the same preference information, as done in synchronous NIMBUS [21], or by slightly modifying the preference information multiple times and optimizing the same scalarization function, as done in the reference point algorithm [28].

In this paper, we explore the concept of using multiple scalarization functions to create a new space: *Preference Incorporated Space* (PIS). First, we study the mathematical properties of this new space. More specifically, we study the effect of optimizing in the PIS, introducing a new paradigm in preference based optimization: Interactive Optimization using Preference Incorporated Space (IOPIS) algorithm. The IOPIS algorithm enables us to make use of preference informa-

tion with any a posteriori EA in an interactive way, as the preference information is encoded directly in the optimization problem in the PIS. It also enables us to control the dimension of the space in which dominance is judged, equal to the number of chosen scalarization functions. We then introduce the IOPIS algorithm, a modular algorithm that takes a given number of specified scalarization functions, and uses a DM's preferences to convert a generic MOP to an MOP in the preference incorporated space. This can then be solved interactively with any appropriate non-interactive EA together with DM's preferences. As examples, we implement two versions of the new algorithm: IOPIS/RVEA and IOPIS/NSGA-III, where the new problem in the PIS is optimized using decomposition based EAs RVEA [4] and NSGA-III [5], respectively.

The rest of the paper is organized as follows. Section 2 discusses the background of multiobjective optimization, EAs, and scalarization functions. Section 3 discusses the mathematical properties of the PIS and introduces the IOPIS algorithm with a visual explanation. In Section 4, we conduct an experimental study to compare the performances of the two implementations of the IOPIS algorithm with state of the art a posteriori EAs and their interactive variants and discuss the results. Finally, we draw conclusions in Section 5. All implementations and experimental data presented in this paper are open source and publicly available at `https://desdeo.it.jyu.fi` as a part of the DESDEO framework.

2 Background

2.1 Multiobjective Optimization

An MOP can be defined as:

$$\begin{aligned} &\text{minimize } \{f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ &\text{subject to } \mathbf{x} \in S, \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$ are vectors of decision variables belonging to the feasible set $S \subset \mathbb{R}^n$. The $k\,(\geq 2)$ objective functions f_i map vectors of S to $\mathbb{R}$. The objective function values $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))$ form objective vectors in the objective space $\mathbb{R}^k$. A solution $\mathbf{x}^1 \in S$ of problem (1) is said to *dominate* another solution $\mathbf{x}^2 \in S$ (written as $\mathbf{f}(\mathbf{x}^1) \succ \mathbf{f}(\mathbf{x}^2)$) if $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2)$ for all $i = 1, \ldots, k$ and $f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$ for at least one $j = 1, \ldots, k$. Pareto optimal solutions are solutions of the MOP which are not dominated by any other solution in S. For this reason, they are also referred to as *non-dominated solutions*. Sometimes, it is desirable for DMs to consider a subset of Pareto optimal solutions with bounded trade-offs [18]. Such solutions are called *properly Pareto optimal* solutions.

We can define the set of solutions of problem (1), known as a Pareto set, as:

$$PS_{OS} = \{\mathbf{x} \in S \mid \nexists_{\mathbf{x}^* \in S}\ \mathbf{f}(\mathbf{x}^*) \succ \mathbf{f}(\mathbf{x})\}, \tag{2}$$

where the subscript OS refers to the fact that the set was obtained by considering the objective vectors in the objective space. We can now define an *ideal point* and a *nadir point* of problem (1). These points represent the lower and upper

bounds of the ranges of the objective function values among the Pareto optimal solutions, respectively. The ideal point $\mathbf{z}^* = (z_1^*, \ldots, z_k^*)$ can be calculated as $z_i^* = \min_{x \in S} f_i(\mathbf{x})$. The nadir point $\mathbf{z}^{nad} = (z_1^{nad}, \ldots, z_k^{nad})$ can be calculated as $z_i^{nad} = \max_{x \in PS_{OS}} f_i(\mathbf{x})$. It should be noted that calculating the nadir point requires the calculation of the PS_{OS}. Hence, the calculation of the nadir point is tricky in problems with more than two objectives and needs to be estimated [8,18]. Any objective vector $\mathbf{z}$ is defined to be *achievable* if $\mathbf{z}$ belongs to the set:

$$T = \{\mathbf{z} \in \mathbb{R}^k \mid \exists_{\mathbf{x} \in S}\ \mathbf{f}(\mathbf{x}) \succ \mathbf{z} \text{ or } \mathbf{f}(\mathbf{x}) = \mathbf{z}\}. \tag{3}$$

By definition, the nadir point is an achievable point, while the ideal point is not.

2.2 Evolutionary Algorithms

Decomposition-based methods such as NSGA-III [5], RVEA [4], and many variants of MOEA/D [31] have become popular in the evolutionary multiobjective optimization community. These methods decompose the objective space into sections using directional vectors called reference vectors, reference points, or weights. For simplicity, in what follows, we will be using the term reference vectors (RVs). These RVs, usually spread uniformly in the objective space, represent individual single-objective optimization problems. The RVs are typically generated using a simplex lattice design, and the number of RVs is equal to $\binom{l+k-1}{k-1}$, where l is a parameter controlling the density of the RVs. Subsets of the population which lie in the decomposed region associated with an RV (in the objective space) evolve in the direction of that RV based on scalar fitness values calculated using the RV and their objective function values.

As mentioned in the introduction, EAs which approximate the entire Pareto front exhibit many downsides. Methods have been proposed to get around those downsides by incorporating the preferences of the DM in an interactive fashion, see, e.g. [23, 26, 30]. One of the ways to incorporate a DM's preferences in decomposition-based EAs is to manipulate the spread of the RVs to account for the preferences [4,15]. In many such methods, the DM is required to provide their preferences in the form of a *reference point* in the objective space [12, 26, 27]. The components of a reference point are desirable values of each objective function, which may or may not be achievable. Then, uniformly spread RVs are translated towards this point. This translation introduces a new scalar hyperparameter which controls the final spread of the RVs around the reference point. This method introduces a few new problems, though. Firstly, the effect of changing the value of the newly introduced hyperparameter may be difficult for a DM to understand. But an appropriate value for this hyperparameter is important as it has been observed that a small spread of RVs may lead to a degradation in population diversity, which prohibits the convergence of the EA [1, 10].

2.3 Achievement Scalarizing Functions

As mentioned, scalarization functions are functions that map a vector to a real-valued scalar. The weighted sum function and the Chebyshev function used by

MOEA/D and the angle-penalized distance function used by RVEA are examples of scalarization functions [4,31]. To be regarded as a good scalarization function, it must have some desirable properties [25]. Firstly, the solutions obtained by optimizing the scalarization function should be Pareto optimal. Secondly, these solutions should be satisfactory according to the preferences of a DM, if the preferences are feasible. Finally, any Pareto optimal solution should be discoverable by changing the preferences provided by the DM.

Unfortunately, no single scalarization function satisfies the three conditions concurrently [25]. However, if we relax the conditions to only account for properly Pareto optimal solutions, rather than all Pareto optimal solutions, then all three conditions can be satisfied by some scalarization functions. In this paper, we focus on a subclass of scalarization functions, known as achievement scalarizing functions (shortened to achievement function) [29]. An achievement function is a continuous function $s : \mathbb{R}^k \to \mathbb{R}$. Achievement functions are characterized by either being strictly increasing and order-representing, or strongly increasing and order-approximating [29]. We will focus on the latter kind as they satisfy all three relaxed desirable properties.

Theorem 1. *[29] Let us consider $\mathbf{z}^1, \mathbf{z}^2 \in \mathbb{R}^k$ such that $\mathbf{z}^1 \succ \mathbf{z}^2$. Then for any order-approximating achievement function $s : \mathbb{R}^k \to \mathbb{R}$, we have*

$$s(\mathbf{z}^1) < s(\mathbf{z}^2). \tag{4}$$

From Theorem 1, it can be concluded that solving the following problem:

$$\begin{array}{c} \text{minimize } s(\mathbf{f}(\mathbf{x})) \\ \text{subject to } \mathbf{x} \in S \end{array} \tag{5}$$

will lead to a Pareto optimal solution of problem (1) [28, 29].

A general formulation of an (order-approximating) achievement function is:

$$s(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}) = \max_{i=1,\ldots,k} \left[\frac{f_i(\mathbf{x}) - \bar{z}_i}{\mu_i} \right] + \rho \sum_{i=1}^{k} \left(\frac{f_i(\mathbf{x}) - \bar{z}_i}{\mu_i} \right), \tag{6}$$

where ρ is a small positive scalar and μ_i are positive scalars and $\bar{\mathbf{z}} \in \mathbb{R}^k$ is a reference point provided by the DM [24]. Minimizing $s(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}})$ has the effect of optimizing problem (1) by sliding a cone along the line $\bar{\mathbf{z}} + \lambda\boldsymbol{\mu}$, where $\lambda \in \mathbb{R}$, so that a minimum (> 0) number of solutions lie in the cone [20]. Bounds of the trade-offs in the solutions obtained by (5) can be controlled by changing ρ [28].

The general formulation (6) represents achievement functions that can take preferences in other forms, not just reference points [24]. Different achievement functions differ in how $\boldsymbol{\mu}$ is set, which means they are optimizing along different directions, albeit starting from the same reference point $\bar{\mathbf{z}}$. Hence, they may lead to different solutions even if the same reference point is provided to them. For the implementation of the IOPIS algorithm, we focus on the GUESS [2] and STOM [22] scalarization functions (based on e.g., [3, 20]). For the GUESS function, $\mu_i = z_i^{nad} - \bar{z}_i$ and for the STOM function, $\mu_i = \bar{z}_i - z_i^*$. As $\mu_i > 0$ for all

$i = 1, \ldots, k$, it follows that for these two achievement functions, $z_i^* < \bar{z}_i < z_i^{nad}$ for all $i = 1, \ldots, k$. Another achievement function of note is the achievement scalarizing function (ASF) used in the reference point method [28], which is used in the experimental study section. For ASF, $\mu_i = z_i^{nad} - z_i^*$.

3 Optimization in Preference Incorporated Space

3.1 Properties of Preference Incorporated Space

Let there be a set of achievement functions $\mathbf{s} = \{s_1, \ldots, s_q\}$ with $q \geq 2$. Then we can define a PIS as the set $\{\mathbf{s}(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}) \in \mathbb{R}^q\}$, and a new MOP in the PIS as:

$$\begin{aligned} &\text{minimize } \mathbf{s}(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}) = \{s_1(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}}), \ldots, s_q(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}})\} \\ &\text{subject to } \mathbf{x} \in S. \end{aligned} \tag{7}$$

Two solutions $\mathbf{x}^1$ and $\mathbf{x}^2$ can now be compared in two spaces. As stated in Section 2.1, a solution $\mathbf{x}^1$ is said to dominate another solution $\mathbf{x}^2$ *in the objective space* if $\mathbf{f}(\mathbf{x}^1) \succ \mathbf{f}(\mathbf{x}^2)$. A solution $\mathbf{x}^1$ is said to dominate $\mathbf{x}^2$ *in the PIS* if $\mathbf{s}(\mathbf{f}(\mathbf{x}^1), \bar{\mathbf{z}}) \succ \mathbf{s}(\mathbf{f}(\mathbf{x}^2), \bar{\mathbf{z}})$. Similar to (2), we can define the solutions to problem (7), i.e., the Pareto set obtained by optimizing in the PIS as:

$$PS_{PIS} = \{x \in S \mid \nexists_{x^* \in S} \ \mathbf{s}(\mathbf{f}(\mathbf{x}^*), \bar{\mathbf{z}}) \succ \mathbf{s}(\mathbf{f}(\mathbf{x}), \bar{\mathbf{z}})\}. \tag{8}$$

We modify the desirable properties of scalarization functions as stated in [25] to reflect properties related to the PIS as:

1. Pareto optimal solutions in the PIS remain Pareto optimal in the objective space.
2. Pareto optimal solutions in the PIS follow the preference given by the DM in the objective space.
3. Any properly Pareto optimal solution of problem (1) can be discovered by changing the reference point of problem (7).

It can be shown that the first condition is true regardless of the choice or number of the achievement functions.

Theorem 2. *Let PS_{PIS} be the set of Pareto optimal solutions of problem* (7)*. Let PS_{OS} be the set of Pareto optimal solutions of problem* (1)*. Then,*

$$PS_{PIS} \subset PS_{OS}. \tag{9}$$

Proof. Suppose $\mathbf{x} \in PS_{PIS}$ but $\mathbf{x} \notin PS_{OS}$. Therefore, there exists some $\mathbf{x}^*$ such that $\mathbf{f}(\mathbf{x}^*) \succ \mathbf{f}(\mathbf{x})$. Thus, according to Theorem 1, $s^i_{\bar{\mathbf{z}}}(\mathbf{f}(\mathbf{x}^*)) < s^i_{\bar{\mathbf{z}}}(\mathbf{f}(\mathbf{x}))$ for all $i \in \{1, \ldots, q\}$. Hence, $\mathbf{s}_{\bar{\mathbf{z}}}(\mathbf{f}(\mathbf{x}^*)) \succ \mathbf{s}_{\bar{\mathbf{z}}}(\mathbf{f}(\mathbf{x}))$, which contradicts $\mathbf{x} \in PS_{PIS}$. □

The set PS_{PIS} represents the trade-offs between the values of the various achievement functions in problem (7). As the different achievement functions are different interpretations of the same preference information obtained from a

DM, the solutions in the set PS_{PIS} represent the trade-offs between those interpretations. Hence, it can be said that solutions obtained by solving problem (7) follow the preferences given by the DM. Moreover, as PS_{PIS} includes solutions which minimize individual achievement functions present in PIS, and as any properly Pareto optimal solution in the objective space can be found using the achievement functions by changing the reference points, it follows that the third condition also holds. Note that these results are valid for all order-approximating scalarization functions, and not just STOM and GUESS functions.

Solving the MOP in the PIS has a few benefits compared to solving the MOP in the objective space. Firstly, we can control the dimension of the PIS, which is equal to the number of achievement functions chosen. This means that, given some number (≥ 2) of achievement functions, we can use any multiobjective EA (or biobjective EA, as PIS can be a two dimensional space) to solve problem (7), regardless of the number of objectives in the original problem. Secondly, controlling the dimension of the optimization problem also gives us an indirect control over the number of function evaluations needed by the EA during the optimization process. This is because the number of solutions required to adequately represent the set of Pareto optimal solutions increases with increasing the dimension of the objective space (for problem (1)) or PIS (for problem (7)). Hence, choosing fewer achievement functions than k is an easy way to reduce the number of function evaluations needed by an EA to solve an optimization problem. Thirdly, by incorporating the preference information in the PIS, we gain the ability to use any non-interactive EA in an interactive fashion. This modularity enables easy use of well-tested EAs without needing to change them to enable interaction with a DM.

3.2 The IOPIS algorithm

The IOPIS algorithm takes a formulation of the optimization problem of the form (1) as input. The algorithm also takes as its input a set of achievement functions $\mathbf{s}$. The ideal point $\mathbf{z}^*$ and the nadir point $\mathbf{z}^{nad}$ of the problem are also taken as inputs. As the calculation of the nadir point can be tricky in problems with more than two objectives, approximate values of the nadir point (and ideal point) can also be used. The solutions are generated between these two points. Hence, the DM can use their expertise to give the approximate values of the points within which to search. The interactive solution process begins when these points are shown to the DM. The following four steps are repeated iteratively until the DM has received a satisfactory solution:

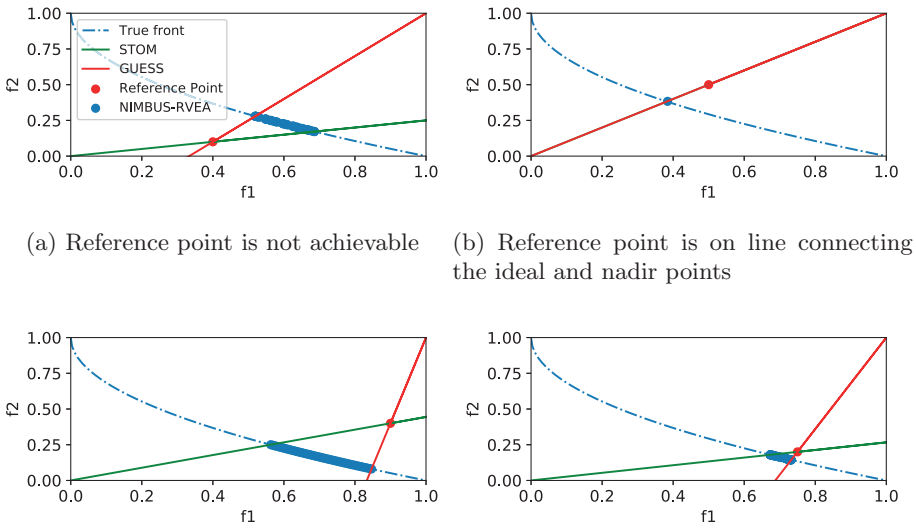
1. *Preference elicitation*: The DM is asked to give their preferences as a reference point based on the information currently available to them.
2. *Problem creation*: Using the original objectives, the known estimates of the ideal and nadir points, the reference point, and the set of achievement functions, a new optimization problem is created in the PIS, as shown in (7).
3. *Problem solution*: Solve the problem created in the previous step with an EA. If this is the first iteration of the algorithm, start the EA with a new

population, generated in a manner specific to the selected EA. In subsequent iterations, the population from the previous iteration is used as the starting population.

4. *Display solutions*: Display the solutions obtained in step 3 to the DM. The DM can indicate the maximum number of solutions to be shown at a time in step 4. If the number of solutions generated in step 3 exceeds the limit, e.g., clustering can be applied before displaying solutions.

3.3 Visual Interpretation

Even though the IOPIS algorithm is designed for optimization problems with more than two objectives, a biobjective problem is easily visualizable to demonstrate the algorithm. Here we use the ZDT1 problem [32] to study the effect of the choice of the reference point on the solutions returned by one implementation of the IOPIS algorithm. In this implementation, STOM and GUESS scalarization functions are used with NSGA-III to solve the resulting MOP in the PIS.



(a) Reference point is not achievable

(b) Reference point is on line connecting the ideal and nadir points

(c) Reference point is achievable

(d) Reference point is close to the front

Fig. 1. Solutions obtained for various reference points for the ZDT1 problem.

Four different reference points are given to the algorithm. Each subfigure in Figure 1 shows the corresponding objective vectors returned by the IOPIS algorithm. In each subfigure, the blue dashed curve represents the true Pareto front of the problem and the red point is the reference point. The green line represents the direction along which the STOM function optimizes, whereas the red line represents the direction along which the GUESS function optimizes.

1. The reference point is not achievable (Figure 1a): There is no solution that achieves values close to the reference point. Minimizing each achievement function individually returns the solution corresponding to the point of intersection of the line representing that achievement function and the Pareto front. As can be seen, the solutions returned by the algorithm include those solutions, and nondominated solutions in between.
2. Reference point is on the line joining the ideal and nadir point (Figure 1b): Due to the nature of the chosen achievement functions, only a single solution is returned by the algorithm. This is because if the reference point is on the line connecting the ideal and nadir, both achievement functions optimize along the same line. This behaviour can be changed by choosing a different set of achievement functions to form the PIS, or by shifting the reference point slightly to increase the diversity of the solutions.
3. The reference point is achievable and dominated (Figure 1c): The algorithm returns a set of solutions that satisfy the given reference point. As in the first case, optimal solutions of the individual achievement functions are included.
4. The reference point is close to the front (Figure 1d): Bringing the reference point closer to the front has the effect of reducing the spread of the solutions returned by the algorithm, hence solutions are returned in a narrower region.

The spread of the solutions is controlled by the position of the reference point. A DM who does not know very well what is realistic may provide a reference point far from the front. In such cases, the algorithm will return a diverse set of solutions (with an exception and possible resolutions discussed in point 2. above). After being provided with such solutions, the DM will have more knowledge about the trade-offs involved among the solutions, and may want to fine-tune their search in a narrow region. This is easily accomplished by providing a reference point closer to the now known region of the front. This methodology of control is similar to the one proposed in the reference point method [28].

4 Numerical Results

4.1 Experimental Setup

In this study, two versions of the interactive IOPIS algorithm were implemented. IOPIS/NSGA-III uses NSGA-III to solve the problem in the PIS, while IOPIS/RVEA uses RVEA. In both implementations, the STOM and GUESS functions are used as achievement functions to form the PIS. These algorithms were compared against a posteriori RVEA [4] and NSGA-III [5]. Interactive versions of the two a posteriori algorithms (iRVEA and iNSGA-III) were also implemented and included in this study. The details of iRVEA can be found in [12] and iNSGA-III was implemented in a similar manner. RVEA and NSGA-III were chosen for this study as they have been shown to work well in problems with $k > 2$ [4, 5]. Even though the problem in the PIS here is biobjective, the implementations of the IOPIS algorithm use RVEA and NSGA-III to ensure that only the effect of optimizing in the PIS is reflected in the results, and not the choice of the EA.

The algorithms were compared using the DTLZ{2-4} [7] and WFG{1-9} [13] problems, with 3-9 objectives each. The number of variables was kept as $10+k-1$, as recommended in [7]. For the IOPIS EAs, each component of the nadir point was randomly generated from a halfnormal distribution with the underlying normal distribution centered around 1 and having a scale of 0.15, then being scaled up by a factor equal to the true nadir point components (1 for the DTLZ problems, varying values for the WFG problems). This led to the generation of nadir points with components up to 50% worse than the true nadir point. This was done to test the performance of the IOPIS EAs in cases where only approximate values of the nadir point are available. The true ideal point was provided to the IOPIS EAs, as the calculation of it is relatively simpler.

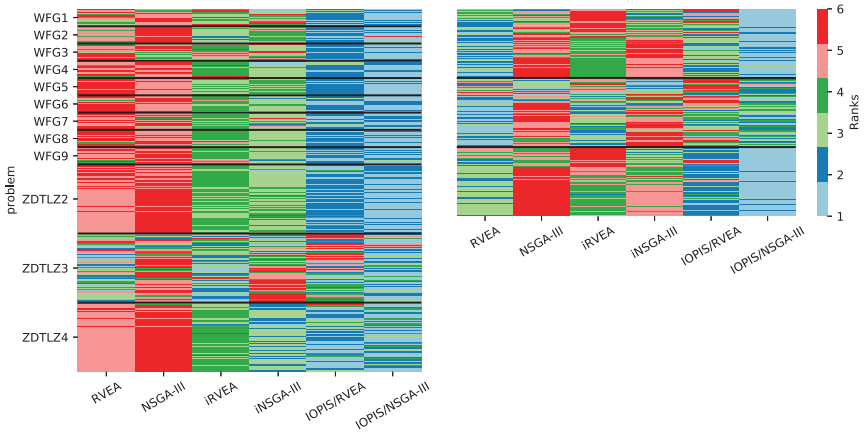
Each EA was run for four iterations. For each EA, an iteration consisted of a constant number of generations: One of {100, 150, 200, 250} for the DTLZ problems, 100 for the WFG problems (The reason for using only 100 generations per iteration for WFG problems will be discussed in the next subsection.). All other hyperparameters, such as the number of solutions or algorithm specific hyperparameters, were set to values recommended in their respective papers. In each iteration, all interactive EAs received a common reference point randomly generated in a hyperbox with the ideal and nadir points as opposing vertices. The non-interactive EAs were ran through the iterations uninterrupted. Hence 336 tests with the DTLZ problems[1] and 252 tests with the WFG problems[2] were conducted for each of the six EAs. The algorithms were compared based on the optimality and the preferability of the solutions returned at the end of each iteration, and the number of function evaluations conducted.

4.2 Experimental Results

The Pareto optimal solutions of the DTLZ2-4 problems form a hypersphere in the objective space, centered around the ideal point, which is the origin, and a radius of one. Hence, calculating the Euclidean norm of the objective vectors of the solutions returned by the EAs is a measure of optimality, with lower values of the norm being closer to the Pareto front. However, these values cannot be compared between problems, nor can they be compared for the same problem but with a different number of objectives. Instead, the median of the norm of the solutions returned at the end of each test was calculated for each of the six EAs. These values were then used to rank each EA from 1 to 6 for every test, lower ranks being given to better (lower) median norm values. A similar procedure was followed for comparing methods based on the preferability of the solutions returned by the EAs. The achievement function used in the reference point method (ASF) [28] was chosen as the metric of preferability. The median ASF values of the solutions were then used to rank the different EAs. The true nadir point was used in the calculation of the ASF values. For the tests involving the WFG 1-9 problems, ranks were only calculated based on median ASF values.

[1] = 3 (problems) * 7 (objectives) * 4 (generations per iteration) * 4 (iterations)
[2] = 9 (problems) * 7 (objectives) * 4 (iterations)

The Pareto fronts for these problems are not spherical, hence ranks based on median norm values are not relevant.



(a) Ranks based on median ASF values (b) Ranks based on median norm values

Fig. 2. Heatmaps of ranks of algorithms based on the median ASF value or median norm value of the solutions obtained.

Heatmaps of the ranks based on ASF and norm are shown in Figures 2a and 2b, respectively. A paired colormap was used in the creation of the heatmaps which gave ranks 1 and 2 a blue hue, ranks 3 and 4 a green hue, and ranks 5 and 6 a red hue. This choice brings forward a clear clustering in the rankings of the 6 EAs. As seen in Figure 2a, iRVEA and iNSGA-III tend to return more preferable solutions than their non-interactive counterparts. This behaviour is expected as RVEA and NSGA-III focus on the entire Pareto front, whereas iRVEA and iNSGA-III focus on a limited region. However, as seen in Figure 2b, the solutions returned by iRVEA were farther away from the Pareto front compared to RVEA. This is because as iRVEA has a much lower diversity of solutions compared to RVEA, which hampers the optimization process.

In both heatmaps, the PIS based EAs get ranks 1 or 2 in most tests, i.e., these algorithms returned solutions that were more preferable, and closer to the Pareto front than the other four algorithms. It should also be noted that IOPIS/NSGA-III performed better than IOPIS/RVEA in most cases. Further investigation of the PIS is required on this. The results obtained on the DTLZ3 problem are also interesting. RVEA returned solutions which were closer to the Pareto front, compared to the other methods. While the IOPIS EAs still outperformed RVEA based on the preferability of the solutions, RVEA outperformed an interactive method iNSGA-III. This is happening as iNSGA-III failed to converge to the Pareto front because of the lack of diversity of the solutions, and

got stuck on one of the local fronts of the problem. While there was a correlation between the problem type and the performance of the methods (IOPIS EAs got ranks one or two more often in the WFG problems compared to the DTLZ problems), there was no correlation between the performance of the method and the number of objectives. In the case of the DTLZ problems, the performance was also not dependent on the number of generations per iteration, i.e., there was no improvement in the results after a hundred generations (per iteration). This is why the number of generations per iterations was fixed to 100 for the tests involving the WFG problems.

The final metric of comparison is the number of function evaluations conducted. Given a constant number of generations, the number of function evaluations is linearly correlated with the population size, which is equal to the number of RVs in the EAs considered in this paper. For RVEA, NSGA-III, iRVEA and iNSGA-III, the RVs (and hence the number of function evaluations) increase exponentially with an increasing number of objectives. As the IOPIS algorithms operate in the low-dimensional PIS, the number of reference vectors, and hence the number of function evaluations, is independent of the number of objectives, and significantly lower than that for the other algorithms considered in the study. It should also be noted that for all of the tests, only an approximate nadir point was provided to the IOPIS EAs, and yet the IOPIS EAs obtain better results than the current state of the art algorithms.

5 Conclusions

A new space PIS, where preferences are incorporated, was proposed as a new paradigm of solving MOPs interactively. This new space makes the creation of interactive EAs very modular, as the algorithm only needs to modify the problem to enable interactivity, rather than the EA itself. As examples, this enabled easy creation of the IOPIS/NSGA-III and IOPIS/RVEA implementations.

The results obtained in the numerical experiments were very promising. The new interactive EAs outperformed standalone NSGA-III and RVEA, as well as their interactive versions. The solutions obtained by the IOPIS EAs were closer to the Pareto optimal front, more preferable based on the reference point and spent less computational resources in the form of function evaluations. Further study of the landscape of the PIS is needed. The effect of choosing different achievement functions, their implications on the interaction mechanism by a DM and the solutions returned by the algorithm also needs to be studied.

Acknowledgements. This research was supported by the Academy of Finland (grant numbers 322221 and 311877). The research is related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization, jyu.fi/demo) of the University of Jyväskylä.

References

1. Bechikh, S., Kessentini, M., Said, L.B., Ghédira, K.: Chapter four - Preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. In: Hurson, A.R. (ed.) Advances in Computers, vol. 98, pp. 141–207. Elsevier (2015)
2. Buchanan, J.T.: A naïve approach for solving MCDM problems: The GUESS method. Journal of the Operational Research Society **48**(2), 202–206 (1997)
3. Buchanan, J., Gardiner, L.: A comparison of two reference point methods in multiple objective mathematical programming. European Journal of Operational Research **149**(1), 17—34 (2003)
4. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation **20**(5), 773–791 (2016)
5. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation **18**(4), 577–601 (2014)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
7. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002). pp. 825–830. IEEE (2002)
8. Deb, K., Miettinen, K.: Nadir point estimation using evolutionary approaches: Better accuracy and computational speed through focused search. In: Ehrgott, M., Naujoks, B., Stewart, T.J., Wallenius, J. (eds.) Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, Proceedings. pp. 339–354. Springer, Berlin, Heidelberg (2010)
9. Deb, K., Saxena, D.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: Proceedings of the World Congress on Computational Intelligence (WCCI-2006). pp. 3352–3360 (2006)
10. Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. In: GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. pp. 635—-642. ACM, New York (2006)
11. Eskelinen, P., Miettinen, K., Klamroth, K., Hakanen, J.: Pareto Navigator for interactive nonlinear multiobjective optimization. OR Spectrum **32**(1), 211–227 (2010)
12. Hakanen, J., Chugh, T., Sindhya, K., Jin, Y., Miettinen, K.: Connections of reference vectors and different types of preference information in interactive multiobjective evolutionary algorithms. In: Proceeding of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8 (2016)
13. Huband, S., Barone, L., While, L., Hingston, P.: A scalable multi-objective test problem toolkit. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) Evolutionary Multi-Criterion Optimization, Third International Conference, Proceedings. pp. 280–295. Springer, Berlin. Heidelberg (2005)
14. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). pp. 2419–2426 (2008)

15. Li, K., Chen, R., Min, G., Yao, X.: Integration of preferences in decomposition multiobjective optimization. IEEE Transactions on Cybernetics **48**(12), 3359–3370 (2018)
16. Luque, M., Ruiz, F., Miettinen, K.: Global formulation for interactive multiobjective optimization. OR Spectrum **33**(1), 27–48 (2011)
17. Meignan, D., Knust, S., Frayret, J.M., Pesant, G., Gaud, N.: A review and taxonomy of interactive optimization methods in operations research. ACM Transactions on Interactive Intelligent Systems **5**(3), 1–43 (2015)
18. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)
19. Miettinen, K., Hakanen, J., Podkopaev, D.: Interactive nonlinear multiobjective optimization methods. In: Greco, S., Ehrgott, M., Figueira, J.R. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys, pp. 927–976. Springer, New York (2016)
20. Miettinen, K., Mäkelä, M.M.: On scalarizing functions in multiobjective optimization. OR Spectrum **24**(2), 193–213 (2002)
21. Miettinen, K., Mäkelä, M.M.: Synchronous approach in interactive multiobjective optimization. European Journal of Operational Research **170**(3), 909–922 (2006)
22. Nakayama, H., Sawaragi, Y.: Satisficing trade-off method for multiobjective programming. In: Grauer, M., Wierzbicki, A.P. (eds.) Interactive Decision Analysis. pp. 113–122. Springer, Berlin, Heidelberg (1984)
23. Ruiz, A.B., Luque, M., Miettinen, K., Saborido, R.: An interactive evolutionary multiobjective optimization method: Interactive WASF-GA. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) Evolutionary Multi-Criterion Optimization, 8th International Conference, Proceedings. pp. 249–263. Springer, Cham (2015)
24. Ruiz, F., Luque, M., Miettinen, K.: Improving the computational efficiency in a global formulation (GLIDE) for interactive multiobjective optimization. Annals of Operations Research **197**(1), 47–70 (2012)
25. Sawaragi, Y., Nakayama, H., Tanino, T.: Theory of Multiobjective Optimization. Elsevier (1985)
26. Thiele, L., Miettinen, K., Korhonen, P.J., Molina, J.: A preference-based evolutionary algorithm for multi-objective optimization. Evolutionary Computation **17**(3), 411–436 (2009)
27. Vesikar, Y., Deb, K., Blank, J.: Reference point based NSGA-III for preferred solutions. In: Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1587–1594 (2018)
28. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) Multiple criteria decision making theory and application, pp. 468–486. Springer (1980)
29. Wierzbicki, A.P.: A mathematical basis for satisficing decision making. Mathematical Modelling **3**(5), 391–405 (1982)
30. Xin, B., Chen, L., Chen, J., Ishibuchi, H., Hirota, K., Liu, B.: Interactive multiobjective optimization: A review of the state-of-the-art. IEEE Access **6**, 41256–41279 (2018)
31. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**(6), 712–731 (2007)
32. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8**(2), 173–195 (2000)

PIII

SCORE BAND VISUALIZATIONS: SUPPORTING DECISION MAKERS IN COMPARING HIGH-DIMENSIONAL OBJECTIVE VECTORS IN MULTIOBJECTIVE OPTIMIZATION

by

Bhupinder Singh Saini, Kaisa Miettinen, Kathrin Klamroth, Ralph E. Steuer, Kerstin Dachert

Submitted to a journal.

SCORE Band Visualizations: Supporting Decision Makers in Comparing High-Dimensional Objective Vectors in Multiobjective Optimization

Bhupinder Singh Saini[1,*], Kaisa Miettinen[1], Kathrin Klamroth[2], Ralph E. Steuer[3], and Kerstin Dächert[4]

[1]University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland
[2]University of Wuppertal, School of Mathematics and Natural Sciences, Gaußstr. 20, 42119 Wuppertal, Germany
[3]Department of Finance, University of Georgia, Athens GA 30502, USA
[4]Dresden University of Applied Sciences, P.O. Box 120701, D-01008 Dresden, Germany
[*]Corresponding author: Bhupinder Singh Saini, bhupinder.s.saini@jyu.fi

August 31, 2022

Abstract

Clearly arranged visualizations are needed in optimization problems with a large number of objective functions and, often simultaneously, a large number of solution alternatives to support the decision making processes. This paper contributes to the visualization of such optimization problems by proposing the simultaneous clustering of solutions and correlated objectives to re-organize parallel coordinate plots in a systematic and meaningful way.

Parallel coordinate plots are a widely used visualization technique to represent different solutions. We propose a novel visualization technique called SCORE bands to be used with parallel coordinate plots to support the decision maker in identifying patterns in the solutions and correlations among the objectives. The decision maker can, for example, get an overall impression of the trade-offs among the solutions and zoom in and out as desired. The decision maker also gets information about the correlations among the objectives from their ordering and from their distances. The SCORE bands decrease the amount of information to be digested at a time. Additionally, the interactive nature of the visualizations allows

the decision maker to affect the views to gain insight into phenomena represented in the set of solutions. We demonstrate the added value of SCORE bands with different examples including a real-life problem with nine objectives.

Keywords: Multiple objective programming; Parallel coordinate plots; Correlated objectives; Interactive visualization; Pareto optimality

1 Introduction

The aim of multiobjective optimization methods is to support a domain expert, to be referred to as a decision maker (DM), in finding the best balance among conflicting objective functions. Many methods generate so-called Pareto optimal solutions, where a Pareto optimal solution is one in which no objective can be improved without impairing at least one of the others. Different methods generate varying amounts of Pareto optimal solution candidates to compare; but the task of comparison gets more demanding when the number of objective functions and the number of solution candidates increases.

Carefully selected visualizations can help a DM in gaining insight in different trade-offs among solution candidates. Means of visualization for multiobjective optimization purposes are surveyed, e.g., in Gettinger et al., 2013; Korhonen and Wallenius, 2008; Lotov and Miettinen, 2008; Miettinen, 2014; Woodruff et al., 2013. Examples of popular visualization techniques are parallel coordinate plots (also known as value paths (Geoffrion et al., 1972)), spider web charts, scatterplot matrices, petal diagrams, star coordinate plots and glyphs. Further techniques include heatmaps (Hettenhausen et al., 2010), knowCube (Trinkaus & Hanne, 2005), interactive decision maps (Lotov et al., 2004), the prosection method (Tusar & Filipic, 2015), PaletteViZ (Talukder & Deb, 2020), and 3d-radvis (Ibrahim et al., 2016).

As discussed, e.g., in Fonseca et al., 2015, visualizations can be applied for various purposes in multiobjective optimization ranging from following the progress of the solution process to visualizing uncertainty and to identifying information to be visualized. Here we focus on visualizing solution candidates in the objective space, that is, vectors consisting of objective function values. Recent developments in visualization methods include open-source building blocks for implementing parallel coordinate plots (Raseman et al., 2019) and considerations of visualizations from the needs of specific application domains, e.g., (Haara et al., 2018; Liu et al., 2018; Meignan et al., 2015). In many studies, parallel coordinate plots have been found useful in visualizing solution candidates supported by, e.g, clustering (Cajot et al., 2019; Yang et al., 2020). In parallel coordinate plots, objective functions are typically represented by vertical axes and solution candidates are represented by polylines. Naturally, the order of the axes affects the interpretability since trade-offs in the objectives that are next to each other are easier to be inspected. Even though Ankerst et al., 1998 is not about multiobjective optimization, the measures of similarity proposed could be used to order objectives in a parallel coordinate plot. An approach for

deriving the order of the axes using Spearman's rank correlation was proposed in Zhen et al., 2017. In both Ankerst et al., 1998 and Zhen et al., 2017, the axes are placed equidistant to each other.

In this paper, we propose means for supporting a DM in understanding the information contained in a set of Pareto optimal objective vectors that consist of objective function values of the solution candidates. Our aim is to show trends in both objectives and solution candidates so that it is easier to digest major insights. Roughly speaking, we visually cluster both objectives and candidates. We propose to use Pearson correlation coefficients of all objective pairs to calculate the order of the objectives. Moreover, we visualize the correlation information by changing the distance between neighboring axes based on the value of the Pearson correlation coefficient of the corresponding objectives. Thus, we not only determine the order in which the objectives should be displayed but illustrate correlation information visually. This is particularly helpful when the number of objective functions is above three. While the ideas behind this research are reported in Dächert et al., 2020 (applying different clustering tools), this is the full development of the initiatives outlined in that piece (including the idea about modifying the distances between axes in a parallel coordinate plot, which does not appear elsewhere other than in Dächert et al., 2020).

When the number of solution candidates is high, one can filter out undesired solutions as e.g., in knowCube (Trinkaus & Hanne, 2005). But if one wants to understand better what kind of trade-offs are represented in the data available, as we do in this paper, clustering can be applied to first show the bigger trends among the objectives, and then the DM can be given the ability to zoom in on clusters of special interest to examine individual solutions more closely. In support of this kind of an analysis, we propose use of the SCORE band visualizations.

Overall, our novel contribution to visualize Pareto optimal objective vectors with modified parallel coordinate plots is three-fold: ordering the objectives, reflecting correlation among the objectives in the location of the objectives (with different distances) and applying SCORE bands to visualize solution candidates in a visually pleasing way, as seen in the graphical abstract. We do not care how the set of solutions to be visualized has been generated as long as they do not dominate each other. The new visualizations can support the DM during a solution process to get an overall understanding or focus on solutions of interest, provide preferences as well as identify the final, most preferred solution. Thus, the proposed visualizations can be used with different multiobjective optimization methods.

In general, the data to be visualized does not need to come from a multiobjective optimization problem. Instead, it can correspond to a multiple criteria decision making problem. In principle, the visualization technique can even be used for data analysis.

The rest of this paper is structured as follows. In Section 2, we introduce main concepts and notations used, and in Section 3, we propose our new ways of visualizing sets of Pareto optimal solutions, including SCORE bands. We give examples in Section 4 as well as discuss different ways of utilizing the new

visualizations. Finally, we conclude in Section 5.

2 Concepts and Notation

2.1 Multiobjective Optimization and Pareto Optimality

We consider *multiobjective optimization problems*

$$\min\{\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), ..., f_k(\boldsymbol{x})) : \boldsymbol{x} \in X\} \tag{MOP}$$

with $k \geq 2$ real-valued objective functions $f_i : X \to \mathbb{R}$, $i = 1, \ldots, k$ and a feasible set X. The objective functions may represent, for example, economical and ecological goals, or the cost and the quality of a decision. Using the fact that maximization problems can be reformulated as equivalent minimization problems, we assume throughout this paper that all objective functions are to be minimized. Accordingly, we assume that a rational DM prefers smaller objective values over larger objective values in all objectives. This is reflected in the concept of *Pareto optimal* decisions for which none of the objective function values can be improved without deterioration of at least one other objective function value. Pareto optimality can be best detected by comparing *objective vectors* of feasible decisions in the k-dimensional *objective space* $\mathbb{R}^k$: An objective vector $\boldsymbol{z}^1 = \boldsymbol{f}(\boldsymbol{x}^1)$ *dominates* $\boldsymbol{z}^2 = \boldsymbol{f}(\boldsymbol{x}^2)$ in $\mathbb{R}^k$ if and only if $z_i^1 \leq z_i^2$ for all $i = 1, \ldots, k$ and $\boldsymbol{z}^1 \neq \boldsymbol{z}^2$. Then a feasible decision $\boldsymbol{x} \in X$ is Pareto optimal if and only if there is no other feasible decision $\bar{\boldsymbol{x}} \in X$ such that $\boldsymbol{f}(\bar{\boldsymbol{x}})$ dominates $\boldsymbol{f}(\boldsymbol{x})$. Corresponding to the set of all Pareto optimal decisions, i.e., the *Pareto set*, we have their respective images in the objective space constituting a *Pareto front*, i.e., the set of all Pareto optimal objective vectors of problem (MOP). In this paper, by solutions we refer to objective vectors.

To support the DM in the selection of a most preferred solution, a clearly arranged graphic presentation of a representative subset of the Pareto front is crucial. This is particularly true when the number of objective functions increases, i.e., when k is (considerably) larger than 2 or 3. Indeed, while for biobjective problems the Pareto front can be visualized in a 2-dimensional plot that immediately shows the *trade-offs* between the two objectives, this is no longer true for higher-dimensional problems. Already in a three-objective case, a direct visualization of the Pareto front in the objective space is difficult, and it is generally not useful at all in the case of more than 3 objectives. At the same time, optimization problems with an increasing number of objective functions are becoming more and more relevant and popular in practical applications, and hence there is a growing need for efficient and meaningful presentations of interesting (Pareto optimal) solution candidates.

We focus on visualizations of a finite set of Pareto optimal objective vectors or objective vectors that do not dominate each other (i.e., nondominated vectors) on *parallel coordinate plots* (see, e.g., Geoffrion et al., 1972; Inselberg and Dimsdale, 1987; Wegman, 1990). Here, each objective function is associated with a vertical bar, also called an axis, that represents the range of possible

function values, while each solution is represented by its respective values on these bars which are connected by a *value path*. See Figure 2 for an illustration.

2.2 Structured Datasets with Partially Correlated Objective Functions

For purposes of illustration, we consider two instances of linear problems given by

$$\min\{\boldsymbol{f}(\boldsymbol{x}) = C\boldsymbol{x} \,:\, A\boldsymbol{x} \leq \boldsymbol{b},\, \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{u}\}, \tag{AD}$$

where $C \in \mathbb{R}^{k\times n}$ is the objective matrix, $A \in \mathbb{R}^{m\times n}$ the constraint matrix, $\boldsymbol{b} \in \mathbb{R}^m$ the right-hand-side vector, and $\boldsymbol{u} \in \mathbb{R}^n$ a vector of upper bounds on the variable values. The problems are designed so that they naturally induce clusters of similar objective functions and of similar solutions. The first instance of problem (AD) has three variables and six objective and the coefficients are chosen as specified in Table 1. The corresponding feasible set as well as objective vectors are illustrated in Figure 1.

	x_1	x_2	x_3		
C	-5	0.5	1	min	
	-5	1	0.5	min	
	2.5	-25	5	min	
	1	-5	0.5	min	
	12.5	25	-125	min	
	1	0.5	-5	min	
s.t.	1	1	1	$\leq$	1.0
	2	1	0	$\leq$	1.7
u	0.8	0.8	0.8		
					all vars ≥ 0

Table 1: Objective and constraint coefficients of the 3-variable, 6-objective instance of (AD)

The first instance has 11 Pareto optimal extreme points $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{11}\}$. In Figure 1, they are indicated by dots. We denote the corresponding objective vectors by (AD1). The vectors $-\boldsymbol{c}^i$, $i = 1, \ldots, 6$, in the graph illustrate the negative gradient directions of the objectives, indicating the direction of optimization. Note that while their directions are accurate, their lengths are merely suggestive because of the differences in scale.

As seen, the six objectives are clustered into three batches of two each, with $-\boldsymbol{c}^1$ and $-\boldsymbol{c}^2$ pointing almost down the x_1-axis, $-\boldsymbol{c}^3$ and $-\boldsymbol{c}^4$ pointing almost up the x_2-axis, and $-\boldsymbol{c}^5$ and $-\boldsymbol{c}^6$ pointing along the x_3-axis. This suggests that an appropriate clustering of the objective functions would be in the groups $\{f_1, f_2\}$,

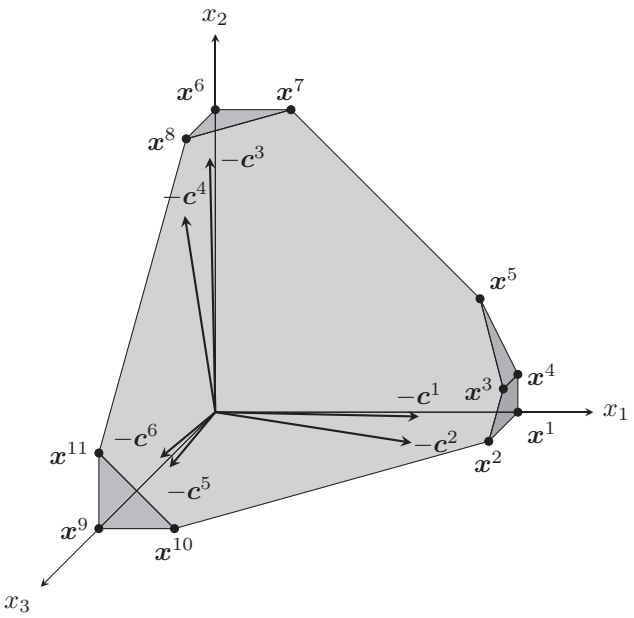


Figure 1: 3-variable instance (AD1) whose 6 objectives are clustered into 3 batches of 2 each.

$\{f_3, f_4\}$, and $\{f_5, f_6\}$, while the 11 Pareto optimal extreme points should be clustered according to geometrical closeness on the polyhedral feasible set, i.e., in the groups $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^5\}$, $\{\boldsymbol{x}^6, \boldsymbol{x}^7, \boldsymbol{x}^8\}$ and $\{\boldsymbol{x}^9, \boldsymbol{x}^{10}, \boldsymbol{x}^{11}\}$.

Our second instance of problem (AD) has nine objectives. We consider it later in the paper in connection with its dataset (AD2) of 141 Pareto optimal objective vectors. We share all datasets used in this paper at https://zenodo.org/record/5944515.

3 SCORE Bands: Solution clustering and correlated objective visualization via bands

In this section, we describe the algorithm by which SCORE bands and their visualizations are constructed so as to enable the re-imagination of the concept of a parallel coordinate plot of this paper. The purpose is to support decision making in multiobjective optimization with visualizations of sets of Pareto optimal (or nondominated) objective vectors in order to highlight key information contained in them while minimizing clutter that might otherwise distract from the decision making process. Bear in mind that while the goal of a visualization is to tailor to the needs of a given DM, the algorithm may have to process many Pareto optimal objective vectors. Possessing parameters designed to be easy to operate (to adapt to the needs of the problem at hand and the DM by e.g. an

analyst supporting a DM), the algorithm consists of four steps:

1. **Solution clustering**: Find clusters in the solutions. This information helps the DM understand the distribution of solutions in the objective space.

2. **Axis ordering**: Calculate the optimal ordering for the objective axes of the parallel coordinate plot visualization. This helps visualizing information about the relationships between different objectives, such as trade-offs or correlations.

3. **Axis placement**: Expand or contract the space between the objective axes to highlight or suppress the relationship between neighboring objectives in the parallel coordinate plot. This usage of the inter-axes space can help the DM focus on important relationships among the solutions or objectives.

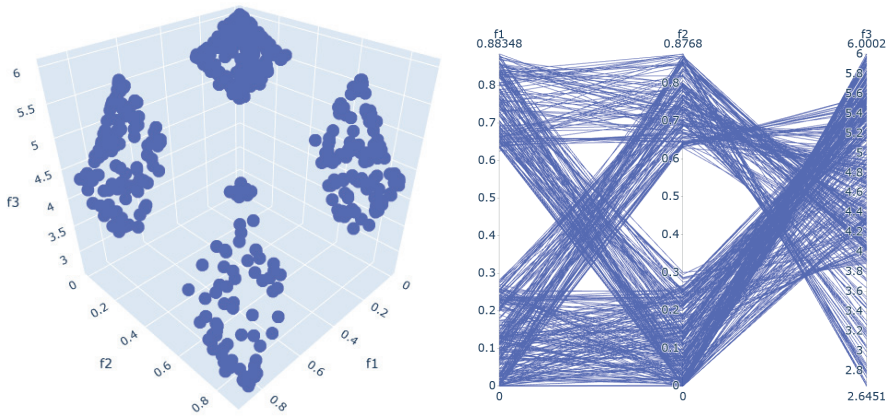
4. **Solution visualization**: Visualize the information extracted in the previous three steps in an effective, visually accessible, and pleasing manner.

Other simple visualizations can be presented to a DM in parallel to the SCORE bands. They can provide additional insight into the problem and make it easier to understand the SCORE band visualization.

In Subsections 3.1 through 3.5, we describe the various components of the aforementioned algorithm. We also demonstrate the advantage of using the components individually with a dataset of 1036 objective vectors generated for the DTLZ7 problem (Deb et al., 2005) with 3 objectives. The dataset, to be referred to as (3-DTLZ7) is visualized in Figure 2a as a 3-D scatter plot and in Figure 2b as a parallel coordinate plot. Note that the SCORE band algorithm can be used to visualize objective vectors related to problems with, at least theoretically, any number of objectives, and we only use a three-objective problem here to describe the algorithm. We compare the results against some standard visualization techniques.

3.1 Solution clustering

Clustering can be a very useful tool for DMs. Clustering can highlight patterns in the objective space by identifying groups of solutions that are close to each other (i.e., in a cluster) and far from solutions in other groups. In Figure 2a, the 3D scatter plot clearly shows that the solutions are spread among four clusters. However, as can be seen in Figure 2b, the cluster information is harder to appreciate in the parallel coordinate plot. While the four clusters are clearly distinguishable with regard to the first two axes, they are not nearly as distinguishable with regard to the third. However, the problem is alleviated by giving colors to the individual solutions according to their cluster. Figure 3 demonstrates the effectiveness in the parallel coordinate plot shown. It now becomes trivial to distinguish the four clusters, even in the third objective. The spread of each cluster is visible along each axis.



(a) 3D scatter plot (b) Parallel coordinate plot

Figure 2: (3-DTLZ7) set of nondominated objective vectors visualized using two techniques.

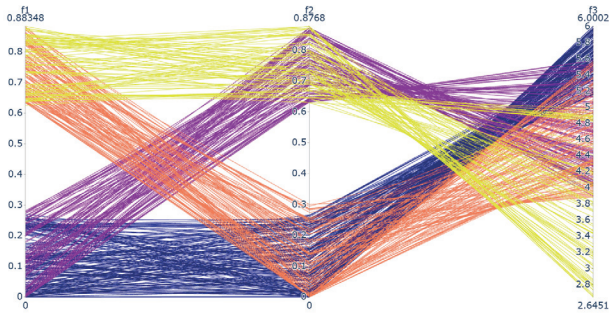


Figure 3: (3-DTLZ7) set of nondominated objective vectors plotted in a parallel coordinate plot and colored according to clustering information.

3.2 Axis ordering

The order of the objectives in a parallel coordinate plot can make a significant difference in interpretation. An axis in a parallel coordinate plot can have at most two neighboring axes. Hence, the relationship of an objective to its neighboring objectives is very prominent visually. Crossing over of the value paths of individual solutions between neighboring axes signifies a negative correlation. If instead, the value paths are primarily parallel to each other, a DM can conclude the two neighboring objectives are highly positively correlated. Finally, a chaotic tangle of value paths signifies a lack of correlation between the neighboring objectives. In static visualizations, i.e., plots that a user cannot interact with or change, this information comes at the cost of a lack of information about non-neighboring objective pairs. Dynamic visualizations can help the DMs solve

this problem by allowing manual reordering of the axes in real-time, but it may be time-consuming to find the most informative order. However, static visualizations are still necessary for certain media where dynamic visualizations are impossible, such as in print. Hence, methods to most informatively order the axes in a parallel coordinate plot to highlight information relevant to a DM are still required.

One such method to derive a good order of the objectives (represented by a permutation π) is to solve a travelling salesperson problem $TSP(c_{i,j})$, where $c_{i,j} \in \mathbb{R}$ is a measure of the distance between two objectives. The ordering is given by a permutation π of the set $\{1, 2, ..., k\}$, where k is the number of objectives, such that f_{π_i} is the i^{th} objective to be placed as an axis in the parallel coordinate plot. We provide the user with two options for the distance metric. In the first option (referred to as *Metric 1* in the following), $c_{i,j} = -\rho(f_i, f_j)$, where $\rho(f_i, f_j)$ is the Pearson correlation coefficient between f_i and f_j. By using this metric, objectives that are positively correlated are placed closer to each other. Alternatively, the user can choose to use the absolute value of the Pearson correlation coefficients instead: $c_{i,j} = -|\rho(f_i, f_j)|$. This second option is referred to as *Metric 2* in the following. Metric 2 highlights both positive and negative correlations.

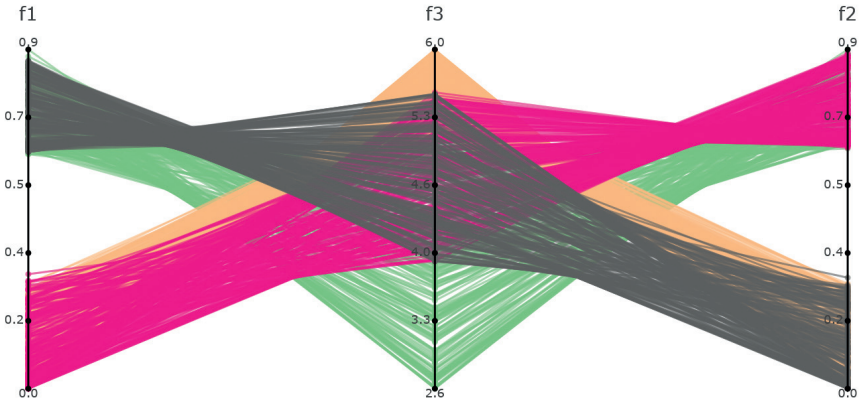


Figure 4: (3-DTLZ7) nondominated objective vectors plotted in a parallel coordinate plot, colored according to the clustering information, and with axes ordered according to Metric 2, i.e., the absolute values of the Pearson correlation coefficients.

Figure 4 shows the effectiveness of using the axes ordering method described above on the (3-DTLZ7) dataset. The second metric, which highlights both positive and negative correlations, was used to generate the figure. One can immediately notice the visual symmetry in the figure across the f_3 axis. The symmetry corresponds to a similar symmetry seen in the 3D scatter plot visualization (Figure 2a) around the f_3 axis. On the other hand, the standard parallel coordinate plot (Figure 2b) obscures this symmetry to some extent.

In static visualizations, the axes ordering method can help the DM to gain insight into the problem swiftly. In dynamic visualizations, the method can provide a default first view, which can then be altered by a DM interactively as desired.

3.3 Axis placement

A standard parallel coordinate plot dedicates an equal amount of space to each objective axis pair. However, in reality, the information that a DM can gather from inter-axis space can vary significantly between different objective pairs. For example, some objective pairs may be highly correlated, whereas others may be hardly correlated at all. Thus, we can encode relevant information by altering the space. For example, by varying the relative distances between the objective axes, we can visually show objective clusters. Objectives that behave similarly are placed closer to each other, whereas objectives that behave dissimilarly are placed farther apart.

We provide two different methods for calculating relative distances between neighboring axes ($\text{dist}_i = \text{dist}(f_{\pi_i}, f_{\pi_{i+1}})$):

$$\begin{aligned} &\text{Method 1:} \quad \text{dist}_i = 1 - \rho(f_{\pi_i}, f_{\pi_{i+1}}) + \delta \quad \text{for all } i = 1, \ldots, k-1 \\ &\text{Method 2:} \quad \text{dist}_i = \frac{1}{|\rho(f_{\pi_i}, f_{\pi_{i+1}})|} + \delta \quad \text{for all } i = 1, \ldots, k-1, \end{aligned}$$

where δ is a user-provided distance parameter which increases the minimum distance between the axes. Based on our experiments, we recommend Method 1 to be used with Metric 1 (to calculate the axes order), and Method 2 to be used with Metric 2. These relative distances can be multiplied by a scaling factor to fit a desired width, for example, the width of a monitor or a page. Note that due to the symmetrical nature of the considered DTLZ7 problem, this aspect of the SCORE band visualizations cannot be recognized in Figure 4. However, we present more problems in the later sections of the paper which demonstrate the utility of varying relative distances between axes.

3.4 Solution visualization

As mentioned, parallel coordinate plot visualizations tend to grow complicated and cluttered with an increasing number of objectives and solutions. As the number of objectives increases, the individual solution traces cross-over more often due to the trade-offs among the different solutions. On the other hand, adding more solutions to the plot increases the complexity by simply increasing the density of information in the visualization. Together, this can result in visualizations that are difficult to understand, even with the helpful features described in the previous subsections.

One way to resolve this issue is to plot simplified abstractions rather than individual solutions. For example, instead of plotting all solutions as individual traces, the clusters (as identified in Subsection 3.1) can be plotted as bands as

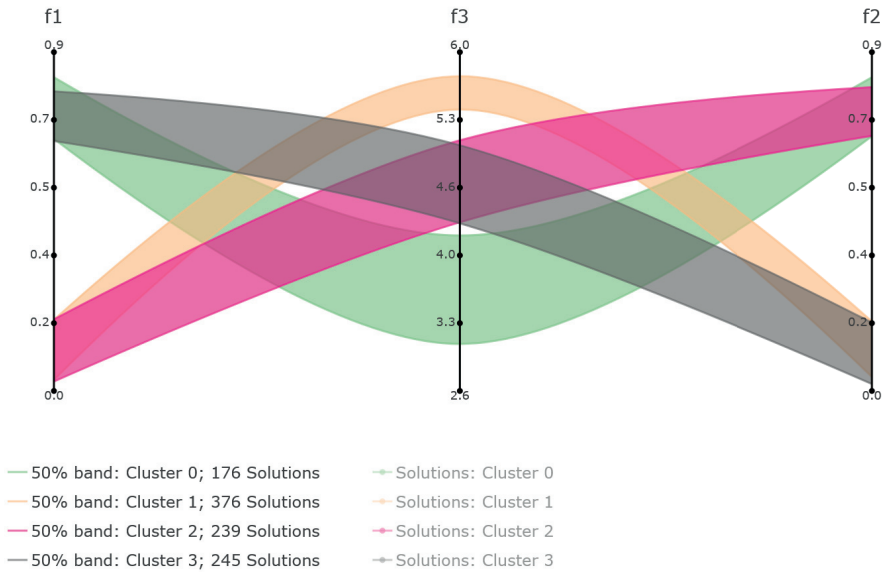


Figure 5: Nondominated objective vectors (3-DTLZ7) plotted as a SCORE band visualization.

the basic unit of visualization. In Figure 5, we showcase this idea by plotting the (3-DTLZ7) dataset. We call the result a SCORE band visualization. Each band exemplifies the pattern of the trade-offs followed by the solutions of the corresponding cluster while keeping the visualization simple. The width of the band at any axis represents the spread of corresponding objective values achieved by the solutions belonging to the cluster. The height, width, and shape of the bands can be calculated in various ways. We propose that the center of a band (on each axis) be placed on the median value achieved by solutions belonging to that cluster.

A statistical measure of spread, such as standard deviation, interquartile range, or confidence interval, should be used to determine the width of the band along each axis. We use interquartile range in Figure 5, referred to as "50% bands" in the legend. Once we have the height and width at each axis, we draw the band by interpolating between the axes. A traditional parallel coordinate plot does this by linear interpolation, leading to piece-wise linear traces for each solution. However, we have found that using spline interpolation leads to a more aesthetically pleasing visualization without extraneous information. We color the bands according to clustering information and make them translucent to make it easy to distinguish between the different clusters and follow their patterns.

Note that the SCORE band visualization is meant to be a first or default visualization to be used in the decision making process. The bands make it easy for a DM to identify patterns in the solutions and arrive at a region of inter-

est. Based on this information, the DM can, for example, give their preferences for the next step of an interactive decision making process. Alternatively, the DM may choose to investigate further in one's region of interest by selectively visualizing the solutions belonging to clusters of interest or eliminate clusters from consideration. We have implemented the SCORE band visualization as an open-source Python package and will make the code available. The package creates interactive visualizations which support both SCORE bands and traditional parallel coordinate plots. A DM can show or hide various bands or solution clusters by interacting with the plot.

3.5 Supporting visualizations

Supporting visualizations, which provide a DM with relevant information in simple plots, can be helpful in a decision making process. Often, it may be impossible or undesirable to incorporate such information into the SCORE band visualization. The types of possible useful supporting visualizations depend on the optimization problem. For example, in problems involving geographic locations, plotting some characteristics on a map may be helpful.

We identify two generic visualizations that compliment the SCORE band visualizations, which can be helpful in any problem. The first is a heatmap of the Pearson correlation coefficient values between different objectives. Apart from relaying the correlation information, it can help explain the order of objectives in the SCORE band visualization. It can also enable the DM to manually decide the order of objectives in a SCORE band visualization or parallel coordinate plot. The second supporting visualization is generated by applying dimensionality reduction techniques such as t-SNE (van der Maaten & Hinton, 2008) on the solutions and plotting the results in a 2-D scatter plot. Depending on the choice of the dimensionality reduction technique, such visualization may help a DM understand the local structure of various parts of the Pareto front, the presence or absence of clusters, or connectivity between various parts of the front.

4 Implementation details

We have implemented the SCORE band visualization and a graphical user interface (GUI) tool to help create the plots from data as a part of the open-source interactive optimization framework DESDEO (Misitano et al., 2021). We use the `Plotly` package (Plotly Technologies Inc., 2015) to implement the SCORE band visualization and the related `Dash` (Hossain, 2019) package to create a web-based GUI. There are various customization options available to a user to tailor the SCORE band visualization to different needs. A user can import data into the GUI in the form of a CSV file. The visualization can then be created without further input from the user. Alternatively, the GUI provides a form, which the user can use to change the various parameters of the SCORE band visualization. The GUI can be seen in Figure 6. The options available to the

user are:

1. CSV file: It should only contain the objective vectors, such that each column represents a different objective. The first row should contain the objective names.

2. Solution clustering: The different clustering algorithms supported are: DBSCAN (default), Bayesian Gaussian mixture models, K-Means, spectral clustering, Ward hierarchical clustering, and agglomerative clustering. Providing the number of clusters is necessary for all algorithms except DBSCAN and Bayesian Gaussian mixture models. We use the `scikit-learn` package to train these models (Pedregosa et al., 2011).

3. Axes ordering: The user can choose whether to use the absolute value of Pearson correlation coefficient as a distance metric or not. By default, Metric 1 is used.

4. Axis placement: The user can choose the distance function to be used (Method 1 is used by default) and the distance parameter δ (default value $= 0.4$).

5. Solution visualization: The user can choose whether to display individual solutions, cluster medians, and SCORE bands in the plot. The user can also disable each of the three options for each cluster independently. By default, only the bands are visualized.

6. Supporting visualizations: The user can choose from the following dimensionality reduction techniques: locally linear embedding (LLE), LTSA LLE, Hessian LLE, modified LLE, isomap embedding, multidimensional scaling, spectral embedding, and t-distributed stochastic neighbor embedding (default). We use the implmentation of these algorithms from the `scikit-learn` package.

Additionally, we use the `NumPy` (Harris et al., 2020) and `Pandas` (Reback et al., 2020) Python packages for data handling. We use the `SciPy` Python package (Virtanen et al., 2020) to extract statistical information (such as Pearson correlation coefficients) from the data. Finally, we use the `tsp_solver2` package to solve the TSP problem using a greedy algorithm (Shintyakov, 2020).

5 Case Studies

In this section, we demonstrate the usage of the SCORE band visualizations with a variety of datasets. These include artificially generated datasets, datasets obtained from multiobjective optimization benchmark problems, and real-life multiobjective optimization problems. We showcase how the various parameters of the SCORE band visualizations can be changed to highlight different aspects of the explored datasets. We also show how supporting visualizations can help users understand the SCORE band visualizations and the data.

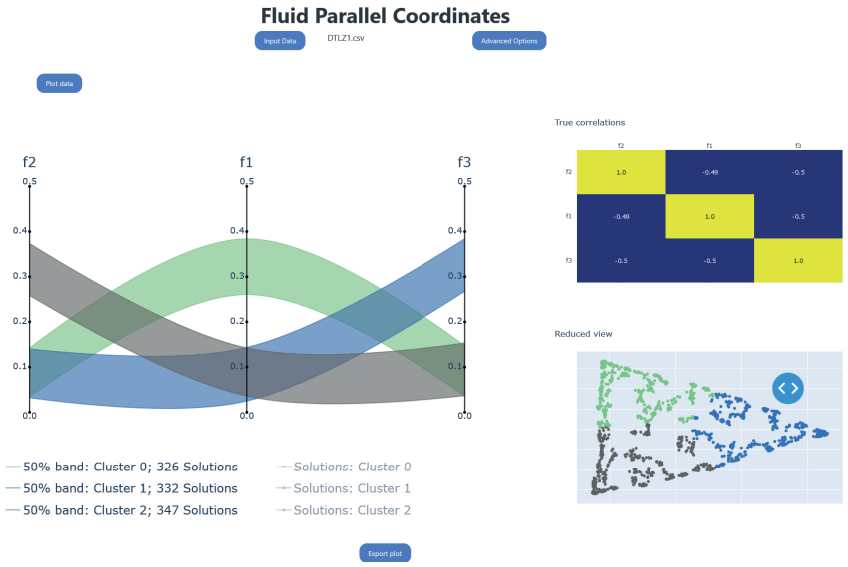


Figure 6: A GUI application to analyze datasets using SCORE bands. The GUI provides support for uploading datasets in a CSV format, allows the user to customize the visualization parameters, displays the SCORE bands as well as supporting visualizations, and allows the user to export the visualization as an image file.

5.1 Artificial datasets

We begin by visualizing a small artificial dataset (AD1) consisting of 11 6-dimensional objective vectors introduced in Section 2.2. We visualize (AD1) using a standard parallel coordinate plot in Figure 7. The figure shows that there are vectors in clusters, but the exact number of clusters is not noticeably clear. Additionally, as parallel coordinate plots are not designed to display information related to correlation of objectives, that information is lost in this visualization.

Figure 8 visualizes (AD1) using the SCORE bands. We used Bayesian Gaussian mixture modelling to calculate the objective vector clusters and Method 1 to determine the axis placement. The three clusters of objective vectors are immediately clear as three bands of different colors. The clusters of objective functions are also clearly visible as three pairs: (f2, f1), (f3, f4), (f6, f5). The objectives that are closer to each other (f1 and f2, for example) have a very low degree of "crossing over" of bands, signifying a high correlation. On the other hand, the bands "cross over" much larger (vertical) distances between neighboring objectives that are farther apart (such as f1 and f3), signifying a negative correlation.

The second dataset (AD2) mentioned in Section 2.2 has three clusters of objectives with high in-group correlation (consisting of 2, 3, and 4 objectives

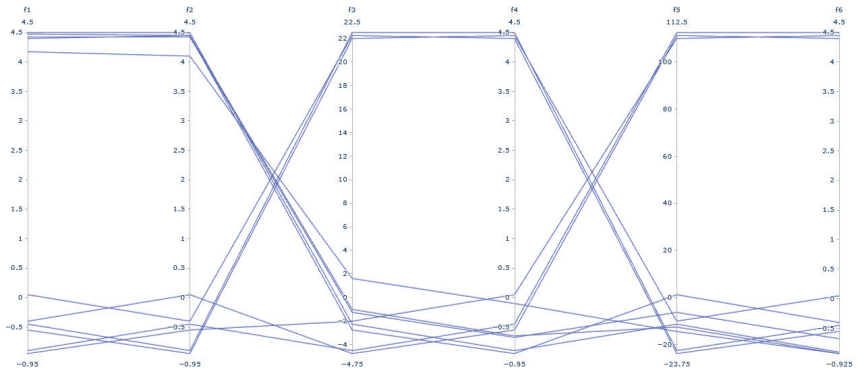


Figure 7: Visualizing (AD1) using parallel coordinate plot

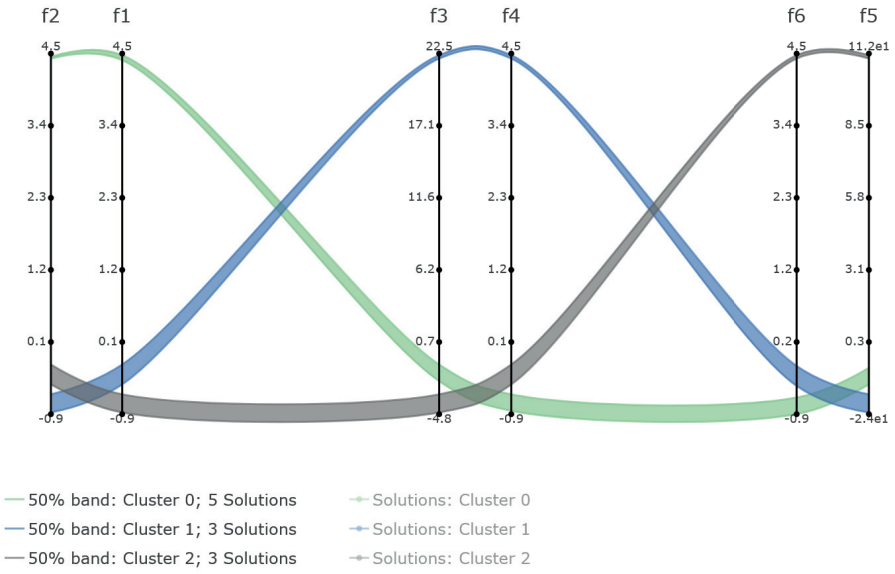


Figure 8: Visualizing (AD1) using SCORE band visualization

respectively). While creating the dataset, we ordered the objectives such that consecutive objectives (for example, f1 and f2, or f5 and f6) are negatively correlated. As seen in Figure 9, such datasets can be particularly challenging to interpret using a standard parallel coordinate plot.

We visualize the dataset (AD2) using SCORE bands in Figures 10a and 10b. We used the same parameter values for the visualization as for (AD1). Figure 10a shows the dataset in the form of bands of clustered objective vectors, whereas Figure 10b hides the bands and shows the solutions directly. The three

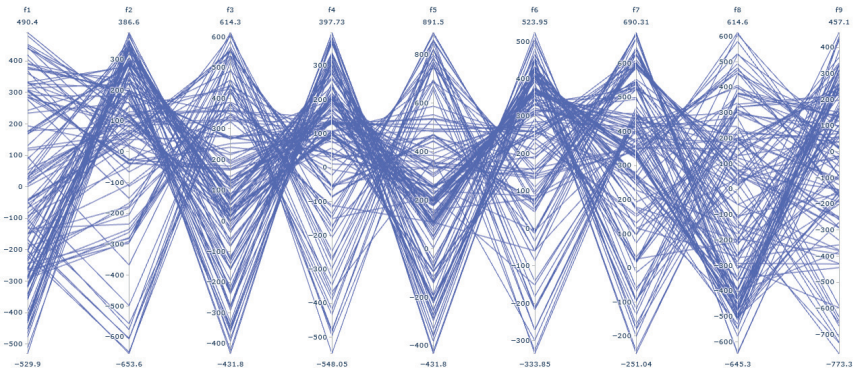


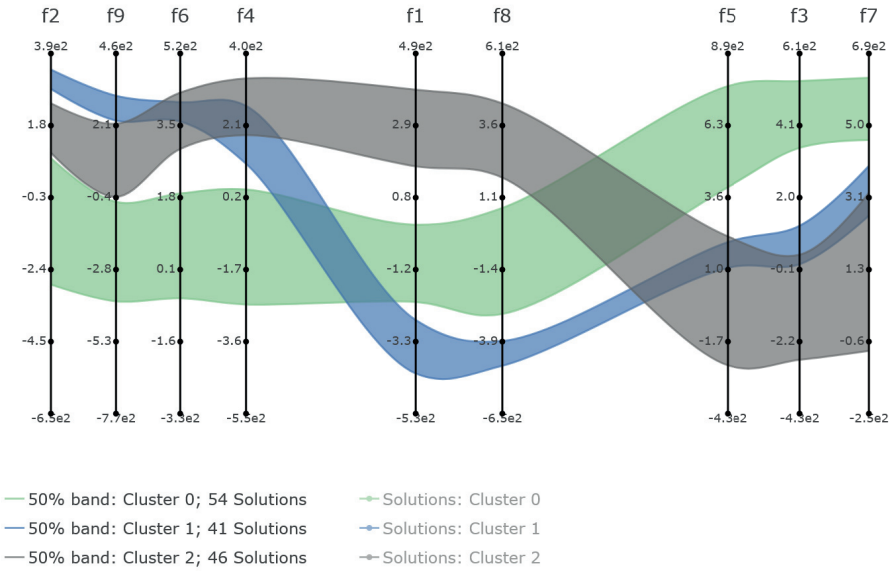
Figure 9: Visualizing (AD2) using parallel coordinate plot

clusters of objective functions are clearly visible (with 4, 2 and 3 objectives forming the three clusters). As mentioned earlier, neighboring objectives that are placed closer to each other have a high correlation. A consequence of this fact is that a DM can simultaneously improve objectives belonging to such clusters without much compromise. We can see this in Figure 10b with objectives f5, f3, and f7. Most of the solutions (especially those belonging to cluster 2 (black)) have value paths that are nearly parallel to each other, with minimal crossing over. We can improve these three objectives simultaneously (at the cost of other objectives). Hence, by ordering and placing axes according to Pearson correlation coefficients, we can simplify the decision-making process.

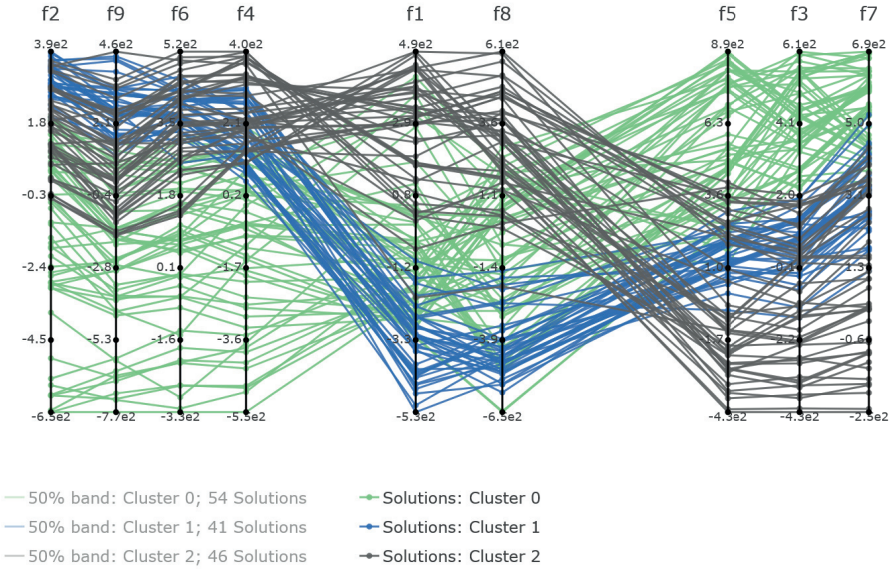
Figure 10b is further simplified by introducing bands in Figure 10a. As (AD2) did not have clustered objective vectors, the clusters and bands created for Figure 10a may be misleading. Some DMs may still prefer Figure 10a as a starting point for decision making. The bands provide simple abstractions for the patterns followed by groups of objective vectors (regardless of whether the clusters genuinely exist or not). Once a DM identifies a region of interest using the bands, we can hide the bands and show individual solutions. We discuss this aspect further in the following subsection.

5.2 Benchmark dataset

We show the effectiveness of using clustering algorithms even with datasets, where no real clusters exist using the DTLZ5 benchmark problem with a degenerate Pareto front. We consider 3 objectives and denote the set of 1000 nondominated objective vectors by (3-DTLZ5). Figures 11 and 12 show (3-DTLZ5) as SCORE band visualizations. For generating Figure 11, we used the DBSCAN algorithm to generate the solution clusters. As there are no clusters in the dataset, the visualization puts all solutions in a single cluster. Hence, the resulting singular band does not provide much information to the DM, and it



(a) Visualization of bands



(b) Visualization of individual solutions

Figure 10: Visualization of (AD2) using SCORE bands.

is necessary to display individual solutions, instead.

Figure 12, on the other hand, uses Gaussian mixture models for clustering. The algorithm forcibly breaks the dataset down into many clusters of similar

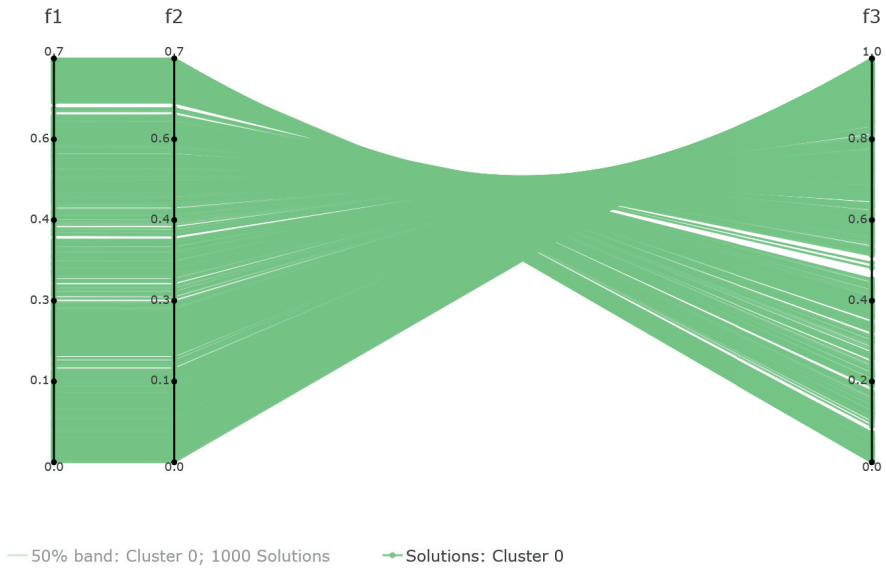


Figure 11: SCORE band visualization of nondominated objective vectors for (3-DTLZ5) using DBSCAN as the clustering algorithm (the band is hidden and individual solutions are shown).

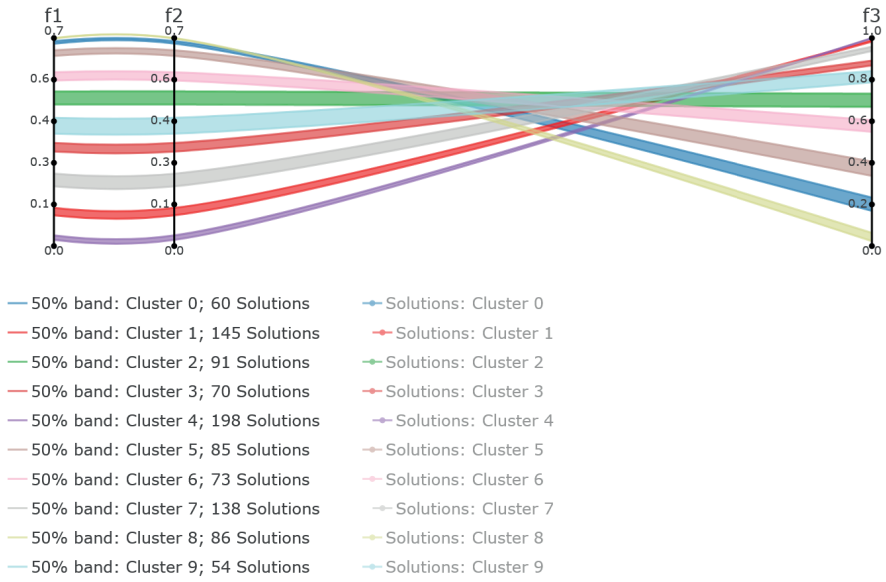


Figure 12: SCORE band visualization of (3-DTLZ5) using Gaussian mixture models as the clustering algorithm.

sizes. These clusters group solutions that are close to each other in the objective space, and individual clusters lie adjacent to other clusters along the degenerate curve of the Pareto front. These clusters are created randomly, and running the clustering algorithm multiple times returns slightly different groupings (both in the number and members of clusters). While the individual clusters have no real significance, they provide a simpler way of understanding the patterns in the dataset. Instead of focusing on hundreds or thousands of solutions, as seen in Figure 11, a DM can focus on a much smaller number of bands. Information about the trade-offs and correlations can easily be understood by following the paths of the bands and comparing the relative distances between the axes, respectively. It is also easier to compare a small number of bands to discover a region of interest than doing the same in a plot with thousands of solutions. When a DM finds such a region, they can focus on the solutions belonging to the clusters in the regions and hide all other solutions/bands, effectively "zooming" into the region of interest.

5.3 Real-life data-based problems

To demonstrate the usage of the SCORE band visualizations in real-life problems, we use the general aviation aircraft design (GAA) problem (Shah et al., 2011). We use 709 nondominated objective vectors for the problem obtained in Mazumdar et al., 2020 with eleven objectives. We visualize this (GAA) dataset using SCORE bands in Figures 13 and 14 and present a supporting visualization in Figure 15 in the form of a heatmap of pairwise Pearson correlation coefficients of the eleven objectives.

Figure 13 uses DBSCAN as the clustering algorithm for (GAA), whereas Figure 14 uses Gaussian mixture models. In both figures, it is immediately apparent that there are three groups of objectives: two groups with high in-group correlations ({f1, f2, f3, f5, f6} and {f7, f8, f9}) and the third group with low correlations to all objectives {f4, f10, f11}. We can verify this property by looking at the pairwise correlations in Figure 15. Furthermore, by following the trace of one of the bands (for example, the grey band for "Cluster 7" in Figure 13 which has the lowest value in the f2 objective), it is clear that there is no or minimal in-group trade-off between the objectives of the first two groups. On the other hand, there are significant out-group trade-offs between the objectives of the same two groups. Such a behaviour in the data makes the decision-making process significantly simpler as instead of focusing on the trade-offs between **eleven** objectives, a DM can focus on the first **two** groups and the remaining **three** objectives.

Even though Figures 13 and 14 visualize the same dataset, they look significantly different because of the behaviour of the corresponding clustering algorithms. Using the DBSCAN algorithm results in a very simplified plot, which reduces the time taken by a DM to visually gather the information needed to understand simple patterns in the data. However, this simplicity hides the solutions that can clearly be seen in Figure 14. While this is a major downside in static visualizations, we solve this problem by making the plots interactive and

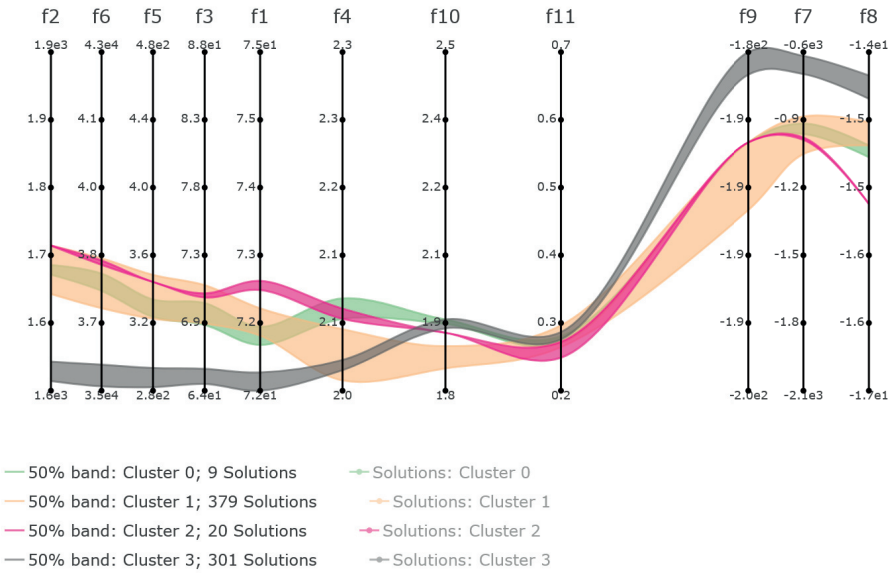


Figure 13: Score band visualization of the GAA dataset using DBSCAN as the clustering algorithm

enabling the DM to visualize all solutions from one or more clusters at any time. We cannot predict the behaviour of the clustering algorithms, as the results are dependent on both the nature of the objectives and the distribution of solutions to be visualized (discovered by the optimization algorithm). Hence, we recommend that DMs should be presented with at least two different SCORE band visualizations using different clustering algorithms.

6 Conclusions

In this paper, we have proposed SCORE band visualizations as a novel way of presenting the objective vectors of a multiobjective optimization problem with many objectives to a decision maker. The proposed technique explicitly addresses the case of many objectives in which it becomes difficult for the decision maker to process the solution candidates generated. While different methods generate varying amounts of solution information, the task of evaluating this information, all other things equal, only gets more demanding as the number of objectives increases.

Already in the three-objective case, a direct visualization of a Pareto front in the objective space is difficult, and it is generally not useful at all in cases of more than three objectives. Hence, there is a growing need for economical and meaningful presentations of Pareto optimal or nondominated objective vectors.

SCORE bands support decision making by identifying patterns in the infor-

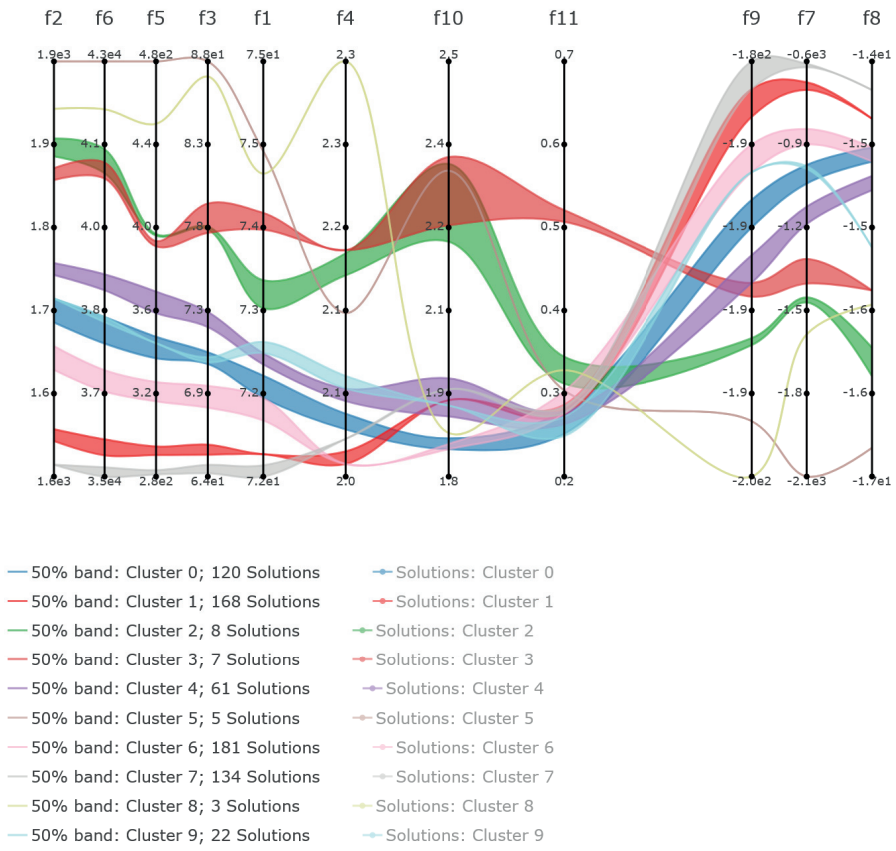


Figure 14: SCORE bands of the (GAA) dataset using Gaussian mixture models as the clustering algorithm

	f2	f6	f5	f3	f1	f4	f10	f11	f9	f7	f8
f2	1.0	1.0	0.99	0.98	0.95	0.67	0.2	0.16	-0.96	-0.98	-0.89
f6	1.0	1.0	0.99	0.99	0.96	0.7	0.25	0.18	-0.96	-0.98	-0.89
f5	0.99	0.99	1.0	0.99	0.94	0.66	0.2	0.14	-0.97	-0.99	-0.87
f3	0.98	0.99	0.99	1.0	0.96	0.71	0.27	0.23	-0.95	-0.98	-0.89
f1	0.95	0.96	0.94	0.96	1.0	0.74	0.32	0.28	-0.88	-0.94	-0.91
f4	0.67	0.7	0.66	0.71	0.74	1.0	0.58	0.33	-0.63	-0.65	-0.69
f10	0.2	0.25	0.2	0.27	0.32	0.58	1.0	0.48	-0.13	-0.26	-0.41
f11	0.16	0.18	0.14	0.23	0.28	0.33	0.48	1.0	-0.1	-0.15	-0.25
f9	-0.96	-0.96	-0.97	-0.95	-0.88	-0.63	-0.13	-0.1	1.0	0.96	0.78
f7	-0.98	-0.98	-0.99	-0.98	-0.94	-0.65	-0.26	-0.15	0.96	1.0	0.88
f8	-0.89	-0.89	-0.87	-0.89	-0.91	-0.69	-0.41	-0.25	0.78	0.88	1.0

Figure 15: Pearson correlation coefficient values between the objectives in the (GAA) dataset.

mation and breaking it down into digestible components to make it easier to gain major insights. The decision maker can easily grasp correlated objectives since they are arranged next to each other. Moreover, in contrast to common techniques, we vary the distance among objectives to transmit the degree of correlation. The user can choose between a visualization that shows all single solutions as well as their combination into bands. The latter reduce the cognitive burden by reducing the amount of hundreds or thousands of solutions to a much smaller number of colored bands. Our aim is to provide assistance by an intelligent way of visualization which is applicable beyond multiobjective visualization.

Our future research directions include integrating SCORE bands in an interactive decision making process. This will allow us to utilize these visualizations to provide preference information to interactive algorithms such as those implemented in the DESDEO framework. This will facilitate us to conduct case studies and analyze how decision makers use this tool to solve real problems.

Acknowledgements

The idea of this research was developed at the Dagstuhl Seminar 20031 (Scalability in Multiobjective Optimization). This research was partly funded by the Academy of Finland (grant number 322221). The research is related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization, jyu.fi/demo) of the University of Jyvaskyla.

References

Ankerst, M., Berchtold, S., & Keim, D. (1998). Similarity clustering of dimensions for an enhanced visualization of multidimensional data. *Proceedings of the IEEE Symposium on Information Visualization*, 52–60.

Cajot, S., Schüler, N., Peter, M., Koch, A., & Maréchal, F. (2019). Interactive optimization with parallel coordinates: Exploring multidimensional spaces for decision support. *Frontiers in ICT*, *5*, article 32.

Dächert, K., Klamroth, K., Miettinen, K., & Steuer, R. (2020). KaKaRaKe – user-friendly visualization for multiobjective optimization with high-dimensional objective vectors. In C. Fonseca, K. Klamroth, G. Rudolpth, & M. Wiecek (Eds.), *Scalability in multiobjective optimization, report from dagstuhl seminar 20031* (pp. 97–103). Dagstuhl.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In A. Abraham, L. Jain, & R. Goldberg (Eds.), *Evolutionary multiobjective optimization. advanced information and knowledge processing* (pp. 105–145). Springer.

Fonseca, C. M., Antunes, C. A., Lacour, R., Miettinen, K., Reed, P. M., & Tusar, T. (2015). Visualization in multiobjective optimization. In S. Greco, K. Klamroth, J. Knowles, & G. Rudolph (Eds.), *Understanding complexity in multiobjective optimization, report from dagstuhl seminar 15031* (pp. 129–139).

Geoffrion, A., Dyer, J., & Feinberg, A. (1972). An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management Science*, *19*(4), 357–368.

Gettinger, J., Kiesling, E., Stummer, C., & Vetschera, R. (2013). A comparison of representations for discrete multi-criteria decision problems. *Decision Support Systems*, *54*(2), 976–985.

Haara, A., Pykäläinen, J., Tolvanen, A., & Kurttila, M. (2018). Use of interactive data visualization in multi-objective forest planning. *Journal of Environmental Management*, *210*, 71–86.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.

Hettenhausen, J., Lewis, A., & Mostaghim, S. (2010). Interactive multi-objective particle swarm optimization with heatmap-visualization-based user interface. *Engineering Optimization*, *42*(2), 119–139.

Hossain, S. (2019). Visualization of Bioinformatics Data with Dash Bio. In C. Calloway, D. Lippa, D. Niederhut, & D. Shupe (Eds.), *Proceedings of the 18th Python in Science Conference* (pp. 126–133).

Ibrahim, A., Rahnamayan, S., Martin, M. V., & Deb, K. (2016). 3d-radvis: Visualization of Pareto front in many-objective optimization. *2016 ieee congress on evolutionary computation (cec)* (pp. 736–745). IEEE.

Inselberg, A., & Dimsdale, B. (1987). Parallel coordinates for visualizing multidimensional geometry. In T. Kunii (Ed.), *Computer graphics 1987, proceedings of computer graphics international* (pp. 25–44).

Korhonen, P., & Wallenius, J. (2008). Visualization in the multiple objective decision-making framework. In J. Branke, K. Deb, K. Miettinen, & R. Slowinski (Eds.), *Multiobjective optimization: Interactive and evolutionary approaches* (pp. 195–212). Springer.

Liu, J., Dwyer, T., Marriott, K., Millar, J., & Haworth, A. (2018). Understanding the relationship between interactive optimisation and visual analytics in the context of prostate brachytherapy. *IEEE Transactions on Visualization and Computer Graphics*, *24*(1), 319–329.

Lotov, A. V., Bushenkov, V. A., & Kamenev, G. K. (2004). *Interactive decision maps: Approximation and visualization of Pareto frontier*. Kluwer Academic Publishers.

Lotov, A. V., & Miettinen, K. (2008). Visualizing the Pareto frontier. In J. Branke, K. Deb, K. Miettinen, & R. Slowinski (Eds.), *Multiobjective optimization: Interactive and evolutionary approaches* (pp. 213–243). Springer.

Mazumdar, A., Chugh, T., Hakanen, J., & Miettinen, K. (2020). An interactive framework for offline data-driven multiobjective optimization. In B. Filipič, E. Minisci, & M. Vasile (Eds.), *Bioinspired optimization methods and their applications* (pp. 97–109). Springer International Publishing.

Meignan, D., Frayret, J.-M., & Pesant, G. (2015). Interactive planning system for forest road location. *Journal of Heuristics*, *21*(6), 789–817.

Miettinen, K. (2014). Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum*, *36*(1), 3–37.

Misitano, G., Saini, B. S., Afsar, B., Shavazipour, B., & Miettinen, K. (2021). DESDEO: The modular and open source framework for interactive multiobjective optimization. *IEEE Access*, *9*, 148277–148295.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Plotly Technologies Inc. (2015). *Collaborative data science*. https://plot.ly

Raseman, W., Jacobson, J., & Kasprzyk, J. (2019). Parasol: An open source, interactive parallel coordinates library for multi-objective decision making. *Environmental Modelling & Software*, *116*, 153–163.

Reback, J., McKinney, W., jbrockmendel, den Bossche, J. V., Augspurger, T., Cloud, P., gfyoung, Sinhrks, Klein, A., Roeschke, M., Hawkins, S., Tratner, J., She, C., Ayd, W., Petersen, T., Garcia, M., Schendel, J., Hayden, A., MomIsBestFriend, ... Mehyar, M. (2020). *Pandas-dev/pandas: Pandas* (Version latest). Zenodo. https://doi.org/10.5281/zenodo.3509134

Shah, R. A., Reed, P. M., & Simpson, T. W. (2011). Many-objective evolutionary optimisation and visual analytics for product family design. In L. Wang, A. H. C. Ng, & K. Deb (Eds.), *Multi-objective evolutionary optimisation for product design and manufacturing* (pp. 137–159). Springer.

Shintyakov, D. (2020). *tsp-solver*. Retrieved January 27, 2022, from https://github.com/dmishin/tsp-solver

Talukder, A., & Deb, K. (2020). Paletteviz: A visualization method for functional understanding of high-dimensional Pareto-optimal data-sets to aid multi-criteria decision making. *IEEE Computational Intelligence Magazine*, *15*(2), 36–48.

Trinkaus, H. L., & Hanne, T. (2005). knowCube: A visual and interactive support for multicriteria decision making. *Computers and Operations Research*, *32*(5), 1289–1309.

Tusar, T., & Filipic, B. (2015). Visualization of Pareto front approximations in evolutionary multiobjective optimization: A critical review and the prosection method. *IEEE Transactions on Evolutionary Computation*, *19*(2), 225–245.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*(86), 2579–2605.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272.

Wegman, E. J. (1990). Hyper-dimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, *85*, 664–675.

Woodruff, M. J., Reed, P. M., & Simpson, T. W. (2013). Many objective visual analytics: Rethinking the design of complex engineered systems. *Structural and Multidisciplinary Optimization*, *48*(1), 201–219.

Yang, D., Di Stefano, D., Turrin, M., Sariyildiz, S., & Sun, Y. (2020). Dynamic and interactive re-formulation of multi-objective optimization problems for conceptual architectural design exploration. *Automation in Construction*, *118*, article 103251.

Zhen, L., Li, M., Cheng, R., Peng, D., & Yao, X. (2017). Adjusting parallel coordinates for investigating multi-objective search. In Y. Shi, K. Tan, M. Zhang, K. Tang, X. Li, Q. Zhang, Y. Tan, M. Middendorf, & Y. Jin (Eds.), *Simulated evolution and learning* (pp. 224–235). Springer.

PIV

OPTIMISTIC NAUTILUS NAVIGATOR FOR MULTIOBJECTIVE OPTIMIZATION WITH COSTLY FUNCTION EVALUATIONS

by

Bhupinder Singh Saini, Michael Emmerich, Atanu Mazumdar, Bekir Afsar, Babooshka Shavazipour, Kaisa Miettinen 2022

Journal of Global Optimization, 83, 865 – 889

https://doi.org/10.1007/s10898-021-01119-7

Journal of Global Optimization
https://doi.org/10.1007/s10898-021-01119-7

Optimistic NAUTILUS navigator for multiobjective optimization with costly function evaluations

Bhupinder Singh Saini[1] · Michael Emmerich[1,2] · Atanu Mazumdar[1] · Bekir Afsar[1] · Babooshka Shavazipour[1] · Kaisa Miettinen[1]

Received: 20 November 2020 / Accepted: 2 December 2021

Abstract
We introduce novel concepts to solve multiobjective optimization problems involving (computationally) expensive function evaluations and propose a new interactive method called O-NAUTILUS. It combines ideas of trade-off free search and navigation (where a decision maker sees changes in objective function values in real time) and extends the NAUTILUS Navigator method to surrogate-assisted optimization. Importantly, it utilizes uncertainty quantification from surrogate models like Kriging or properties like Lipschitz continuity to approximate a so-called optimistic Pareto optimal set. This enables the decision maker to search in unexplored parts of the Pareto optimal set and requires a small amount of expensive function evaluations. We share the implementation of O-NAUTILUS as open source code. Thanks to its graphical user interface, a decision maker can see in real time how the preferences provided affect the direction of the search. We demonstrate the potential and benefits of O-NAUTILUS with a problem related to the design of vehicles.

Keywords Interactive methods · Multiobjective optimization problems · Decision makers · Preference information · Computational cost · Kriging

1 Introduction

Multiobjective optimization deals with the simultaneous minimization or maximization of multiple conflicting objective functions. There, instead of a single optimal solution, so-called Pareto optimal solutions can be identified with different trade-offs. They form a Pareto optimal set. Typically, preference information from a domain expert, a decision maker (DM), is needed to identify the most preferred one among the mathematically incomparable Pareto optimal solutions.

✉ Bhupinder Singh Saini
bhupinder.s.saini@jyu.fi

[1] Faculty of Information Technology, University of Jyvaskyla, P.O. Box 35 (Agora), FI-40014 Jyvaskyla, Finland

[2] Faculty of Science, Leiden Institute of Advanced Computer Science, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Published online: 03 January 2022

In multiobjective optimization, relevant questions include: How can we scale to problems with a large number of objectives? How can we solve problems with realistic, computationally expensive objective function formulations? How can an algorithm conveniently integrate a DM's preferences into the search? This article answers these questions by proposing a novel method called *Optimistic NAUTILUS Navigator method*, for short, O-NAUTILUS, which extends the interactive NAUTILUS Navigator method [38] to an *online data-driven approach* [18], sparingly needing new objective function evaluations during the interactive solution process and displaying additional information to support the DM. O-NAUTILUS combines two methodologies, which we briefly discuss next: surrogate modelling and navigation methods.

When dealing with computationally expensive problems, the computing resources are usually limited and one has to think about how to use them wisely. For instance, in engineering optimization, numerical simulation is often needed to compute the objective function values, and the evaluation of an objective function can take from several minutes up to hours [25]. One can reduce the computational cost by replacing the original (expensive) objective functions with computationally less costly approximation functions, typically learned from previous evaluation data. They are called *surrogate models* or *metamodels* [2,19,37,39,44,46]. Different surrogate models have been developed in the literature (e.g., radial basis functions [36], neural networks [23,32], support vector regression [3], polynomial regression [17] and Kriging [22,25]) and utilized in various optimization methods. To get an overview of the available methods to handle computationally expensive multiobjective optimization problems, we refer to [43] (for exact and deterministic methods) and [7] (for evolutionary methods). To solve such problems, we cannot necessarily rely on mathematical properties such as differentiability or convexity. Metaheuristic approaches like evolutionary algorithms do not make any such assumptions [8]. However, neither their global nor local convergence can always be guaranteed [40].

An important aspect when using surrogate models is the handling of prediction uncertainty. Although replacing the original expensive functions with an approximated one is intended to reduce the computational cost of function evaluations, it often leads to a loss of accuracy. Approximations include some errors making the solutions inexact. Therefore, in multiobjective optimization with expensive black-box functions, we cannot always guarantee to reach actual Pareto optimality but can only compute approximations, particularly when we have a limited computation budget. Consequently, we cannot always explore all parts of the feasible region, and some Pareto optimal solutions which can be of interest to a DM may remain undiscovered. However, it is possible to estimate the range of improvements that may be achieved by further exploration using uncertainty quantification techniques. As a surrogate model, Kriging (Gaussian process regression) [22,25,33] is frequently used since it provides uncertainty quantification in the form of a local mean squared error in addition to the predicted value of the original function. Furthermore, some mathematical properties (such as Lipschitz continuity) may hold, which can be used to provide lower and upper bounds of function values at yet un-evaluated decision vectors. In the literature, Lipschitz continuity has been utilized in deterministic approaches which can guarantee the global convergence of solutions under certain conditions (e.g., [11,16,35,39,42,46]). For Lipschitz continuous functions, lower and upper bounds can be relatively plainly calculated [24,34,48].

The second methodology in this article is navigation, a special type of interactive method [14]. As mentioned, preference information of a DM is typically needed to find the most preferred solution. We can classify multiobjective optimization methods based on when preference information is incorporated [27]. In *a priori* methods, the DM provides hopes and expectations first, and then a solution which matches them as well as possible is found, but

the hopes may be unrealistic. Alternatively, a representative set of Pareto optimal solutions is found, and then the DM must select the best of them in *a posteriori* methods. However, generating a representative set may be computationally demanding and comparing many solutions cognitively demanding. Interactive methods aim to avoid these shortcomings.

In interactive methods [27,29,31], the DM takes part in the solution process iteratively and directs it with one's preference information. At the same time, (s)he learns about the problem, trade-offs involved and what kind of solutions are available. Thanks to learning, (s)he can also adjust preferences if so desired. Furthermore, only a limited amount of information needs to be processed at a time, which decreases cognitive load. The DM can provide preference information in different ways, for instance, as it is done in this paper, by providing aspiration levels representing desirable objective function values. These aspiration levels constitute a so-called reference point.

Many different interactive multiobjective optimization methods have been developed in the literature (see, e.g. [29,31] and references therein) and most of them deal with Pareto optimal solutions. Because of trade-offs between the objective functions, to achieve any improvement in one objective, the DM must sacrifice in some others, and this may hinder the DM's willingness to move. Accordingly, trade-off-free interactive methods such as the NAUTILUS family [30] have been proposed. They start from an inferior solution and iteratively approach Pareto optimal solutions by simultaneously improving all the objectives while following the DM's preferences. The solution process ends when a Pareto optimal solution is reached.

NAUTILUS Navigator [38] combines NAUTILUS ideas with navigation [14]. Supported by a visual user interface, the DM can navigate to see how objective function values evolve in real time and improve all objective values simultaneously. The DM directs the navigation with reference points and the search progresses towards them.

The contribution of this article is as follows. We focus at solving problems that contain computationally expensive objective functions. Our new method, O-NAUTILUS, uses surrogate models with uncertainty quantification, and can be applied in combination with both heuristic and deterministic optimization algorithms. In O-NAUTILUS, we consider the probability or possibility of extending the estimated Pareto optimal set, constructed by using surrogate models. In our proposed method, we use uncertainty quantification in the form of (confidence) bounds from Kriging or Lipschitzian models, to build and update an optimistic approximation of the Pareto optimal set and then use this information in the interactive method.

For biobjective problems, there have been some attempts in the literature to visually aid the DM in choosing a preferred solution using surrogate models in [47]. However, our proposed method is not limited to biobjective problems. Another significant difference is that we use the optimistic approximation as a part of an interactive method to aid the DM in making targeted function evaluations. Accordingly, the DM has an option to extend the search area and cross the current borders of the estimated Pareto optimal set for further discovery towards optimistic boundaries. This will trigger an exploration phase: new evaluations with the costly objective functions are conducted in a targeted way in order to assess possibilities to extend the Pareto optimal set and find an improvement in the preferred direction.

An important characteristic of O-NAUTILUS is the alternating phases in the algorithm which require the presence of a DM (computationally fast phase) and which do not require the presence of a DM (computationally expensive phase). The DM may use one's expertise to judge how long a single exact function evaluation would take. The DM's attention is not needed during the computationally expensive phase.

The rest of the paper is structured as follows. We provide some background material together with concepts and notations in Sect. 2. In Sect. 3, we introduce our new method,

O-NAUTILUS. We demonstrate the applicability of O-NAUTILUS, and compare it with NAUTILUS Navigator with a case study in Sect. 4. Finally, we conclude and mention future research directions in Sect. 5.

2 Background: basic concepts and notation

This section covers basic concepts and notation of multiobjective optimization, NAUTILUS methods and surrogate models needed in the rest of the paper. As said, we apply Kriging and Lipschitzian models as surrogate models.

2.1 Multiobjective optimization

We consider multiobjective optimization problems with $k \geq 2$ objective functions $f_i : S \to \mathbb{R}$

$$\begin{aligned} &\text{minimize } \{f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ &\text{subject to } \mathbf{x} \in S, \end{aligned} \tag{1}$$

where vectors of decision variables (for short, *decision vectors*) $\mathbf{x} = (x_1, \ldots, x_n)^T$ belong to the feasible set $S \subset \mathbb{R}^n$ in the decision space. We define *objective vectors* as vectors in the *objective space* $\mathbb{R}^k$ that consist of objective function values $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T$. Here we assume that objective functions are continuous and their evaluations are expensive.

Because objective functions are typically conflicting with each other, it is not possible to find a solution with all objectives reaching their individual optima and thus we consider so-called Pareto optimal solutions. In them, no objective function value can be improved without impairment in at least one of the others. We say that $\mathbf{z}^{(1)}$ *dominates* $\mathbf{z}^{(2)}$ (written as $\mathbf{z}^{(1)} \succ \mathbf{z}^{(2)}$) with $\mathbf{z}^{(1)}, \mathbf{z}^{(2)} \in \mathbb{R}^k$, if $z_i^{(1)} \leq z_i^{(2)}$ for $i = 1, \ldots, k$ and $z_j^{(1)} < z_j^{(2)}$ for at least one index j. If $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ do not dominate each other, they are called mutually nondominated. Furthermore, a decision vector $\mathbf{x}^* \in S$ and the corresponding objective vector $\mathbf{f}(\mathbf{x}^*)$ are called Pareto optimal, if there does not exist another $\mathbf{x} \in S$ such that $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{f}(\mathbf{x}^*)$. Typically, problem (1) has many Pareto optimal solutions constituting a so-called Pareto optimal set E. Its image in the objective space is called a Pareto (optimal) front $\mathbf{f}(E)$. Here, we refer to objective vectors that are mappings of decision vectors as solutions. In addition, we call vectors in the objective space without any corresponding decision vector as points.

Usually, we need a DM with domain expertise to decide which Pareto optimal solution is the most preferred one satisfying her/his preferences. We denote it as $\mathbf{z}_{\text{pref}}$. An analyst can also take part in the solution process. By an analyst we refer to a human or a computer program supporting the DM and typically taking care of mathematical aspects.

Information about the ranges of objective function values in the Pareto front can be useful for the DM. The best individual optima are components of an *ideal point* $\mathbf{z}^\star = (z_1^\star, \ldots, z_k^\star)^T$ with $z_i^\star = \min_{\mathbf{x} \in S} f_i(\mathbf{x}) = \min_{\mathbf{x} \in E} f_i(\mathbf{x})$ for $i = 1, \ldots, k$. The worst values represented in a *nadir point* $\mathbf{z}^{\text{nad}} = (z_1^{\text{nad}}, \ldots, z_k^{\text{nad}})^T$ with $z_i^{\text{nad}} = \max_{\mathbf{x} \in E} f_i(\mathbf{x})$ for $i = 1, \ldots, k$ are in practice difficult to calculate because the set E is unknown. The nadir point can be approximated (see, e.g., [27] and references therein). It is also possible to ask the DM to provide the worst possible objective function values (s)he can think of and constitute a nadir point of them.

An important concept in this paper is reachability. For a point $\mathbf{z} \in \mathbb{R}^k$, if $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{z}$, we say that $\mathbf{x} \in S$ is reachable from $\mathbf{z}$. Furthermore, we define a reachable region as a subset

of decision vectors in the Pareto optimal set which are reachable from $\mathbf{z}$. In addition, the image of the reachable region from $\mathbf{z}$ in $\mathbb{R}^k$ is also called a reachable region and it contains all objective function values which can be reached from $\mathbf{z}$. In the following, we assume that $\mathbf{z}$ is clear from the context. Therefore, we shorten the term as a reachable region without explicitly mentioning $\mathbf{z}$.

As mentioned in the introduction, an example of providing preference information is a reference point $\mathbf{q} = (q_1, \ldots, q_k)^T$ consisting of desirable values of each objective function provided by the DM. If the individual values q_i can be simultaneously achieved or improved in a feasible solution, the reference point is called achievable and if this is not the case, it is called unachievable.

Scalarization functions such as an achievement scalarization function (ASF) [27] can be used for solving multiobjective optimization problems in an interactive way. For a reference point $\mathbf{q}$, the ASF can be defined as:

$$s^{\mathbf{z}}(\mathbf{f}(\mathbf{x})) = max_{i=1,\ldots,k}\left[\frac{f_i(\mathbf{x}) - q_i}{z_i^{\text{nad}} - z_i^{\star\star}}\right] + \rho\sum_{i=1}^{k}(f_i(\mathbf{x}) - q_i), \tag{2}$$

where ρ is a small, positive augmentation coefficient, $z_i^{\star\star} = z_i^{\star} - \epsilon$ $(i = 1, \ldots, k)$ are components of a so-called utopian point $\mathbf{z}^{\star\star} \in \mathbb{R}^k$ and $\epsilon > 0$ is a small scalar. With preferences given as a reference point $\mathbf{q}$, we can get a Pareto optimal solution to problem (1) by minimizing the ASF in (2); see, e.g. [27,45] for details.

2.2 Overview of NAUTILUS family

As mentioned in the introduction, the idea of the interactive methods in the NAUTILUS family [30] is to enable the DM in identifying one's most preferred solution by starting from a bad solution (like a nadir point) and proceeding iteratively by gaining improvement in all objectives simultaneously. In this way, the DM receives solutions that dominate each other from one iteration to another and gets a Pareto optimal solution only at the end of the solution process. By avoiding the need of trading off between the objectives, the DM can reach any Pareto optimal solution [28].

When applying interactive methods that operate with Pareto optimal solutions throughout the solution process, the DM has to allow sacrifices in at least one objective function to find a new Pareto optimal solution. This may hinder the DM's willingness to move, referred to as anchoring [4]. Furthermore, according to the prospect theory [21], past experiences affect people's hopes and expectations, and we do not react symmetrically to gains and losses. Because of this, the DM may converge prematurely and fail to find one's most preferred solution.

All methods in the NAUTILUS family enable the DM to freely focus on the part of the Pareto front that is interesting without making sacrifices. Family members differ from each other in the way the DM provides preference information to direct the solution process and how solutions are generated from iteration to iteration. These differences are described as a NAUTILUS framework in [30].

Since the DM gradually approaches the Pareto front, the reachable region, that is, the part of the Pareto front that still can be reached without trading off, shrinks. In other words, there are other parts of the Pareto front that can only be reached if the DM goes backwards and, in that way, widens up the reachable region.

The latest member of the NAUTILUS family is NAUTILUS Navigator [38]. With it, the DM can navigate in the reachable region in real time and improve simultaneously all objective values. The information shown to the DM is a visual presentation of how the reachable ranges shrink when one approaches the Pareto front.

To be more specific, the reachable ranges describe intervals of objective function values in the subset of Pareto optimal solutions which still are reachable from the current point without trading off. If the solution process starts from the nadir point, the reachable range of each objective function is defined at the beginning with the ideal and nadir points. When the solution process continues, the ranges are defined by the so-called current iteration point and the point with the best reachable values.

To get started, NAUTILUS Navigator needs a set of solutions that approximate the Pareto optimal front. They are all assumed to be mutually nondominated. Besides, during the navigation, we lose the connection to the decision space. This is not a problem because we can guarantee that at the end of the navigation process, the DM will reach a nondominated solution and can find the corresponding decision vector in the decision space. For details of the method, see [38].

2.3 Surrogate models

In the following, by exact objective function evaluations, we mean the evaluation of the objective functions in (1). For the O-NAUTILUS method, we utilize methods that can predict function values at yet un-evaluated decision vectors $\mathbf{x}^+ \in S$ utilizing a set of N already evaluated decision vectors, say $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$, with $\mathbf{y}^{(1)} = \mathbf{f}(\mathbf{x}^{(1)}), \ldots, \mathbf{y}^{(N)} = \mathbf{f}(\mathbf{x}^{(N)})$. In multiobjective optimization, a common strategy is to train a surrogate model for each of the objective functions f_i separately. We will denote the predictions of function values with $\hat{y}_i \approx f_i(\mathbf{x}^+)$ and the corresponding exact function values with $y_i = f_i(\mathbf{x}^+)$.

Furthermore, we need methods that can assess the uncertainty of the predictions by providing an uncertainty quantification in the form of ranges in which the true outcome is (likely) to be found. Kriging models and Lipschitzian models are two common classes of such surrogate models, which is the reason why we chose them as surrogate models in our discussion. The upper bound of these ranges will be denoted with $\overline{f}_i(\mathbf{x}^+)$, and the lower bound with $\underline{f}_i(\mathbf{x}^+)$. The ranges may have a probabilistic interpretation, such as in the Kriging models, or a possibilistic, exact interpretation, such as in the Lipschitzian models.

2.3.1 Kriging and Gaussian process regression

The Kriging method and Gaussian process regression are mathematically very similar[1]. To obtain a prediction, function values at neighboring decision vectors of the new decision vector are weighted by distance and a factor that is determined in a training process. While the training can be time-consuming, predictions and uncertainty quantification for a new decision vector are very fast for such vectors that have not been evaluated yet. Therefore, in order to find promising regions for new evaluations, a Kriging model can be evaluated in many different decision vectors. As mentioned in the general introduction to surrogate models, also in the Kriging methods we handle multiple objectives typically by learning them separately

[1] Kriging (named after geo-scientist Krige) seeks to find a best linear unbiased predictor assuming the observed data is a realization of a stochastic processes or random field (not necessarily of the Gaussian type). Gaussian process regression is motivated by Bayesian reasoning and uses conditional mean and variance of Gaussian random fields to model and bound the objective function at a given decision vector.

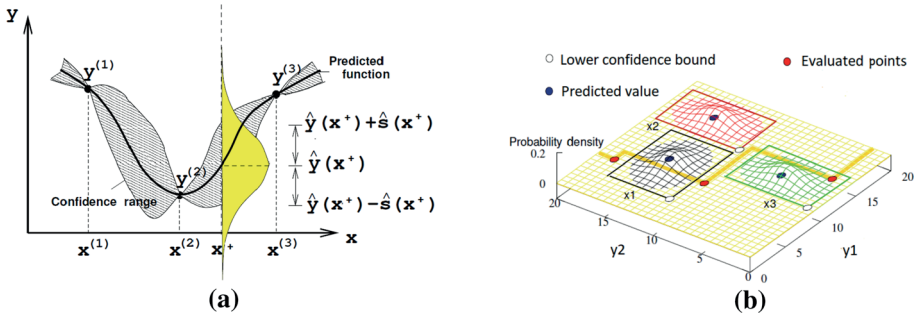


Fig. 1 Kriging prediction in **a** 1-D and **b** 2-D objective spaces

for each objective function [9]. The predictive distribution at a new, yet un-evaluated decision vector $\mathbf{x}^+$ is then given by an independent multivariate normal distribution for each decision vector with mean values $\hat{y}_i(\mathbf{x}^+)$ and standard deviations $\hat{s}_i(\mathbf{x}^+)$ for the objective functions f_i, $i = 1, \ldots, k$. Based on this, we can compute probabilistic confidence ranges with a lower bound $\underline{f}(\mathbf{x})$ and an upper bound $\overline{f}(\mathbf{x})$, respectively, defined for $i = 1, \ldots, k$ as:

$$\underline{f}_i(\mathbf{x}^+) = \hat{y}_i(\mathbf{x}^+) - \alpha \hat{s}_i(\mathbf{x}^+) \tag{3}$$

$$\overline{f}_i(\mathbf{x}^+) = \hat{y}_i(\mathbf{x}^+) + \alpha \hat{s}_i(\mathbf{x}^+), \tag{4}$$

where $\alpha \geq 0$ is a user-defined confidence level.

An illustration of a Kriging model for a 1-D decision vector and a single objective function is provided in Fig. 1a and for a biobjective problem in Fig. 1b. In Fig. 1a, the vertical axis (y) denotes the function values obtained at three decision vectors, namely $y^{(1)} = f(\mathbf{x}^{(1)})$, $y^{(2)} = f(\mathbf{x}^{(2)})$ and $y^{(3)} = f(\mathbf{x}^{(3)})$. The predictions at a new decision vector $\mathbf{x}^+$ are given by a 1-D normal distribution with a mean value $\hat{y}(\mathbf{x}^+)$ and a standard deviation $\hat{s}(\mathbf{x}^+)$ quantifying the uncertainty of the prediction. Figure 1b illustrates three predictions for a biobjective problem. Again, the uncertain outcome of the expensive evaluations is quantified by means of normal distributions that are indicated in the figure by their probability density function (PDF). In the figure, we deal with bi-variate distributions. They can be used to estimate 2-D confidence ranges which are indicated in the figure by rectangles. From these, it is straightforward to compute optimistic bounds for the objective vector resulting from an evaluation at a specific decision vector.

The Kriging or Gaussian process method underlies several statistical assumptions, most importantly a distance based correlation between outputs at decision vectors, where the distance is measured in the decision space. For an in-depth description of the Kriging method and its statistical motivation the reader is referred to [41]. In our experiments we use a standard implementation of Kriging with an isotropic, exponential kernel.

2.3.2 Lipschitz bounds for prediction and uncertainty quantification

The knowledge of the Lipschitz constant can help in designing global search algorithms [24]. Lower or upper bounds (also called shells) for the values of a Lipschitz continuous function can then be relatively simply computed and have been used to construct such global optimization algorithms [24,34,48]. As will be shown, using a Lipschitz constant will also yield an alternative, yet linear approach to surrogate modeling with uncertainty quantification in the form of a confidence range. As it is also based on distances in the decision space and

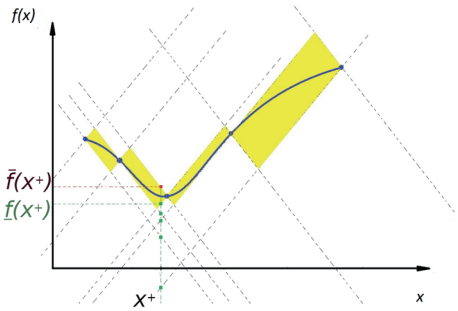


Fig. 2 Computation of Lipschitz bounds in one dimension

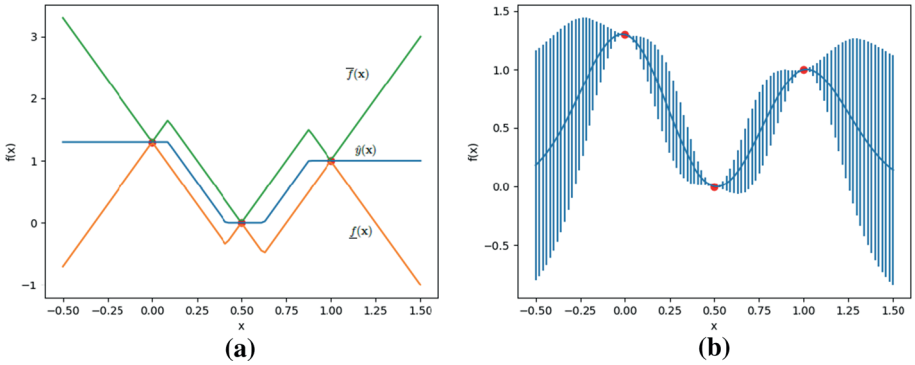


Fig. 3 Lipschitz envelope for a 1-D function (**a**). Kriging uncertainty ranges for the same data (**b**)

provides confidence bounds, it is structurally quite similar to the aforementioned Kriging method.

A function f is called Lipschitz continuous if there exists a real positive constant L (called Lipschitz constant) such that for all $\mathbf{x}, \mathbf{x}' \in S$

$$d(f(\mathbf{x}), f(\mathbf{x}')) \leq Ld(\mathbf{x}, \mathbf{x}'),$$

where $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a distance function. Let us again assume we are given a data set (set of evaluations) with N evaluated points (decision vectors). Then, if $\mathbf{x}^+$ denotes an un-evaluated decision vector, its output $y^+ = f(\mathbf{x}^+)$ is bounded by the interval $[\underline{f}(\mathbf{x}^+), \overline{f}(\mathbf{x}^+)] \subseteq \mathbb{R}$, with

$$\underline{f}_i(\mathbf{x}^+) = \max_{t=1,\ldots,N} \{\mathbf{y}_i^t - L_i d(\mathbf{x}^+, \mathbf{x}^{(t)})\} \text{ and} \tag{5}$$

$$\overline{f}_i(\mathbf{x}^+) = \min_{t=1,\ldots,N} \{\mathbf{y}_i^t + L_i d(\mathbf{x}^+, \mathbf{x}^{(t)})\}. \tag{6}$$

An example of the computation of Lipschitz bounds using these equations is given in Fig. 2. Moreover, we can define a prediction as the average of the upper and lower bound, i.e. $\hat{y}(\mathbf{x}^+) = \frac{1}{2}(\overline{f}(\mathbf{x}^+) + \underline{f}(\mathbf{x}^+))$, thus creating a Lipschitzian model. An illustration of the Lipschitzian envelope is given in Fig. 3a and a figure of the same envelope using Kriging is provided in Fig. 3b. The example data is part of an interactive Python plot, that can be accessed and modified at https://trinket.io/python3/c38e5ebdbc. Since the distance function can also be defined for multivariate decision spaces, the Lipschitzian bound calculation can also be used efficiently for high-dimensional decision spaces.

3 O-NAUTILUS method

In this section, we introduce the details of the O-NAUTILUS method. To be able to describe the O-NAUTILUS algorithm, we first need to introduce the major components and concepts utilized in the method. Hence, we describe them first and then provide a detailed pseudo-code of the algorithm.

The starting point of O-NAUTILUS is a pre-generated set of solutions. We do not make any specific assumptions regarding this set. It can, for example, be a rough approximation of the Pareto front, or even a space filling sample of solutions. The O-NAUTILUS method extends the NAUTILUS Navigator method by visualizing not just the reachable ranges, but also optimistic ranges, which represent solutions which are predicted to have good objective values, but are not represented in the set of known solutions. The O-NAUTILUS method does this by working with two sets of fronts at once. The first set is the nondominated front from the set of known solutions, the same as the one used in NAUTILUS Navigator. The second set is an optimistic estimation of the Pareto front given by a set of points in the objective space that is calculated by multiobjective minimization on the lower bounds $\underline{f}_i(\mathbf{x}), \mathbf{x} \in S, i = 1, \ldots, k$ that are estimated using the surrogate model. We call it an optimistic front to be described further in the next subsection.

Using the information provided by the optimistic front and the corresponding ranges enables the DM to strategically conduct function evaluations. In terms of navigation, this means going beyond the reachable ranges and stepping into an "optimistic area" by conducting function evaluations that are likely to find solutions in that area. This has a two-fold benefit. Firstly, the function evaluations are conducted in regions of interest of the DM. Hence, no resources are wasted in finding solutions that may not be of interest to the DM. Secondly, the newly evaluated decision vectors can be added back to the known set of solutions. Based on this new known set, surrogate models can be trained again. As this new known set contains solutions in the region of interest of the DM, the surrogate models themselves perform better in the region of interest. This means that the optimistic predictions obtained by the models are more accurate in the region of interest. Hence, as the algorithm continues, the DM gets an increasingly improved picture of the objective space in the region of interest.

In Sects. 3.1 through 3.4, we describe various components of the O-NAUTILUS method. These components are modular and can be trivially replaced by alternatives which serve a similar purpose. Section 3.5 then describes the O-NAUTILUS method, which puts together the aforementioned components to support a DM to identify the most preferred solution.

3.1 Optimistic pareto Front from surrogate models

As described in (3) and (5), Kriging and Lipschitzian models, respectively, can be used to estimate or determine an optimistic lower bound for an objective $f_i, i = 1, \ldots, k$, at any point in the decision space $\mathbf{x} \in S$ as $\underline{f}_i(\mathbf{x})$. To get an optimistic Pareto front from the surrogate models, we simply solve the following problem:

$$\begin{aligned} &\text{minimize } \{\underline{f}_1(\mathbf{x}), \ldots, \underline{f}_k(\mathbf{x})\} \\ &\text{subject to } \mathbf{x} \in S, \end{aligned} \tag{7}$$

with any appropriate solver.

Figure 4 shows the concept of an optimistic front (for a biobjective example) graphically. The blue (darker in greyscale) points belong to a set of exactly evaluated decision vectors P. The orange (lighter in greyscale) points belong to the set of solutions obtained by solving

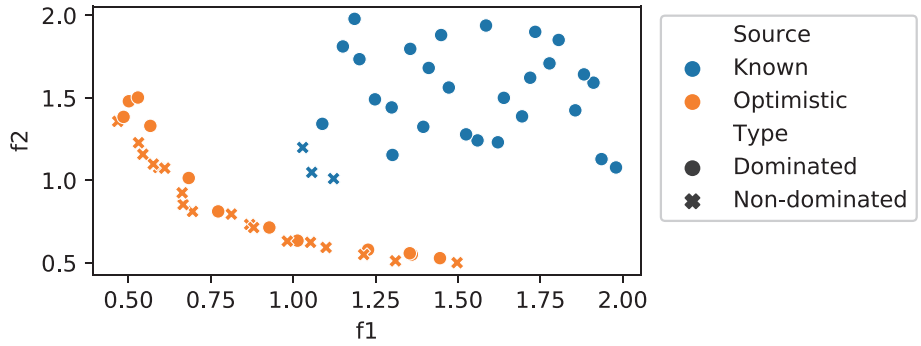


Fig. 4 Simplified figure to demonstrate the optimistic front

(7), to be denoted by P^+. The nondominated points from P form the *known front*, whereas the nondominated points from P^+ form the *optimistic front*. These are represented as blue (darker) crosses (known front) and orange (lighter) crosses (optimistic front) in Fig. 4. These fronts will be used by the other steps of the O-NAUTILUS method.

3.2 Reachable ranges

In Fig. 5, a path of the O-NAUTILUS method is visualized in the objective space as a 2-D scatter plot (5a). The blue (darker) points in Fig. 5a represent P, whereas the orange (lighter) points represent P^+. The path is also visualized as a reachable ranges plot (5b) which shows the reachable ranges for various objectives at different steps [38]. The vector $\mathbf{z}^{(i)}$ in the objective space is the *step point* at the i^{th} step and has a function similar to the *current point* in [38].

The intention of this visualization is to show how for every step point the reachable ranges change and how this can be visualized in the 2-D objective space plot (5a) and in the reachable ranges plot (5b). Note that the objective space visualization becomes impractical in higher dimensions, whereas the reachable ranges plot can still be used.

For the j^{th} objective, the *known reachable range* at step i is defined as:

$$\left[\min_{y\in P, y\succ \mathbf{z}^{(i)}} y_j, \max_{y\in P, y\succ \mathbf{z}^{(i)}} y_j\right], \tag{8}$$

whereas the *optimistic reachable range*, which is newly introduced in this paper, is defined by replacing P by P^+ in (8). These values are displayed in the *reachable range paths* as described in [38].

Focusing on the visualization of the path, we here omit details of the algorithmic procedure and interaction with the DM, which will be discussed in Sect. 3.3. The path starts in Fig. 5 at point ①. In the first step, the DM aims at merely improving f_2 which leads to ②. Then, the subsequent moves ②–⑤ are conducted such that f_1 and f_2 are equally improved. The last move ⑤–⑥ steps into the 'orange (lighter) region' where additional exploration in terms of exact function evaluation of the objective functions in the region of interest are required in order to assess the feasibility of the move. The fact that the move does not step beyond the orange region is indicating that such explorations will be promising and have a realistic chance of success.

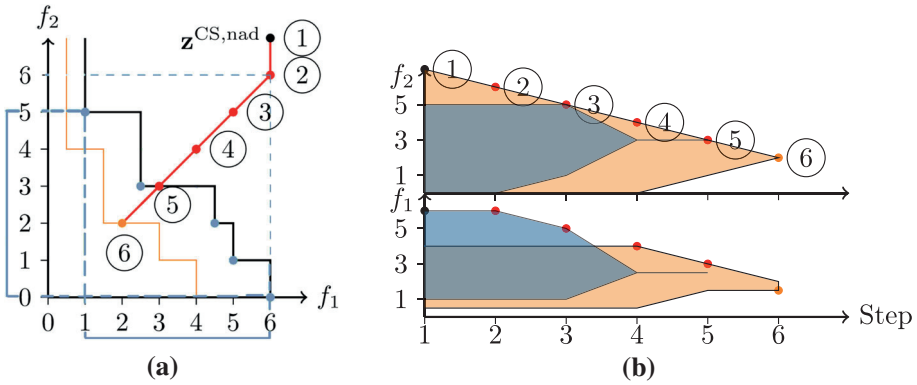


Fig. 5 Path of O-NAUTILUS visualized as a 2-D scatter plot (**a**) and as reachable ranges plot (**b**)

Looking at the relation between Fig. 5a (coordinate plot) and Fig. 5b (reachable ranges plot), let us focus on a single point on the path, say ②. As improvement is expected in all objectives, the maximal improvement of this point for which we still can guarantee the existence of a solution is indicated by the blue dashed lines (5a). This equates to values ranging from 1 to 6 units for f_1 and from 0 to 5 units for f_2, as shown by the span along the two axes in Fig. 5a. The blue (darker) ranges in Fig. 5b show the same span at step ②. The orange (lighter) optimistic range, which extends the lower bound of the blue (darker) range, indicates how much we can still realistically expect to maximally improve by further exploration. This range is determined by the optimistic front.

3.3 Navigation

O-NAUTILUS uses the concept of navigation to help a DM make function evaluations at regions of interest and arrive at a desirable solution. Similar to NAUTILUS Navigator, the navigation begins at the worst possible objective values, a point that we will call a *combined set nadir point* ($\mathbf{z}^{\mathrm{CS,nad}}$). This point is calculated as the supremum of the combined set of known and optimistic fronts. The DM then takes a "step" by advancing the step point towards the solutions in any preferred direction, thus, gaining in each objective without any trade-offs. Unlike NAUTILUS Navigator, however, the DM does not reach a solution on the known front at the end of the navigation. This is because, unlike NAUTILUS Navigator, which uses an unchanging front, both the known and the optimistic fronts in O-NAUTILUS can change with further exact function evaluations.

Instead, the navigation is conducted in the following way. Firstly, a *combined set ideal point* ($\mathbf{z}^{\mathrm{CS},*}$) is defined as the infimum of the combined set of known and optimistic fronts. From this, the *combined set utopian point* ($\mathbf{z}^{\mathrm{CS},**}$) is generated in a corresponding manner to how the utopian point is derived from the ideal point. The *combined set nadir point* is calculated as described in the previous paragraph. Then, a hyperbox is formed with the combined set utopian and nadir points as the opposing corners. This hyperbox is divided using equidistant "rungs" perpendicular to the line connecting the combined set utopian and nadir points, starting and ending on those points, as shown in Fig. 6. The step point is then constrained to be inside the hyperbox and on one of these rungs throughout the navigation process. The number of these rungs is one more than the total number of steps to be taken during the navigation process, and it is pre-defined by an analyst. A higher step count divides

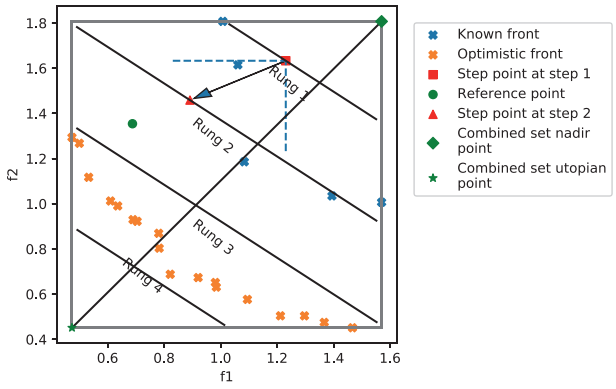


Fig. 6 Progressing the step point from step 1 to step 2 in the O-NAUTILUS algorithm. The step point (red square) jumps from rung 1 to rung 2 in the direction of the reference point (green circle). Here, combined set nadir and utopian points act as the zeroth and fifth rung, respectively

the hyperbox into smaller sections, giving the algorithm a higher resolution. This gives the DM a finer control over the navigation process.

The navigation begins at the step point at the zeroth rung, i.e. the combined set nadir point. The DM provides preference information that is used to define the direction of navigation (and hence, improvement). At any step, preference information (given in the form of aspiration levels) is valid if the corresponding reference point dominates the current step point. This is shown graphically in Fig. 6, for step 1. The step point is represented as the red square point on "Rung 1". The reference point given, shown as a green circle, dominates the step point.

Once valid preference information is provided, the step point moves from the current rung to the next, in the direction of the reference point, as shown by the arrow in Fig. 6. The step points then keep jumping on the successive rungs in the same direction at a rate that can be controlled by a DM or an analyst. At every step, the reachable ranges are calculated and shown to the DM. The DM can update preference information, i.e., provide a new reference point at any step. Note that the basic unit of O-NAUTILUS is called a "step", as opposed to "iteration", which is the terminology used in most other interactive methods. In them, two iterations are typically separated by a DM providing new preference information. On the other hand, the steps in O-NAUTILUS proceed at a constant rate even if no new preference information is obtained from the DM after the first one. The rate can have a default value.

The following formula is used to calculate the successive step points:

$$\begin{aligned} \mathbf{z}^{(\mathrm{i}+1)} &= \mathbf{z}^{(\mathrm{i})} + ss^{(i)}\mathbf{sd}^{(i)} \\ \mathbf{sd}^{(i)} &= \frac{\left(\mathbf{z}^{\mathrm{pref}} - \mathbf{z}^{(\mathrm{i})}\right)}{\|\mathbf{z}^{\mathrm{pref}} - \mathbf{z}^{(\mathrm{i})}\|} \\ ss^{(i)} &= \left(\frac{\|\mathbf{z}^{\mathrm{CS},**} - \mathbf{z}^{\mathrm{CS,nad}}\|}{i_{total}}\right)^2 \Bigg/ \frac{\left(\mathbf{z}^{\mathrm{pref}} - \mathbf{z}^{(\mathrm{i})}\right) \cdot \left(\mathbf{z}^{\mathrm{CS},**} - \mathbf{z}^{\mathrm{CS,nad}}\right)}{i_{total}\|\mathbf{z}^{\mathrm{pref}} - \mathbf{z}^{(\mathrm{i})}\|}, \end{aligned} \tag{9}$$

where i_{total} is the pre-defined maximum number of steps, $\mathbf{z}^{\mathrm{pref}}$ is a valid reference point, $\mathbf{sd}^{(i)}$ is the step direction and $ss^{(i)}$ is the step size at the i^{th} step (length of the arrow in Fig. 6). The $\cdot$ symbol is used to denote the dot product of vectors, whereas $\|\cdot\|$ denotes the magnitude or $\ell 2$ norm of a vector.

The navigation continues as long as there are known solutions reachable from the succeeding rung. At the end of these steps, the DM has a few options. (S)he can choose the last remaining known solution as the final solution, or restart the navigation process and navigate in a different direction to explore the two fronts. These options are available in NAUTILUS Navigator as well. In addition to these options, O-NAUTILUS provides the option to conduct a function evaluation at a single point. The DM, utilizing the information provided by the optimistic regions in the reachable ranges path, can decide whether to conduct an exact function evaluation to find a solution in the current region of interest (the optimistic regions of the reachable ranges plot). The mechanism to find such a potential solution to be evaluated is described in Sect. 3.4.

Based on the results of the function evaluation, the DM can choose to end the solution process, choosing the newly evaluated solution as the final solution. Alternatively, the DM can continue the search for alternative solutions. This is done by including the new solution in the known set of solutions P. Based on this updated set, new (and more accurate) surrogate models are trained. A new set of optimistic solutions P^+ is calculated as described in Sect. 3.1. The navigation is then restarted with the step point at the combined set nadir point, and with the last known aspiration levels as preference information. The DM can thus follow the same path travelled in the previous navigation phase to see how the reachable ranges have changed, or change preferences and follow a new path.

3.4 Expected ASF

Solving a surrogate assisted multiobjective optimization problem efficiently requires updating the surrogate models by utilizing an infill criterion. An infill criterion determines where the next sample is to be evaluated for updating the surrogate models. The infill criterion is obtained by optimizing an acquisition function that provides a mapping from a decision vector to a scalar value. In the literature, different acquisition functions have been suggested, i.e. expected improvement (EI) [20] and expected hypervolume improvement (EHVI) [10]. Both represent a trade-off between exploration and exploitation. The multiplicative EI (mEI) [12] is interesting in the context of the O-NAUTILUS method because it also takes into account a reference point provided by the DM. However, in our tests, solutions produced by mEI did not follow the DM's preferences very well.

We propose an infill criterion called expected ASF (eASF) which is the expected value of (2). We use Monte Carlo sampling [15] to find the expected ASF. For a decision vector $\mathbf{x}$, we sample N_S points using the distribution predicted by the surrogate models. For example, while using Kriging surrogates, we use a normal distribution and the multivariate Gaussian PDF:

$$\mathrm{PDF}_{\mathbf{f}}^{Kriging} = \prod_{i=1}^{k} \frac{1}{\hat{s}_i(\mathbf{x})\sqrt{2\pi}} \exp\left(-\frac{(\hat{y}_i(\mathbf{x}) - y_i)^2}{2\hat{s}_i(\mathbf{x})^2}\right). \tag{10}$$

Here it can be observed that the distribution is Gaussian for Kriging surrogates. However, in case of Lipschitzian surrogates, we use a multivariate uniform distribution as the PDF to draw the samples. The set of samples $\left\{\hat{\mathbf{y}}_1(\mathbf{x}), \ldots, \hat{\mathbf{y}}_{N_S}(\mathbf{x})\right\}$ that is drawn using (10) for a decision vector $\mathbf{x}^+$ is then used to calculate the ASF using (2). The set of ASF values is $\xi^{\mathbf{z}}(\mathbf{x}) = \left\{s_1^{\mathbf{z}}(\mathbf{x}), \ldots, s_S^{\mathbf{z}}(\mathbf{x})\right\}$. The final aquisition function is expected ASF or $E\left[\xi^{\mathbf{z}}(\mathbf{x})\right]$. To find the infill point we solve the following single objective optimization problem:

$$g_{ASF}(\mathbf{x}) = \text{minimize}\left\{E\left[\xi^{\mathbf{z}}(\mathbf{x})\right]\right\}$$

by using an appropriate optimization method. As the expected ASF considers the distribution of the ASF, it takes exploration in the search into account along with exploitation.

3.5 Algorithm description

Algorithm 1: O-NAUTILUS Algorithm

Input: Problem definition(1) MOP, set P of decision vectors and corresponding objective vectors, set of surrogate models SMT, multiobjective optimization algorithm (MOA) and function evaluation budget B.

1 $b \leftarrow 0$ // Function evaluation counter
2 $\mathbf{s}_1, \ldots, \mathbf{s}_k \leftarrow$ `Train(`SMT, P`)` // Surrogate-models
3 $P^+ \leftarrow$ `Optimistic_Optimize(`MOA, $\mathbf{s}_1, \ldots, \mathbf{s}_k$`)`
4 **if** `Function_Evaluations_Needed(`P, P^+`)` **then**
5 $P \leftarrow P \cup$ `Individual_Optima(`$\mathbf{s}_1, \ldots, \mathbf{s}_k, MOP$`)`
6 $b \leftarrow b + k$ // k exact objective function evaluation used
7 Go to step 2
8
9 $\mathbf{z}^{\mathrm{CS},**}, \mathbf{z}^{\mathrm{CS,nad}} \leftarrow$ `Calculate_Utopian_And_Nadir(`P, P^+`)`
10 $i \leftarrow 0$ // Step number
11 $\mathbf{z}^{(\mathrm{i})} \leftarrow \mathbf{z}^{\mathrm{CS,nad}}$ // Step point
12 `Display_Reachable_Ranges(`P, P^+, $\mathbf{z}^{\mathrm{i}}$`)`
13 **if** *DM provides new preference or* $i = 0$ **then**
14 $\mathbf{z}^{\mathrm{pref}} \leftarrow$ `Get_Preference_From_DM()`
15
16 **if** *DM wants to stop* **then**
17 Go to step 29
18
19 **if** *Front P is not reached* **then**
20 $\mathbf{z}^{(\mathrm{i})} \leftarrow$ `Compute_Next_Iteration_Point(`$\mathbf{z}^{(\mathrm{i})}$, $\mathbf{z}^{\mathrm{pref}}$`)`
21 $i \leftarrow i + 1$
22 Go to step 12
23 **if** $b < B$ ***and*** *DM wants to conduct exact function evaluation* **then**
24 $P \leftarrow P +$ `Max_Expected_ASF(`$\mathbf{z}^{(\mathrm{i})}$, $\mathbf{s}_1, \ldots, \mathbf{s}_k$`)`
25 $b \leftarrow b + 1$ // 1 exact point evaluation used
26 Go to step 2
27
28 **else**
29 `Display_Chosen_Solutions(`P, $\mathbf{z}^{(i)}$`)`
30 **end**

Next, the complete interactive O-NAUTILUS algorithm is described. We pay particular attention to the use of exact objective function evaluations and the use of surrogate function evaluations, because the number of exact objective function evaluations will govern the total computational effort. The flow of the method is given in Algorithm 1. The various functions and variables involved in the algorithm are as follows:

1. `Train` takes as its input the choice of surrogate modeling technique (SMT) and the known set of solutions (P), and returns k trained surrogate models $\mathbf{s}_1, \ldots, \mathbf{s}_k$, one for each of the k objectives. Here, SMT can be any surrogate modeling technique capable of giving optimistic predictions. In this paper, as mentioned, we consider the Kriging and Lipschitzian surrogate modeling techniques.

2. `Optimistic_Optimize` uses a multiobjective optimization method, in our implementation an evolutionary algorithm, with the surrogate models $\mathbf{s}_1, \ldots, \mathbf{s}_k$ to find an optimistic Pareto front (P^+) for the problem as described in Sect. 3.1. Note that no exact function evaluations are conducted in this step.
3. `Function_Evaluations_Needed` compares P and P^+ to determine whether further exact function evaluations are needed before the navigation and the involvement of the DM begins.
   This need may arise, for example, if all solutions in P are dominated by the nondominated solutions in P^+. This may happen because of two reasons. Firstly, the solutions in P may be far from the exact Pareto front of the problem. Alternatively, the surrogate models may not provide good predictions in certain regions, especially near the Pareto front. Either case may give misleading information to the DM when (s)he is asked to provide preferences. The first case can be resolved by conducting further exact function evaluations, preferably closer to the front. This will also lead to the generation of more samples for training the surrogate models, resolving the second case. There may be other cases where further exact function evaluations are desirable. The choice is left up to the analyst and not included in the algorithm.
4. `Individual_Optima` uses the surrogate models $\mathbf{s}_1, \ldots, \mathbf{s}_k$ to find k solutions corresponding to the maximum expected ASF (eASF) of the individual surrogate models. These solutions are then evaluated using the exact objective functions. Other strategies may be used in place of `Individual_Optima` to find good solutions to evaluate. For example, an alternative option is to use a few representative solutions from P^+.
5. `Calculate_Utopian_And_Nadir` combines the nondominated solutions from the sets P and P^+ and returns the *combined set utopian point* ($\mathbf{z}^{\mathrm{CS},**}$) and the *combined set nadir point* ($\mathbf{z}^{\mathrm{CS,nad}}$) of this combined set.
6. `Display_Reachable_Ranges` uses P, P^+ and $\mathbf{z}^{(\mathrm{i})}$ to calculate and display the known and optimistic reachable ranges. See also Subsec. 3.2.
7. `Get_Preference_From_DM` stores the most recent DM's preferences as $\mathbf{z}^{\mathrm{pref}}$.
8. `Compute_Next_Iteration_Point` calculates the step point for the next step as described in (9).
9. `Max_Expected_ASF` uses the surrogate models $\mathbf{s}_1, \ldots, \mathbf{s}_k$ to find a solution that follows the DM's preferences and is likely to lie close to the Pareto front, as described in Sect. 3.4. This solution is then evaluated using the exact objective functions and added to the set of known solutions.
10. `Display_Chosen_Solutions` displays the solutions chosen by the DM by calculating and plotting solutions from P that are reachable from $\mathbf{z}^{(i)}$.

A DM can affect the algorithm by providing preferences, controlling when and where to conduct exact function evaluations, and by terminating the algorithm once satisfied. The DM can also pause the algorithm at any point (for example, in between steps during navigation) e.g., to update preference information or to jump backwards. An analyst can further affect the algorithm by choosing the surrogate modeling algorithm used in `Train`, the optimization algorithms used in `Optimistic_Optimize` and `Individual_Optima`, and by setting the number of steps and the rate at which they progress.

4 Case study

In this section, we demonstrate how O-NAUTILUS can be used to solve a multiobjective optimization problem. The implementation of the O-NAUTILUS method and this example

are openly available at https://desdeo.it.jyu.fi as a part of the DESDEO software framework and in a Zenodo repository via the link https://doi.org/10.5281/zenodo.5396677. Kriging is used as the surrogate modeling technique and RVEA [5] as the evolutionary multiobjective optimization algorithm to find the optimistic front as it generalizes well to a high number of objectives. For evaluating expected ASF, CMA-ES [13] is used in our implementation. CMA-ES is a state-of-the-art algorithm for continuous black-box optimization.

A video showcasing the UI of the O-NAUTILUS implementation can be viewed at https://desdeo.it.jyu.fi/o-nautilus. Data generated during the experiments can also be accessed via the same link.

4.1 Crash-worthiness design of vehicles

As the role of the DM is crucial in interactive methods, we can best demonstrate the applicability of these methods with problems where the objective functions are meaningful to a DM. Therefore, we apply O-NAUTILUS to a real-world engineering design optimization problem called crash-worthiness design of vehicles, originally proposed in [26]. It describes the design of the frontal structure of vehicles for crash safety optimization. When a car accident occurs, the frontal structure of the vehicle absorbs the energy caused by the crashing in order to increase the safety of the passengers. Improving the capacity of energy absorption can often lead to an increase in the total mass of the vehicle. However, lightweight designs are needed to reduce the mass and the fuel consumption of a vehicle, accordingly. Therefore, there is a trade-off between the safety and the environmental aspects, and a balanced decision must be made in the capacity of energy absorption and the mass of the vehicle.

In the problem design, a full frontal and an offset-frontal crash test are considered to simulate real-world accidents. A full-frontal crash usually results in a higher deceleration compared to an offset-frontal crash, which can cause severe injuries to the passengers. An offset-frontal test is needed to assess the structural integrity of a vehicle. In the design of the frontal structure, these two crashing types should be considered simultaneously to improve the overall crash-worthiness of the vehicle. Hence, there are three objectives: minimizing the mass of a vehicle (f_1, in kg), minimizing deceleration during the full-frontal crash (f_2, in m/s) and minimizing toe-board intrusion in the offset-frontal crash (f_3, in m). The thickness of five components of the frontal structure of the vehicle have been chosen as the decision variables. A mathematical formulation of the multiobjective optimization problem can be found in Appendix A (see [26] for more details).

4.2 Interactive solution process

Next, we describe the interactive solution process using the O-NAUTILUS method. Before involving the DM, 100 sampling points (P) were generated using the latin hypercube sampling (LHS) technique based on the objective functions. The default number of steps for the navigation was set as 100 and the default rate of navigation as 10 steps per second. First, Kriging models were trained with the available sampling points. We then evaluated the optimistic front (P^+) by using RVEA on the trained Kriging models, as described in Sect. 3.1.

The combined set utopian and nadir points were calculated from P and P^+ as (1661.58, 6.46, 0.058) and (1693.61, 11.28, 0.23), respectively. Initially, the known and optimistic reachable ranges were calculated, as described in Sect. 3.2. The reachable ranges were then shown to the DM in a graphical user interface.

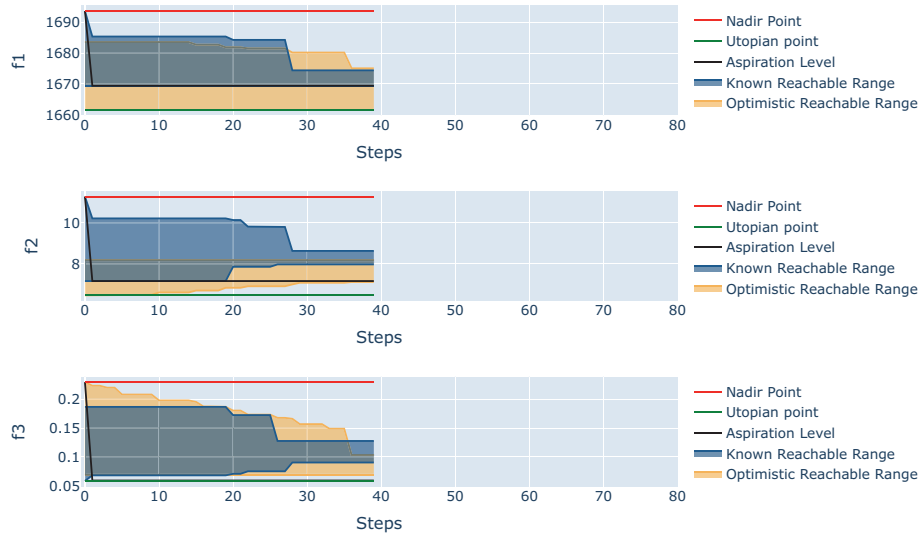


Fig. 7 Navigation view until the first change in the preference information

First, the DM wanted to provide aspiration levels for each objective based on the lower bounds of the known reachable range. As these values seemed very promising, the DM set the aspiration levels 1669.39 kg for f_1, 7.16 m/s for f_2 and 0.058 m for f_3, close to the combined set utopian point. Then, the navigation process was started to see whether these values were achievable or not. The DM saw real-time movement, in the sense that the reachable region started shrinking. Figure 7 shows the navigator view in the user interface of O-NAUTILUS. In this view, the combined set utopian and nadir point is shown by green and red lines, respectively. Also, the aspiration level provided by the DM is shown by the black line. As the navigation continues, the known and the optimistic reachable ranges are shown in blue (darker) and orange (lighter) areas, respectively. As can be seen in Fig. 7, the DM stopped the navigation since the aspiration levels for f_2 and f_3 became unachievable based on the known reachable ranges (shown in blue (darker shade)). However, the optimistic reachable ranges (shown in orange (lighter shade)) indicated that solutions close to the aspiration levels may still be achievable. Therefore, the DM decided to evaluate a new point and provided aspiration levels for f_2 and f_3 based on the current lower bounds of the optimistic reachable ranges with the intention to find a solution in the orange (lighter) area. The new reference point was (1669.39, 7.09, 0.07) and one additional function evaluation was made by using this reference point and eASF, as described in Sect. 3.4. The newly evaluated point (1672.33, 7.30, 0.08) had better objective function values than the current lower bounds of the known reachable area and was added to the set of known solutions.

The navigation was restarted with the preference information which was provided for the additional function evaluation. After a few steps, the known reachable area started to shrink and as a consequence of the newly added point, the DM realized that the aspiration levels for f_2 and f_3 were near the new lower bounds of the known reachable area. Based on the optimistic reachable ranges, there could be solutions better in the first objective without sacrificing in others. Therefore, the DM stopped the navigation and wanted to make another function evaluation by providing a new reference point (1661.58, 7.09, 0.07). The newly

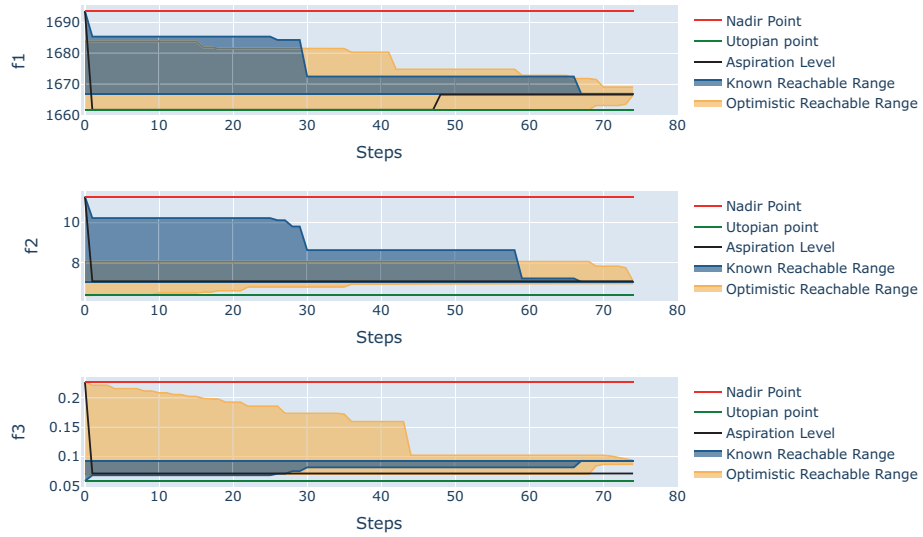


Fig. 8 Navigation view until the Pareto front is reached

evaluated point (1666.37, 7.05, 0.09) had better values than the current lower bound of the known reachable area for the first two objectives and was added to the set of known solutions.

The DM restarted the navigation with the newly added point. After a few steps of the navigation, the DM realized that the known reachable areas were shrinking and while the aspiration levels for f_2 and f_3 were achievable, the aspirations for f_1 was not. Then, he decided to pause the navigation in order to adjust his aspiration level for the first objective based on the lower bounds of known reachable area. He provided a new reference point (1666.60, 7.09, 0.07), which indicates a relaxation in the first objective. At this moment, he decided to let the navigation reach the Pareto front to see if his desires were achievable or not. The change in the aspiration levels and the navigation reaching the Pareto front are illustrated in Fig. 8.

As shown in Fig. 8, the reached solution reflected the desired values for the first and second objectives but not for the third objective. Because of this, the DM was not fully satisfied and realized that it may be possible to improve the third objective based on the final lower bound of the optimistic reachable ranges. Therefore, the DM wanted to conduct one more function evaluation by adjusting the aspiration level for the third objective according to the current optimistic front. He kept the same aspiration levels for the first two objectives. The new reference point for additional function evaluation was (1664.60, 7.09, 0.07). Then, the navigation was restarted with this reference point and the newly found solution added to the known set. The DM did not change his preference information in this last navigation since he had gained enough insight about the feasibility of his aspiration levels. Thus, he let the navigation reach a new final solution. As can be seen in Fig. 9, this time, the reached solution (1667.53, 7.22, 0.08) reflected the DM's preference information quite well, and the DM selected it as the most preferred solution. Table 1 summarizes the details of the interactive solution process. The table lists the values of combined set nadir and utopian point, lower and upper bounds of the known and optimistic reachable ranges, and the aspiration levels given for each objectives for each iterations. By iterations, we mean instances when the DM paused the navigation and provided preference information.

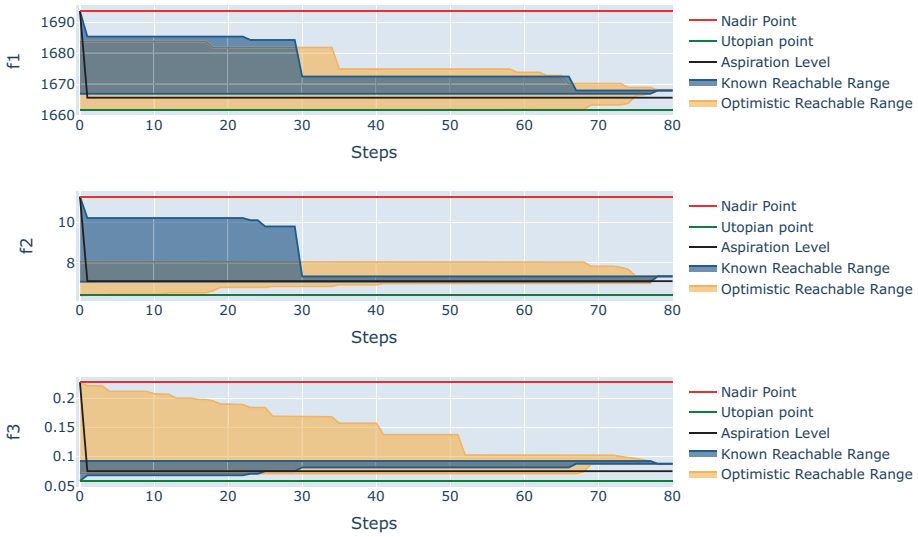


Fig. 9 Navigation view until a satisfactory solution was reached

4.3 Comparison with NAUTILUS navigator

To the best of our knowledge, no quality indicators for assessing and comparing the performance of interactive methods have been developed yet in the literature [1]. Therefore, we make the comparison by solving the same problem with NAUTILUS Navigator.

With O-NAUTILUS, the DM made only 3 additional exact objective function evaluations to reach the most preferred solution. As pointed out in the previous section, 100 function evaluations were made to generate LHS sampling data as an input for the O-NAUTILUS method. Therefore, a total of 103 function evaluations were conducted. Since in O-NAUTILUS, Kriging was used as a surrogate modeling technique and RVEA as an evolutionary algorithm, we used K-RVEA [6], an extension of RVEA which uses Kriging models, to generate the initial data set needed by NAUTILUS Navigator. K-RVEA generated 20 nondominated solutions with the same budget (103 exact function evaluations; 53 for LHS sampling data and 50 for K-RVEA). K-RVEA was run for the same number of generations as RVEA in the previous experiment.

With NAUTILUS Navigator, the DM provided similar preference information per iteration and eventually reached a satisfactory solution, which was (1668.49, 7.00362, 0.13276), in a similar manner. When this solution is compared with the one obtained by O-NAUTILUS, the DM reached a solution reflecting his preferences for the first two objectives but not for the last one. Therefore, we can say that he sacrificed more in the third objective in comparison to the final solution reached with O-NAUTILUS. Based on the DM's experience with both methods, for the sake of brevity we list the differences without numerical details below:

- Optimistic ranges of O-NAUTILUS supported the DM in providing preference information and finding a region of interest.
- Additional function evaluations of O-NAUTILUS supported the DM in finding and adding new solutions in the region of interest which remained unexplored in NAUTILUS Navigator.

Table 1 Summary of some data shown to and provided by the DM when the DM paused the navigation and provided aspiration levels

Iteration number	Objectives	Nadir point	Ideal point	Known reachable		Optimistic reachable		Aspiration levels
				Max	Min	Max	Min	
1	f_1	1693.61	1661.58	1693.61	1669.39	1682.59	1661.58	1669.39
	f_2	11.28762	6.46597	11.28762	7.16433	8.08408	6.46597	7.16433
	f_3	0.23015	0.05833	0.18663	0.05833	0.23015	0.07099	0.05833
2	f_1	1693.61	1661.58	1674.39	1669.39	1673.39	1661.58	1669.39
	f_2	11.28762	6.46595	8.63587	7.97711	8.08402	7.09049	7.09049
	f_3	0.23015	0.05833	0.12756	0.09021	0.14123	0.07099	0.07099
3	f_1	1693.61	1661.58	1672.34	1669.39	1675.29	1661.58	1661.58
	f_2	11.28762	6.43311	8.63587	7.97711	8.08625	7.00589	7.09049
	f_3	0.22751	0.05833	0.12756	0.09021	0.10212	0.07080	0.07099
4	f_1	1693.61	1661.58	1672.34	1669.39	1675.29	1661.58	1666.60
	f_2	11.28762	6.43311	8.63587	7.97711	8.08625	7.00589	7.09049
	f_3	0.22751	0.05833	0.12756	0.09021	0.10212	0.07080	0.07099
5	f_1	1693.61	1661.68	1666.60	1666.60	1668.76	1666.51	1666.60
	f_2	11.28762	6.39665	6.99398	6.99398	7.47180	6.98596	7.09049
	f_3	0.22684	0.05833	0.09337	0.09337	0.09397	0.08507	0.08507

- NAUTILUS Navigator needed optimized points to start the solution process. Because of this, there is a pre-processing stage to generate nondominated solutions by using some appropriate multiobjective optimization method, for instance an evolutionary algorithm. On the contrary, O-NAUTILUS can start with any initial data which is not optimized. This means that if the DM has some available data of function evaluations for her/his optimization problem, it can readily be utilized.
- In NAUTILUS Navigator, the satisfaction degree of the final solution was highly dependent on the nondominated solutions found in the pre-processing stage. However, in O-NAUTILUS, even if initial solutions are not good, the DM can reach very good solutions with the help of additional function evaluations during the solution process. They will be conducted in a focused manner based on her/his preference information.

The positive features of O-NAUTILUS have some trade-offs as well. Because of the optimistic ranges and ability to make additional function evaluations, the DM needed to digest more information at a time in O-NAUTILUS. In NAUTILUS Navigator, as the DM saw only the reachable ranges, and no additional function evaluations were made in the solution process, NAUTILUS Navigator was easier to use than the O-NAUTILUS method. Furthermore, in O-NAUTILUS, the DM needs to wait for the additional exact function evaluation during the solution process. In NAUTILUS Navigator, waiting is out of the question during the interactive phase since no additional function evaluations are conducted during the solution process.

To conclude the comparison, O-NAUTILUS effectively supported the DM by providing optimistic reachable ranges and enabling additional function evaluations in the region of interest of the DM. Even with randomly generated initial points, the method could achieve satisfactory solutions by using few exact function evaluations. However, one needs to digest more information in each iteration and wait for the function evaluations conducted in the solution process.

5 Conclusions

In this paper we have proposed a novel method in the NAUTILUS family, termed O-NAUTILUS for interactive multiobjective optimization. The overarching challenge was to propose a new version of the NAUTILUS family that can be used for optimization problems where expensive function evaluations are to be used sparingly. To meet this challenge, we had to design a new, general algorithmic framework by integrating surrogate models with uncertainty handling into NAUTILUS methods. The new methodology allows for an interactive mode of exploration with alternating phases of decision making and compute-intensive steps where new evaluations of expensive objective functions are conducted to acquire information about promising regions of the decision space.

We have introduced novel concepts to tackle problems with expensive function evaluations. These new concepts have been implemented as an interactive O-NAUTILUS algorithm supported by a graphical user interface. With this user interface, the decision maker can see in real time how the preferences provided affect the direction of the search in terms of reachable ranges evolving. The usefulness of the new concepts is demonstrated as follows:

- We developed O-NAUTILUS as an extension of NAUTILUS Navigator by showing optimistic ranges to the decision maker. This allows the decision maker to schedule additional exact function evaluations to explore regions of the objective space where

(s)he can still expect (based on optimistic bounds) further improvements. This is useful for optimization with expensive function evaluations.
- We created an open source implementation of O-NAUTILUS in Python which can be accessed at https://desdeo.it.jyu.fi.
- We demonstrated the implementation on an instance from the domain of crashworthiness design, which is a typical context where expensive function evaluations occur in practice.
- We augmented the graphical representation of the reachable ranges plot to accommodate optimistic ranges.
- We made several detailed decisions when developing the ideas into a workable algorithm: the type of user interaction and preference model and the choice of surrogate modeling and uncertainty handling techniques. Kriging models were used but also other surrogate models providing uncertainty quantification can be utilized. As an example of alternative models, we mentioned Lipschitzian models.

In this contribution, we have introduced surrogate modeling techniques with uncertainty handling for the first time in the context of NAUTILUS methods. Surrogate modeling is in itself a very active field of research which has recently brought forward many results. Future work should therefore include the choice and update of the surrogate modeling techniques. One should note that although Kriging and Lipschitzian models are utilized in this paper to approximate the optimistic Pareto front, the ideas are not limited to these two surrogate model types and one can also use other surrogate model types which feature uncertainty quantification. Understanding when to apply Kriging and when Lipschitzian models, or other types of surrogate models deserves further analysis. Further experiments are also to be conducted with O-NAUTILUS, especially with real-life problems.

Acknowledgements This research was partly funded by the Academy of Finland (Grants 322221 and 311877). The research is related to the thematic research area Decision Analytics utilizing Causal Models and Multiobjective Optimization (DEMO), jyu.fi/demo, at the University of Jyväskylä.

A Mathematical formulation of the multiobjective optimization problem for the crash-worthiness design of vehicle

Following [26], the multiobjective optimization problem for the crash safety design of vehicles is formulated as follows:

$$\begin{aligned} &\text{minimize } F(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})\} \\ &\text{subject to } 1 \leq x_j \leq 3 \qquad\qquad j = 1, \ldots, 5, \end{aligned} \tag{11}$$

where f_i $(i = 1, 2, 3)$ representing the relevant surrogate models, formed based on the data collected from simulation models as described in [26], have the following formulations:

$$\begin{aligned}
f_1(\mathbf{x}) &= 1640.2823 + 2.3573285x_1 + 2.3220035x_2 + 4.5688768x_3 \\
&\quad + 7.7213633x_4 + 4.4559504x_5 \\
f_2(\mathbf{x}) &= 6.5856 + 1.15x_1 - 1.0427x_2 + 0.9738x_3 + 0.8364x_4 \\
&\quad - 0.3695x_1x_4 + 0.0861x_1x_5 + 0.3628x_2x_4 \\
&\quad - 0.1106x_1^2 - 0.3437x_3^2 + 0.1764x_4^2 \\
f_3(\mathbf{x}) &= -0.0551 + 0.0181x_1 + 0.1024x_2 + 0.0421x_3 - 0.0073x_1x_2 \\
&\quad + 0.024x_2x_3 - 0.0118x_2x_4 - 0.0204x_3x_4 - 0.008x_3x_5 \\
&\quad - 0.0241x_2^2 + 0.0109x_4^2
\end{aligned}$$

Here, the objective functions f_1, f_2, and f_3 are the vehicle mass, deceleration during the full-frontal crash, and toe-board intrusion in the offset-frontal crash, respectively. The decision variable x_j $(j = 1, \ldots, 5)$ represents the thickness of a component of the frontal structure of the vehicle. See [26] for more details about the surrogate construction and problem formulation.

References

1. Afsar, B., Miettinen, K., Ruiz, F.: Assessing the performance of interactive multiobjective optimization methods: a survey. ACM Comput. Surv. **54**(4), 85 (2021)
2. Audet, C.: A Survey on Direct Search Methods for Blackbox Optimization and Their Applications. In: Pardalos, P.M., Rassias, T.M. (eds.) Mathematics without boundaries, vol. 2, pp. 31–56. Springer, Berlin (2014)
3. Aytuğ, H., Sayın, S.: Using support vector machines to learn the efficient set in multiple objective discrete optimization. Eur. J. Oper. Res. **193**(2), 510–519 (2009)
4. Buchanan, J.T., Corner, J.: The effects of anchoring in interactive MCDM solution methods. Comput. Oper. Res. **24**(10), 907–918 (1997)
5. 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)
6. 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)
7. Chugh, T., Sindhya, K., Hakanen, J., Miettinen, K.: A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. Soft Comput. **23**(9), 3137–3166 (2019)
8. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms. Wiley (2001)
9. Emmerich, M.: Single-and multi-objective evolutionary design optimization assisted by Gaussian random field metamodels. PhD dissertation, University of Dortmund (2005)
10. Emmerich, M., Yang, K., Deutz, A., Wang, H., Fonseca, C.M.: A Multicriteria Generalization of Bayesian Global Optimization. In: Pardalos, P.M., Zhigljavsky, A., Žilinskas, J. (eds.) Advances in stochastic and deterministic global optimization, pp. 229–242. Springer, Berlin (2016)
11. Floudas, C.A., Pardalos, P.M. (eds.): Encyclopedia of Optimization. Springer, Berlin (2008)
12. Gaudrie, D., Le Riche, R., Picheny, V., Enaux, B., Herbert, V.: Targeting solutions in Bayesian multi-objective optimization: sequential and batch versions. Ann. Math. Artif. Intell. **88**(1), 187–212 (2020)
13. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
14. Hartikainen, M., Miettinen, K., Klamroth, K.: Interactive nonconvex pareto navigator for multiobjective optimization. Eur. J. Oper. Res. **275**(1), 238–251 (2019)
15. Hastings, W.D.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**(1), 97–109 (1970)
16. Horst, R., Pardalos, P.M. (eds.): Handbook of Global Optimization. Springer, Berlin (1995)

17. Husain, A., Kim, K.-Y.: Enhanced multi-objective optimization of a microchannel heat sink through evolutionary algorithm coupled with multiple surrogate models. Appl. Therm. Eng. **30**(13), 1683–1691 (2010)
18. 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 (2019)
19. Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Glob. Opt. **21**(4), 345–383 (2001)
20. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Opt. **13**(4), 455–492 (1998)
21. Kahneman, D., Tversky, A.: Prospect theory: an analysis of decisions under risk. Econometrica **47**, 313–327 (1979)
22. Knowles, J.: ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evol. Comput. **10**(1), 50–66 (2006)
23. Kourakos, G., Mantoglou, A.: Development of a multi-objective optimization algorithm using surrogate models for coastal aquifer management. J. Hydrol. **479**, 13–23 (2013)
24. Kvasov, D.E., Sergeyev, Y.D.: Lipschitz global optimization methods in control problems. Autom. Remote Control **74**(9), 1435–1448 (2013)
25. Li, M., Li, G., Azarm, S.: A Kriging metamodel assisted multi-objective genetic algorithm for design optimization. J. Mech. Des. **130**(3), 031401 (2008)
26. Liao, X., Li, Q., Yang, X., Zhang, W., Li, W.: Multiobjective optimization for crash safety design of vehicles using stepwise regression model. Struct. Multidiscipl. Opt. **35**(6), 561–569 (2008)
27. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)
28. Miettinen, K., Eskelinen, P., Ruiz, F., Luque, M.: NAUTILUS method: an interactive technique in multiobjective optimization based on the nadir point. Eur. J. Oper. Res. **206**(2), 426–434 (2010)
29. Miettinen, K., Hakanen, J., Podkopaev, D.: Interactive Nonlinear Multiobjective Optimization Methods. In: Greco, S., Ehrgott, M., Figueira, J. (eds.) Multiple criteria decision analysis: state of the art surveys, 2nd edn., pp. 931–980. Springer, Berlin (2016)
30. Miettinen, K., Ruiz, F.: NAUTILUS framework: towards trade-off-free interaction in multiobjective optimization. J. Bus. Econ. **86**(1–2), 5–21 (2016)
31. Miettinen, K., Ruiz, F., Wierzbicki, A.P.: Introduction to Multiobjective Optimization: Interactive Approaches. In: Branke, J., Deb, K., Miettinen, K., Slowinski, R. (eds.) Multiobjective Optimization: Interative and Evolutionary Approaches, pp. 27–57. Springer, Berlin (2008)
32. Mitra, K., Majumder, S.: Successive approximate model based multi-objective optimization for an industrial straight grate iron ore induration process using evolutionary algorithm. Chem. Eng. Sci. **66**(15), 3471–3481 (2011)
33. Mockus, J., Tiesis, V., Zilinskas, A.: The application of Bayesian methods for seeking the extremum. Towards Glob. Opt. **2**(117–129), 2 (1978)
34. Paulavičius, R., Žilinskas, J., Grothey, A.: Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds. Opt. Lett. **4**(2), 173–183 (2010)
35. Pintér, J.D.: Global Optimization in Action: Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications. Springer, Berlin (2013)
36. Qasem, S.N., Shamsuddin, S.M., Hashim, S.Z.M., Darus, M., Al-Shammari, E.: Memetic multiobjective particle swarm optimization-based radial basis function network for classification problems. Inf. Sci. **239**, 165–190 (2013)
37. Regis, R.G., Shoemaker, C.A.: Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. Eng. Opt. **45**(5), 529–555 (2013)
38. Ruiz, A.B., Ruiz, F., Miettinen, K., Delgado-Antequera, L., Ojalehto, V.: NAUTILUS Navigator: free search interactive multiobjective optimization without trading-off. J. Glob. Opt. **74**(2), 213–231 (2019)
39. Sergeyev, Y.D., Kvasov, D.E.: Deterministic Global Optimization: An Introduction to the Diagonal Approach. Springer, Berlin (2017)
40. Sergeyev, Y.D., Kvasov, D.E., Mukhametzhanov, M.S.: On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. Sci. Rep. **8**(453), 1–9 (2018)
41. Stein, M.L.: Interpolation of Spatial Data: Some Theory for Kriging. Springer, New York, NY (1999)
42. Strongin, R.G., Sergeyev, Y.D.: Global Optimization with Non-convex Constraints: Sequential and Parallel Algorithms. Springer, Berlin (2013)
43. Tabatabaei, M., Hakanen, J., Hartikainen, M., Miettinen, K., Sindhya, K.: A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods. Struct. Multidiscip. Opt. **52**(1), 1–25 (2015)
44. Van Beers, W.C.M., Kleijnen, J.P.C.: Kriging for interpolation in random simulation. J. Oper. Res. Soc. **54**(3), 255–262 (2003)

45. Wierzbicki, A.P.: On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spektrum **8**, 73–87 (1986)
46. Zhigljavsky, A., Žilinskas, A.: Stochastic Global Optimization, vol. 9. Springer Science & Business Media, Berlin (2007)
47. Žilinskas, A.: Visualization of a statistical approximation of the Pareto front. Appl. Math. Comput. **271**, 694–700 (2015)
48. Žilinskas, A., Žilinskas, J.: Adaptation of a one-step worst-case optimal univariate algorithm of bi-objective Lipschitz optimization to multidimensional problems. Commun. Nonlinear Sci. Numer. Simul. **21**(1–3), 89–98 (2015)

PV

DESDEO: THE MODULAR AND OPEN SOURCE FRAMEWORK FOR INTERACTIVE MULTIOBJECTIVE OPTIMIZATION

by

Giovanni Misitano, Bhupinder Singh Saini, Bekir Afsar, Babooshka Shavazipour, Kaisa Miettinen 2021

IEEE Access, 9, 148277 – 148295

https://doi.org/10.1109/ACCESS.2021.3123825

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 1, 2021, accepted October 20, 2021, date of publication October 27, 2021, date of current version November 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3123825

DESDEO: The Modular and Open Source Framework for Interactive Multiobjective Optimization

G. MISITANO, B. S. SAINI, B. AFSAR, B. SHAVAZIPOUR, AND K. MIETTINEN
Faculty of Information Technology, University of Jyväskylä, 40014 Jyväskylä, Finland
Corresponding author: G. Misitano (giovanni.a.misitano@jyu.fi)

This work was supported by the Academy of Finland under Grant 322221.

ABSTRACT Interactive multiobjective optimization methods incorporate preferences from a human decision maker in the optimization process iteratively. This allows the decision maker to focus on a subset of solutions, learn about the underlying trade-offs among the conflicting objective functions in the problem and adjust preferences during the solution process. Incorporating preference information allows computing only solutions that are interesting to the decision maker, decreasing computation time significantly. Thus, interactive methods have many strengths making them viable for various applications. However, there is a lack of existing software frameworks to apply and experiment with interactive methods. We fill a gap in the optimization software available and introduce DESDEO, a modular and open source Python framework for interactive multiobjective optimization. DESDEO's modular structure enables implementing new interactive methods and reusing previously implemented ones and their functionalities. Both scalarization-based and evolutionary methods are supported, and DESDEO allows hybridizing interactive methods of both types in novel ways and enables even switching the method during the solution process. Moreover, DESDEO also supports defining multiobjective optimization problems of different kinds, such as data-driven or simulation-based problems. We discuss DESDEO's modular structure in detail and demonstrate its capabilities in four carefully chosen use cases aimed at helping readers unfamiliar with DESDEO get started using it. We also give an example on how DESDEO can be extended with a graphical user interface. Overall, DESDEO offers a much-needed toolbox for researchers and practitioners to efficiently develop and apply interactive methods in new ways – both in academia and industry.

INDEX TERMS Data-driven multiobjective optimization, evolutionary computation, interactive methods, multi-criteria decision making, nonlinear optimization, open source software, Pareto optimization.

I. INTRODUCTION

Optimization in many real-life problems is typically characterized by several conflicting objectives to be considered simultaneously. In these multiobjective optimization problems, the presence of conflicting objectives results in many so-called Pareto optimal solutions with different trade-offs instead of a single optimal solution. These solutions are incomparable without additional information. Therefore, there is a need for a domain expert, referred to as a decision maker (DM), to ultimately choose one of the Pareto optimal solutions as the final one based on his/her preferences.

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

Different types of methods have been developed for solving multiobjective optimization problems in the multiple criteria decision making (MCDM) (e.g., [1]–[3]) and evolutionary multiobjective optimization (EMO) (e.g., [4], [5]) communities. Most MCDM methods incorporate a DM's preferences to focus on subsets of the Pareto optimal solutions reflecting the interests of the DM. These methods have a strong theoretical background and can guarantee Pareto optimality (see, e.g., [3]). Most MCDM methods use so-called scalarization or scalarizing functions to transform the original multiobjective optimization problem with the preference information into a scalarized problem (with a single objective) to be optimized. After this transformation, an appropriate single-objective optimization method is to be used to solve the scalarized problem. By carefully selecting the scalarizing

function, one can guarantee getting a Pareto optimal solution for the original problem so that the DM's preferences are considered. With different preferences, one can typically get different Pareto optimal solutions. For comparisons of different scalarizing functions, see, e.g., [6], [7]. In contrast, EMO methods handle a population of solutions at a time and generate several approximated Pareto optimal solutions to represent different Pareto optimal solutions. They often start from a random set of solutions and use different selection, mutation and recombination operators to create the next generation of solutions. Because of their heuristic nature, they cannot guarantee Pareto optimality, but they can be applied to challenging problems with, e.g., discontinuous or nonconvex functions.

One can classify different multiobjective optimization methods based on when a DM with preference information takes part in the solution process [1], [3]. A *no preference method* is applied in absence of preferences. The DM may provide his/her preferences before or after the solution process in *a priori* or *a posteriori methods*, respectively. In a priori methods, the DM provides hopes and expectations, and the method tries to find the best matching solution. In contrast, a representative set of Pareto optimal solutions is generated in a posteriori methods for the DM to choose from.

The fourth class of methods, known as interactive multiobjective optimization methods, involves the DM during the solution process. In this way, the DM iteratively provides his/her preferences while gradually gaining further insight into the problem and learning about hidden limitations such as the feasibility of the preferences and attainable solutions [8]. Therefore, the DM has a chance to modify his/her preferences based on new insight and learning. Moreover, the cognitive load set on the DM (at a time) is usually low compared to other methods, e.g., a posteriori methods. Indeed, the DM can focus the search on a subset of solutions and only consider Pareto optimal solutions of interest. This also saves computational resources. Because of these reasons, we consider here interactive methods. As mentioned, they consist of iterations. At each iteration, the DM sees a solution or some solutions reflecting the provided preferences and can adjust the preferences to eventually find the most preferred solution. Thanks to learning, the confidence of the DM grows during the solution process.

The DM can provide various types of preference information. Examples of them include so-called reference points whose components represent desired values for objective functions (also called aspiration levels), ranges for acceptable objective function values, classification, pairwise comparisons and selecting desired or undesired solutions out of a subset, to name a few (see, e.g., [3], [9]).

Over the years, different interactive methods have been developed in the literature, and they have shown their potential in various applications, see, e.g., [10]. They differ from each other mainly in terms of preference information used, how solutions reflecting preferences are generated, and what kind of information is provided to the DM [3], [8], [11]. However, their implementations are done in isolation, and they are not readily available. Even though most interactive methods utilize similar components (such as types of preference information, scalarizing functions, sampling techniques), each method has a different way of implementation. These issues slow down the practical usage of interactive methods from different perspectives and introduces various challenges, which we have listed as follows:

1) It is not easy to find implementations of different interactive methods to be applied.
2) Identifying the most suitable interactive methods to be used in various real-life applications is challenging.
3) Comparing interactive methods is difficult because of the lack of having various interactive methods within the same framework.
4) Utilizing the implemented methods or some parts of their implementation in new developments is hard, so every new construction needs to be started from scratch.
5) The lack of openness limits applicability.
6) The iterative nature of the interactive methods, together with some standard components, enables switching between methods in different iterations of the solution process, at least in theory. Nonetheless, separate implementations have been preventing the chance of testing this exciting idea.

To the best of our knowledge, only one framework has been developed for interactive methods, which, to some extent, aims to address the listed issues. It is the so-called DESDEO framework [12]. However, the version discussed in [12] had practical issues in its implementation, overall structure, and modularity and was, thus, not ready for broader usage and extensions. For these reasons, there was a need to first re-structure and then to re-implement a new DESDEO framework, which is introduced in this paper. The new framework has a clear potential in addressing all the six listed challenges.

The new DESDEO framework implemented in Python [13] has a modular structure and , thus, involves reusable modules that can be utilized for implementing new interactive methods or modifying the existing ones. DESDEO enables solving computationally expensive simulation-based and data-driven problems using surrogate models, including uncertainty considerations. It contains implementations of several old and new interactive methods by various developers covering methods of both MCDM and EMO types. Thanks to the modular structure, new or revised methods can be conveniently included in the framework.

DESDEO consists of packages and modules. We introduce them and also demonstrate how DESDEO can be applied to solve problems with analytical expressions as well as data-driven and simulation based problems. The strengths of DESDEO include the option to hybridize scalarization based and evolutionary methods and the convenience of comparing different methods in the same environment. For instance,

there is no need to specify the problem to be solved for each method separately.

The modular structure enables hybridization between different types of methods. By hybridization, we mean the ability to use final or intermediate results of one method in another method, such as generating approximated Pareto optimal solutions utilizing an EMO method and using the solutions in an MCDM method or switching the method during the solution process, e.g., when the DM wants to change the type of preference information. This opens up new opportunities for utilizing different features of various methods while the DM is not limited to using only one method or one type of preferences. Various visualizations and a graphical user interface are also being developed with a similar modular structure in mind. The methods implemented in DESDEO can be utilized by anyone who has basic programming skills in Python, which has become a widely-used programming language in data and business analytics. Since the framework is open source, it is readily available for various applications and can be conveniently tailored for different problems, if needed. It is naturally also open to new contributions and anybody interested is welcome to contribute.

The rest of the paper is structured as follows. In Section II , we outline the general concepts and notations of multiobjective optimization that we use in this paper, briefly review the related open source frameworks for multiobjective optimization in the literature, and overview some interactive methods (referred to in this paper). Section III is targeted at readers interested in contributing to the development of DESDEO. For this, the framework structure introducing packages, modules, and external dependencies is described in detail. Those who only wish to apply the framework for solving multiobjective optimization problems can skip Section III and focus on four diverse illustrative use cases outlined in Section IV. In Section IV, we also give a basic example of a graphical user interface that can be implemented to ease interaction between the interactive methods in DESDEO and the DM. In Section V, we discuss the potential of the modular framework, such as adjusting or hybridizing methods and creating user interfaces for various interactive methods. Finally, we conclude in Section VI.

II. BACKGROUND

In this section, we first introduce the main notation and concepts used in this paper. We then survey the state-of-the-art of open source software frameworks available for multiobjective optimization. Finally, we very briefly outline some of the interactive methods referred to in the use cases considered in Section IV.

A. MULTIOBJECTIVE OPTIMIZATION

We consider the following form of multiobjective optimization problems minimizing $k \geq 2$ objective functions [3]:

$$\begin{aligned} \min \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \\ \text{s.t.} \quad & \mathbf{x} \in S, \end{aligned} \tag{1}$$

where $f_i \colon S \to \mathbb{R}$ $(i = 1, \ldots, k)$ are objective functions and $\mathbf{x} = (x_1, \ldots, x_n)^T$ is a vector of n decision variables in the feasible region $S \subset \mathbf{R}^n$ defined by constraint functions. Without loss of generality, we here assume that all functions are to be minimized. If some function f_i is to be maximized, it is equivalent to minimize $-f_i$.

A decision (variable) vector $\mathbf{x}^* \in S$ is called Pareto optimal if there exists no $\mathbf{x} \in S$, so that for all $i, f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ and for some $j, f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$. The image of Pareto optimal decision vectors in the objective space $\mathbf{R}^k$ is called a Pareto front and it consists of Pareto optimal objective vectors. In the definition of Pareto optimality, a solution is not dominated by any other feasible solution. As we deal with evolutionary methods that cannot guarantee Pareto optimality, we also use the term nondominated solutions. They are not dominated by any solution in the solution set considered (typically referred to as a population), but are not necessarily Pareto optimal.

The best and the worst possible values of objective functions in the Pareto front are represented by an ideal and a nadir point, respectively. The components of the ideal point can be calculated by optimizing each objective function subject to S as a single-objective optimization problem. In contrast, computing the nadir point is difficult in practice as the set of all Pareto optimal solutions is unknown. However, some methods (e.g., a payoff table [14]) are available that can approximate the nadir point (see e.g., [3] and references therein).

B. DATA-DRIVEN MULTIOBJECTIVE OPTIMIZATION

As mentioned in the introduction, DESDEO can be applied to solve different types of multiobjective optimization problems. Typically, the analytical forms of objective functions and constraints cannot be formulated in most real-life problems. In some cases, simulation models can be used to evaluate function values. In other cases, the objective or constraints values must be gained from some real experiences (or laboratory experiments). In either case, evaluating the function values is usually expensive from different perspectives. Therefore, so-called surrogate models can be utilized instead of the original expensive models or experiments.

On the other hand, in today's digital societies, various data from different sources are continuously recorded, which can be used as a new source of information in decision making. Making the most of the data available can lead to data-driven optimization problems. In this case, no other information than the data is available, giving no other option than fitting surrogate models to formulate functions for optimization problems based on the data. Then, surrogate models approximate the objective or constraint values.

Different types of surrogate models, such as probabilistic (e.g., Bayesian network [15] and Markov chain Monte Carlo) or machine learning techniques (e.g., radial basis functions [16], Kriging or Gaussian processes [17], [18], support vector regression [19], and neural networks [20], [21]) exist and can be utilized to derive functions for multiobjective optimization problems. Most of these techniques are freely

available in different Python packages and libraries, which can be used within the DESDEO framework.

C. LITERATURE REVIEW ON OPEN SOURCE FRAMEWORKS FOR MULTIOBJECTIVE OPTIMIZATION

We have surveyed open source frameworks for multiobjective optimization problems. We do not consider closed source and commercial software implementations because they do not provide an opportunity to adjust the methods to one's needs in the way open source software does. Several open source software frameworks have been proposed in the literature. Each of them has its own strengths and limitations and differs in some nuances from the others. In general, many aspects should be considered when selecting an appropriate framework for one's needs. For example, familiarity with the programming language used to implement the framework, the characteristics of the problem to be solved, the availability of visualization tools, and an exemplary user interface can influence the selection of a framework.

Table 1 summarizes well-known open source frameworks proposed for solving multiobjective optimization problems. We also list some common frameworks with a modular structure, where multiobjective optimization methods can be created. Besides the name and the programming language used, the table lists whether the frameworks focus on multiobjective optimization, include MCDM or EMO types of methods, provide a decision-making mechanism where a DM can provide his/her preference information and choose the most preferred solution, visualization tools, and a user interface. The table also states whether the framework has a modular structure or not. In the following, we briefly describe each of the frameworks.

DEAP [22] and Inspyred [23] do not focus specifically on multiobjective optimization but provide Python implementations of e.g., genetic algorithms, simulated annealing, and differential evolution. The (a posteriori) EMO method NSGA-II for multiobjective optimization is also included. Since these two frameworks have been developed with a modular structure, more multiobjective optimization methods can be developed by using the modules available in the framework. Inspyred includes further nature-inspired optimization algorithms such as particle swarm optimization and ant colony optimization.

vOptSolver [24] has been implemented in the Julia language. It integrates several exact algorithms for multiobjective linear optimization problems (including mixed-integer problems).

Platypus [25] involves Python implementations of several well-known EMO methods concentrating, thus, on multiobjective optimization. It also includes an analysis tool for visual comparison of EMO methods by applying some performance indicators.

MOEA [26] is a Java-based framework that enables automatic parallelization of methods across multiple processor cores. It includes most of the state-of-the-art a posteriori EMO methods.

PyGMO [27] is a Python extension of PaGMO (C++) [28] which has implementations of a variety of single- and multiobjective optimization methods and real-life engineering problems in an object-oriented architecture. Automatic parallelization of the implemented methods enables using the underlying multicore architecture efficiently.

jMetalPy [29] extends the Java-based framework jMetal [30] (which contains metaheuristic methods like evolutionary methods) for multiobjective optimization to be used in Python. jMetalPy provides improved data analysis, interactive visualization of Pareto optimal solutions, and increased computational performance by applying libraries available in Python. Additionally, jMetalPy facilitates parallel computing for computationally expensive problems.

Pymoo [31] is a multiobjective optimization framework in Python and offers evolutionary methods for single- and multiobjective optimization problems. It involves several visualization techniques for illustrating results and well-known indicators to compare the performance of the methods.

Finally, PlatEMO [32] is an open source framework developed in MATLAB including many EMO methods, widely used performance indicators, and benchmark problems. It also has a graphical user interface. However, one should note that even though the implementation is openly available, a MATLAB license is required to use it. Therefore, while being commercial software, PlatEMO still allows adjusting its implementation to meet specific needs.

The frameworks mentioned so far do not contain interactive methods. They include either MCDM or EMO types of methods, but not both, and only one of the frameworks comes with a user interface. As this summary shows, overall, DESDEO is unique since it is the only open source framework including interactive methods. Thus, DESDEO fills a gap in the software available in the multiobjective optimization community. DESDEO has a clear modular structure making it easy for users and developers to contribute new contents. Importantly, DESDEO involves both MCDM and EMO types of methods, enabling hybridizing and switching between methods depending on needs and application areas. Moreover, elements for building custom graphical user interfaces for efficient interaction between the DM and interactive methods are a planned future inclusion in DESDEO. These elements are currently under active development and are to be included as additional packages in DESDEO eventually. Therefore, visualization and user interface (UI) items for DESDEO are in parentheses in Table 1 for the time being. However, specialized non-modular graphical user interfaces have been developed for DESDEO in the past as seen in Section IV-F.

D. SOME INTERACTIVE METHODS IMPLEMENTED

As mentioned earlier, different interactive multiobjective optimization methods have been implemented in DESDEO. In this section, we briefly introduce a few that are utilized

TABLE 1. Summary of open source optimization frameworks. In the table, MO stands for multiobjective optimization.

Name	Programming language	MO focus	Method type		Decision making	Interactive methods	Visualization	UI	Modularity
			EMO	MCDM					
ine DEAP	*Python*								✓
Inspyred	*Python*								✓
vOptSolver	*Julia*	✓		✓					
Platypus	*Python*	✓	✓				✓		✓
MOEA	*Java*	✓	✓				✓		✓
PaGMO/PyGMO	*C++/Python*	✓	✓				✓		✓
jMetal/jMetalPy	*Java/Python*	✓	✓		✓		✓		✓
Pymoo	*Python*	✓	✓		✓		✓		✓
PlatEMO	*Matlab*	✓	✓		✓		✓	✓	✓
DESDEO	*Python*	✓	✓	✓	✓	✓	(✓)	(✓)	✓

later in Section IV: the reference point method [33], the synchronous NIMBUS method [34] and the NAUTILUS family [35] (particularly E-NAUTILUS [36]) from MCDM methods, and RVEA [37] and NSGA-III [38] from EMO methods.

The reference point method [33] is a popular interactive multiobjective optimization method in which the DM provides preferences as desired objective function values constituting a reference point. Then, at each iteration, $k + 1$ Pareto optimal solutions reflecting the reference point are found by utilizing an achievement scalarizing function. The DM can iterate (i.e., compare solutions and provide new reference points) until the most preferred solution is found.

In NIMBUS, starting from a Pareto optimal solution, a DM expresses his/her preferences by classifying the objective functions corresponding to the Pareto optimal solution into up to five preference classes to indicate how the current objectives should change to be more preferable to the DM. In each iteration of NIMBUS, based on the DM's preferences, 1–4 Pareto optimal solutions are generated and shown to the DM (the DM decides how many new solutions (s)he wants to see). Besides classification, the DM can ask for the desired number of solutions generated between any two Pareto optimal ones. Like other interactive methods, the solution process continues until the DM has found his/her most preferred solution.

The NAUTILUS family [35] contains interactive trade-off-free methods. This means that the DM does not deal with Pareto optimal solutions but gradually approaches the Pareto front starting from an inferior solution (like a nadir point). Then, following the DM's preferences, all objectives are simultaneously improved until a Pareto optimal solution is reached. During the solution process, the ranges of objective function values that still can be reached without trading-off naturally shrink. Once a Pareto optimal solution is reached, the solution process stops since it is no longer possible to proceed without trading-off. NAUTILUS variants vary regarding the types of preference information used and how solutions are generated in each iteration (see [35] for a comparison of the differences). For example, in each iteration of the original NAUTILUS [39] method, the DM ranks the objective functions based on the preferred improvement of the current objective values. In contrast, in NAUTILUS 2 [40], ratios of improvement are provided by the DM. In E-NAUTILUS [36], which is particularly developed for handling computationally expensive problems, the DM can compare multiple solutions (referred to as *intermediate points*) at each iteration. Finally, NAUTILUS Navigator [41] integrates NAUTILUS with navigation ideas [42], where the DM sees ranges of objective function values that are still reachable from the current iteration point shrinking in real-time and provides preferences as desired aspiration levels and bounds not to be exceeded.

Besides MCDM type of methods, various interactive EMO methods have also been developed and implemented in DESDEO. They include interactive versions [43] of the reference vector-guided evolutionary algorithm (RVEA) [37] and NSGA-III [38]. RVEA and NSGA-III are originally a posteriori methods. The interactive version of NSGA-III has been implemented, corresponding to how RVEA was made interactive in [43]. The main type of preference information used in both is a reference point, but other preference types are also available for RVEA.

III. STRUCTURE OF THE DESDEO FRAMEWORK

In this section, we describe the structure of the DESDEO framework, including packages of the framework and the modules in each package. In addition, we discuss the purpose of each package and its dependencies. We also consider the implementation of the DESDEO framework and its external dependencies. Lastly, we discuss the architectural choices made in DESDEO that any aspiring developer and user of the frameworks should be aware of. This section is intended mostly for those interested in contributing to the framework's development. Those interested only in utilizing the framework for solving multiobjective optimization problems may proceed to Section IV.

A. PACKAGES AND MODULES

In the modular structure of DESDEO, each package is a collection of modules, which contain class and function definitions to tackle specific tasks in modeling and solving multiobjective optimization problems interactively. The main

packages, called *core packages*, and their individual *modules* are presented in Figure 1. Each package has a well-defined purpose and is built to address a certain set of tasks in interactive multiobjective optimization methods. The modules may depend on other packages lower in the structure, as shown in Figure 2.

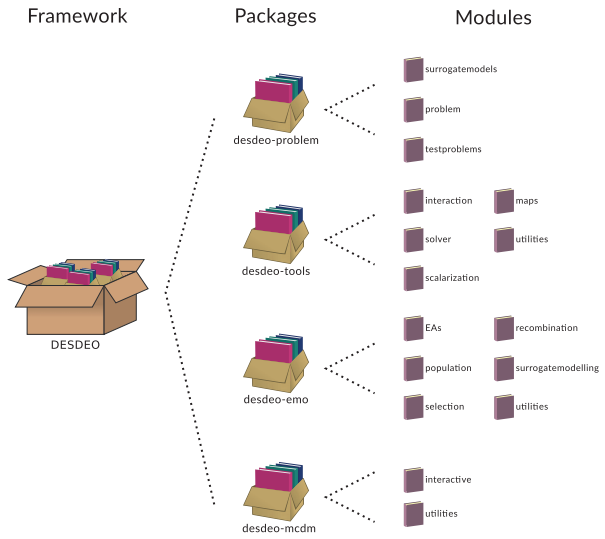


FIGURE 1. The main structure of the DESDEO framework with packages and modules included in each package. Further packages and modules can be added as needed.

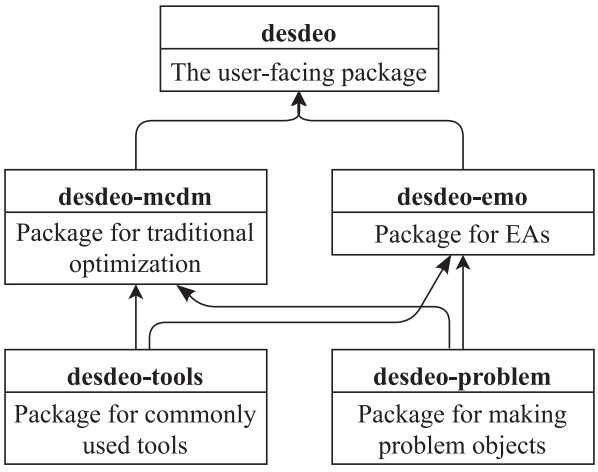


FIGURE 2. The packages of DESDEO and their dependencies on each other.

In Figure 2, the arrows represent the internal dependencies of packages in DESDEO, e.g., the package *desdeo-emo* depends on both the packages *desdeo-tools* and *desdeo-problem*. A modular structure allows users to choose which parts of the framework to use. For example, to model a multiobjective optimization problem, one can use the *desdeo-problem* package and avoid the needless inclusion of the other packages. Additionally, having the framework structured in a modular fashion eases the development of the framework by encapsulating features and functionalities related to interactive multiobjective optimization in their own respective packages.

In what follows, we describe packages included in DESDEO and their dependencies on other packages. The packages also depend on existing popular Python packages, which are discussed further at the end of this section.

The *desdeo-problem* package contains features related to the formulation and modeling of multiobjective optimization problems. Problems can be analytical expressions of functions depending on decision variables, or modeled based on collected data related to the multiobjective optimization problem (either utilizing data available or data obtained by running a problem-specific simulator). The problem can naturally also have constraint functions defining a feasible region. Tools for problem formulation can be found in the module *problem*. As already mentioned, surrogate models may be trained and used to model functions of a multiobjective optimization problem based on data. For example, Gaussian regression is available as a surrogate model but any other machine learning-focused package can be used to train surrogate models. The tools for building surrogates can be found in the *surrogatemodels* module. Moreover, commonly utilized test problems in multiobjective optimization can be found in the *testproblems* module. Such problems include, for example, the DTLZ problems [44]. The *desdeo-problem* package does not depend on any other package in the DESDEO framework.

The *desdeo-tools* package contains utility tools that are expected to be used during any phase of the optimization process, irrespective of the method type (MCDM or EMO) used for optimization. Such tools include abstractions for various preference elicitation techniques, scalarizing functions, and nondominated sorting. The *interaction* module contains methods to ease interaction between a DM and an interactive multiobjective optimization method. The *scalarization* module contains scalarization tools for transforming multiobjective optimization problems into single-objective problems (incorporating preference information). As mentioned in the introduction, we can get Pareto optimal solutions by using appropriate scalarization functions, such as achievement scalarizing functions [45] and the scalarization function of the ε-constraint method [2]. The *maps* module contains tools for transforming objectives from one space to another, such as e.g., the so-called preference incorporated space [46]. Finally, the *solver* module contains tools for solving scalarized problems. These solvers must be appropriate for the characteristics of the problem in question (considering, e.g., the type of variables and the nature of functions involved). The *desdeo-tools* package does not depend on any other package in the DESDEO framework.

The *desdeo-emo* package is the repository of evolutionary algorithms (EAs) and tools which are specifically used with EMO methods. Besides interactive EMO methods, it has

implementations of basic (a posteriori) EMO methods and some a priori methods as they can be used as elements of interactive ones. The package contains the following modules: *population*, *recombination*, *selection*, *EAs*, *surrogatemodelling*, and *utilities*. The first three modules contain abstractions representing the population, crossover and mutation operators as well as selection operators. We use these abstractions as building blocks to implement various evolutionary algorithms in the *EAs* module. New EAs can be implemented by either modifying the implementations in the *EAs* module or by using the building blocks in other modules in entirely new ways. The *surrogatemodelling* module implements certain EA based methods which are specifically designed to train surrogate models. Finally, *utilities* contains miscellaneous tools that are used by one or more EMO methods, but do not fit in the other modules. The *desdeo-emo* package depends on the *desdeo-problem* and *desdeo-tools* packages.

As the name suggests, the *desdeo-mcdm* package contains implementations of interactive multiobjective optimization methods of the MCDM type (involving scalarization functions to generate Pareto optimal solutions). The methods themselves are in the *interactive* module. For example, the synchronous NIMBUS and methods belonging to the NAUTILUS family are implemented in this module. The *utilities* module contains various utilities often needed in MCDM methods. For instance, the utilities include a payoff table method for computing an ideal and an approximation of the nadir point. The *desdeo-mcdm* package depends on the *desdeo-problem* and *desdeo-tools* packages.

Besides the core packages of DESDEO discussed so far, other packages can be, and have also been, developed based on the packages discussed. Examples of these packages consist of specialized graphical UIs and new experimental interactive multiobjective optimization methods not mature enough to be included in DESDEO yet. Due to their experimental nature, we will not discuss these additional packages further here.

As mentioned, the DESDEO framework has been implemented in Python and is available online as open source software on GitHub.[1] The framework makes use of existing Python libraries in the SciPy ecosystem, most notably NumPy [47], SciPy (the library) [48] and Pandas [49]. NumPy offers numerically efficient data structures, which enables an efficient handling of array-like structures present everywhere in the DESDEO framework. SciPy offers existing computational routines. For example, it offers excellent optimization routines for optimizing constrained problems with a single objective. As mentioned, this kind of problem emerges, for instance, when scalarizing a multiobjective optimization problem. In turn, Pandas has excellent and efficient data manipulation routines. They are needed especially when representing data-driven multiobjective optimization problems, which may sometimes consist of large amounts of data requiring extensive feature engineering before modeling a multiobjective optimization problem.

For a more detailed description of each package and module found in DESDEO, the reader is encouraged to check DESDEO's main documentation. The documentation is found online (https://desdeo.readthedocs.io/en/latest/) where the individual documentation of each core package can be found with additional details about implemented classes and functions.

B. ARCHITECTURAL DECISIONS IN DESDEO

A couple of choices have been made during the development of DESDEO. The user of the framework should keep them in mind while developing or using the framework.

As mentioned in Section II, objective functions in multiobjective optimization problems can either be minimized or maximized, but within the optimization methods in DESDEO, functions are always assumed to be minimized. This means that we convert functions to be maximized to functions to be minimized and internally only deal with minimization problems. This choice has been made to remove any possible software bugs, confusion, and guesswork related to keeping track whether an objective is to be minimized or maximized, transforming problems from one type to another, and parsing preference information. Naturally, when displaying information related to a multiobjective optimization problem and its solutions to a DM, the objectives are presented in their original form. The task of making the conversion whenever needed (also in the preference information) is the responsibility of the UI.

As interactive multiobjective optimization methods vary in the type of interaction and preference information required from the DM, abstraction of interaction has been kept simple and non-restrictive in DESDEO. Each interactive method has at least two (object) methods: `start` and `iterate`. As the name suggests, the former is always used to start a method after it has been instantiated. Likewise, the `iterate` method is then used for any subsequent interactions after starting the method. Both the `start` and `iterate` methods return at least one `request` (Python) object. These objects contain all the necessary information to carry out a required interaction with the interactive method in their `content` attribute, which is a Python dictionary. The contents of a `request` may vary depending on the interactive method, but each `content` dictionary in DESDEO comes at least with a `message` entry meant to give a hint to the user of what is expected of them interaction-wise. Each `request` object has a `response` attribute, which is also a dictionary. The `response` dictionary has its own entries, which the user must define to continue iterating the interactive method. After the entries of the `response` have been defined, the `iterate` method can be invoked by giving it the `request` containing the `response` with defined entries as an argument. The `iterate` method will then return a new `request`. Examples of this `request-response` structure can be found in the use cases in Section IV. However,

[1] https://github.com/industrial-optimization-group/DESDEO

it is not expected that a DM oneself would directly handle these Python dictionaries. Instead, it is expected that some external interface is used to facilitate interaction between a DM and DESDEO. The `request` and `response` should be mainly used to store and communicate information to and from interactive methods available in DESDEO.

IV. USE CASES

In this section, we demonstrate how one can use the DESDEO framework to define different types of problems and solve them by applying interactive multiobjective optimization methods. For simplicity, we use a river pollution problem with five objective functions and two decision variables presented in Section IV-A. In Section IV-B, we describe how to define a problem with an analytical formulation and solve it using the synchronous NIMBUS method [34]. This method is in the *desdeo-mcdm* package and incorporates classification types of preferences. Section IV-C is devoted to defining and solving a data-driven problem. The interactive RVEA [43] method in the *desdeo-emo* package is applied, where preference information is given as a reference point. In Section IV-D, we consider some challenges of computationally expensive problems and follow the three-stage approach [50], where first, NSGA-III is utilized in a pre-decision-making stage to generate nondominated solutions. Then, a computationally inexpensive surrogate problem is formed. In the decision-making stage, the DM applies the interactive E-NAUTILUS [36] method to solve the surrogate problem. There can also be a post-decision-making stage to assure the Pareto optimality of the final solution. Here we consider the first two stages as a hybrid way of using methods within the DESDEO framework. Lastly, in Section IV-E we demonstrate how the DM can switch interactive methods in DESDEO to express his/her preferences in different ways. This example also illustrates some of the advantages the DESDEO environment provides. Finally, we discuss some graphical user interfaces in Section IV-F.

As mentioned, the main documentation of DESDEO can be found online (https://desdeo.readthedocs.io/en/latest/). The documentation of each core package discussed in Section III, can be readily accessed through the main documentation. We advice the reader to check the documentation for any additional details related to the use cases considered in Sections IV-B, IV-C, IV-D, and IV-E. The examples shown in these sections can also be found online in a Jupyter Notebook.[2]

A. THE RIVER POLLUTION PROBLEM

The river pollution problem [51] considers a river close to a city. There are two sources of pollution: industrial pollution from a fishery and municipal waste from the city and two treatment plants (in the fishery and the city). The pollution is reported in pounds of biochemical oxygen demanding material (BOD), and water quality is measured in dissolved oxygen concentration (DO).

Cleaning water in the city increases tax rate, and cleaning in the fishery reduces the return on investment. The problem is to improve the DO level in the city and at the municipality border (f_1 and f_2, respectively) while, at the same time, maximizing the percent return on investment at the fishery (f_3) and minimizing addition to the city tax (f_4). We consider a variant of the problem [52] with one more objective to ensure the treatment plants' efficiency by keeping the proportional amount of BOD removed from the water close to the ideal value of 0.65 (f_5). The corresponding multiobjective optimization problem where all objectives have been converted to be minimized is as follows:

$$\begin{aligned}
\min \quad & f_1(\mathbf{x}) = -4.07 - 2.27x_1 \\
\min \quad & f_2(\mathbf{x}) = -2.60 - 0.03x_1 - 0.02x_2 \\
& \qquad - \frac{0.01}{1.39 - x_1^2} - \frac{0.30}{1.39 - x_2^2} \\
\min \quad & f_3(\mathbf{x}) = -8.21 + \frac{0.71}{1.09 - x_1^2} \\
\min \quad & f_4(\mathbf{x}) = -0.96 + \frac{0.96}{1.09 - x_2^2} \\
\min \quad & f_5(\mathbf{x}) = \max\{|x_1 - 0.65|, |x_2 - 0.65|\} \\
\text{s.t.} \quad & 0.3 \leq x_1, x_2 \leq 1.0,
\end{aligned} \tag{2}$$

where the proportional amounts of BOD removed from water in the two treatment plants are, respectively, the decision variables x_1 and x_2.

B. USE CASE 1: PROBLEM WITH AN ANALYTICAL FORMULATION

DESDEO has good support for defining and optimizing problems with analytical formulations. DESDEO provides individual classes to define components of problem (1), i.e., objective functions, variables, and constraint functions separately. Box-constraints for variables are also supported.

Here we analytically define (2) and use modules of the *desdeo-problem* package and NumPy. The imports needed are shown in Source code 1. Notice that this problem has only box-constraints.

```
1 import numpy as np
2
3 from desdeo_problem import MOProblem, Variable,
  ↪  ScalarObjective
```

SOURCE CODE 1. Needed imports for a problem defined analytically. The class `MOProblem` is used to define a problem, the class `Variable` its decision variables, and the class `ScalarObjective` the objective functions.

We define the five objective functions as shown in Source code 2 as individual functions. These functions are expected to return a 1-dimensional NumPy array with each element representing the respective objective value when evaluated with one or more decision variable vectors. These decision

[2]https://desdeo.readthedocs.io/en/latest/notebooks/four_simple_use_cases.html

variable vectors are stored in 2-dimensional NumPy arrays, with each row representing a single vector. The defined functions are then used in the `ScalarObjective` class to instantiate new objects. Finally, each of the objects is stored in a list.

```
def f_1(x: np.ndarray) -> np.ndarray:
    x = np.atleast_2d(x)  # This step is to guarantee that the function works when called with a single decision variable vector as well.
    return -4.07 - 2.27*x[:, 0]

def f_2(x: np.ndarray) -> np.ndarray:
    x = np.atleast_2d(x)
    return -2.60 - 0.03*x[:, 0] - 0.02*x[:, 1] - 0.01 / (1.39 - x[:, 0]**2) - 0.30 / (1.39 + x[:, 1]**2)

def f_3(x: np.ndarray) -> np.ndarray:
    x = np.atleast_2d(x)
    return -8.21 + 0.71 / (1.09 - x[:, 0]**2)

def f_4(x: np.ndarray) -> np.ndarray:
    x = np.atleast_2d(x)
    return -0.96 - 0.96 / (1.09 - x[:, 1]**2)

def f_5(x: np.ndarray) -> np.ndarray:
    return np.max([np.abs(x[:, 0] - 0.65), np.abs(x[:, 1] - 0.65)], axis=0)

objective_1 = ScalarObjective(name="f_1", evaluator=f_1)
objective_2 = ScalarObjective(name="f_2", evaluator=f_2)
objective_3 = ScalarObjective(name="f_3", evaluator=f_3)
objective_4 = ScalarObjective(name="f_4", evaluator=f_4)
objective_5 = ScalarObjective(name="f_5", evaluator=f_5)

objectives = [objective_1, objective_2, objective_3, objective_4, objective_5]
```

SOURCE CODE 2. Defining the objective functions of problem (2).

The variables of the problem are defined similarly in Source code 3. Each `Variable` object is instantiated by providing the variable's name, initial value, and lower and upper bound (i.e., box-constraints). The objects are then stored in a list.

```
x_1 = Variable("x_1", 0.5, 0.3, 1.0)
x_2 = Variable("x_2", 0.5, 0.3, 1.0)

variables = [x_1, x_2]
```

SOURCE CODE 3. Defining the variables and their bounds for problem (2).

Finally, we define the multiobjective optimization problem by instantiating an `MOProblem` object in Source code 4 using the lists of `ScalarObjectives` and `Variables` defined earlier. If the problem had additional constraints, they would be defined in a similar way to objective functions and provided as a third argument (`constraints`) to the initialization method of the `MOProblem` class. However, here we only have box-constraints, which were accounted for when defining the variables in Source code 3.

As mentioned, we solve problem (2) in this case with the synchronous NIMBUS method [34]. Since it needs the ideal and nadir points, we approximate them with the payoff table method found in the *desdeo-mcdm* package's *utilities* module in Source code 5. We store them inside the object defining our problem to have easy access to them later.

```
mo_problem = MOProblem(variables=variables, objectives=objectives)
```

SOURCE CODE 4. Defining the multiobjective optimization problem object of problem (2).

```
from desdeo_mcdm.utilities import payoff_table_method

ideal, nadir = payoff_table_method(mo_problem)

mo_problem.ideal = ideal
mo_problem.nadir = nadir
```

SOURCE CODE 5. Applying the payoff table method to get approximations of the ideal and nadir points.

```
from desdeo_mcdm.interactive.NIMBUS import NIMBUS

nimbus = NIMBUS(mo_problem)

classification_request, _ = nimbus.start()
```

SOURCE CODE 6. Instantiating a NIMBUS object and invoking its `start` method. The `start` method returns two requests of which the second one is irrelevant to this example and is therefore matched to an underscore on line 5.

We can now start solving problem (2) using NIMBUS as shown in Source code 6. After importing the `NIMBUS` class, we instantiate an object of it by providing it `mo_problem`, which was defined earlier. The `start` method returns a `classification_request`, which is used to interact with the method as described in Section III-B. The `message`-entry found in the `content` attribute of `classification_request` is printed in Console 1. We remind the reader that in practice, a UI should handle `requests`. An example of such can be found in Section IV-F.

```
>>>print(classification_request.content["message"])
Please classify each of the objective values in one of the following categories:
        1. values should improve '<'
        2. values should improve until some desired aspiration level is reached '<='
        3. values with an acceptable level '='
        4. values which may be impaired until some upper bound is reached '>='
        5. values which are free to change '0'
Provide the aspiration levels and upper bounds as a vector. For categories 1, 3,
and 5, the value in the vector at the objective's position is ignored. Supply also
the number of maximum solutions to be generated.
```

CONSOLE 1. The message printed in the `request` returned by NIMBUS.

As seen in Console 1, we have been provided with instructions on how to proceed. The content of the `response`,

```
>>>print(
... classification_request.content["objective_values"])
objective values [ -5.746, -2.779, -6.906, -11.626,
↪  0.349 ]
>>>print("ideal", ideal)
ideal [ -6.339, -2.864, -7.499, -11.626, 0 ]
>>>print("nadir", nadir)
nadir [ -4.751, -2.767, -0.321, -1.920, 0.349 ]
```

CONSOLE 2. The objective values of the initial solution computed by NIMBUS, and the ideal and nadir points of the problem.

in the case of NIMBUS, also contains objective vectors, which we can inspect by printing them as done in Console 2.

```
response = {
    "classifications": ["<=", "0", "=", ">=", "<"],
    "levels": [-6.2, 0, 0, -3.0, 0],
    "number_of_solutions": 2,
}

classification_request.response = response

save_request, _ =
↪  nimbus.iterate(classification_request)
```

SOURCECODE 7. Defining a response with preference information required by NIMBUS, and then iterating. Newly computed objective vectors are then printed.

We then define a `response` in Source code 7 containing preference information, in this case, classifications, and continue iterating by invoking the `iterate` method of NIMBUS. The new objective vectors computed in the first iteration of NIMBUS are shown in Console 3.

```
>>>print("objective vectors",
↪  save_request.content["objectives"])
objective vectors [
    array([ -5.746, -2.799, -6.906, -3.000, 0.137 ]),
    array([ -5.545, -2.808, -7.146, -2.398, 5.508 ])
    ]
```

CONSOLE 3. The objective vectors computed by NIMBUS based on the classification given by the DM.

Iterations of the NIMBUS method may continue by defining new `responses` to the `requests` returned by subsequent invocations of the `iterate` method. According to the definition of NIMBUS [34] the subsequent `requests` can also prompt the DM to choose previously computed solutions between which to compute additional solutions, or to select previously computed solutions to be saved into an archive for later viewing, for example. How each of the `requests` is handled in practice and how information is displayed to the DM depends on the choice of a UI, as stated before. Here, we have only shown the information in textual format to showcase the concepts of `requests` and `responses`, which can be found in other interactive methods defined in DESDEO as well.

C. USE CASE 2: DATA-DRIVEN PROBLEM

As mentioned, the DESDEO framework can be used to solve data-driven problems by fitting surrogate models to the data. This means that the data is assumed to contain samples of decision variable values, and corresponding objective vectors and surrogate models are fitted to represent each objective function individually. To demonstrate this, in this use case, we assume that we only have access to a small number of data points generated before the initiation of the solution process. We have generated data points for problem (2) by sampling the feasible region in the decision space using Latin hypercube sampling [53], and evaluated them using the analytical functions to obtain the corresponding objective vectors. A total of 100 points were sampled and the resulting data set saved on disk. The structure of the dataset is shown in Table 2, where the first row contains the names of the columns (decision variables or objectives).

TABLE 2. Format of the raw data used for surrogate-assisted optimization.

x_1	x_2	f_1	f_2	f_3	f_4	f_5
0.8211	0.7949	-5.9339	-3.0502	-6.5023	-3.0557	0.1711
0.7328	0.4363	-5.7336	-2.8925	-6.9258	-2.0271	0.2136
...	...	...	...	...	...	...

```
import pandas as pd
from desdeo_problem.problem import DataProblem


training_data =
↪  pd.read_csv("./data/River_pollution.csv",
↪  comment="#")

problem = DataProblem(
    data=training_data,
    variable_names=["x_1", "x_2"],
    objective_names=["f_1", "f_2", "f_3", "f_4",
    ↪  "f_5"],
    bounds=pd.DataFrame(
        [[0.3, 0.3], [1.0, 1.0]],
        columns=["x_1", "x_2"],
        index=["lower_bound", "upper_bound"]),
)
```

SOURCECODE 8. Formulating the problem using data.

We formulate the problem as shown in Source code 8. The Pandas package is used for importing and handling the data as shown in line 5 with the variable `training_data`. We can now define the problem by instantiating a `DataProblem` object. This is done by passing training data, names of the decision variables and the objective functions, and the lower and upper bounds of the decision variables. If these bounds are not provided, the infimum and supremum of the dataset are assumed to be the bounds.

In Source code 9, we show how the newly created `DataProblem` object can be used to train surrogate models for the objectives. We begin by importing the surrogate modeling technique of choice. Here, we use the `GaussianProcessRegressor` class from `desdeo_problem`, which is a wrapper around the scikit-learn class of the same name. Similar wrappers can be defined for other options of existing surrogate models. The modeling

```
from desdeo_problem.surrogatemodels.SurrogateModels
    import GaussianProcessRegressor

problem.train(
    models=GaussianProcessRegressor,
    model_parameters={"optimizer":"fmin_l_bfgs_b"}
)
```

SOURCECODE 9. Training Gaussian process regression surrogate models for the objectives.

algorithm and model parameters can be passed to the `train` method of the `DataProblem` object, which automatically trains the surrogate models for all objectives. Details about advanced use cases of the `train` method, such as training different kinds of surrogate models for different objectives, can be found in the documentation of *desdeo-emo*. More information about the model parameters is available in the documentation of the package of the model used, in this case, scikit-learn.

Once the surrogate models have been trained, we apply here the interactive RVEA method from the *desdeo-emo* package to solve the resulting problem using reference points as preference information. In Source code 10, we pass the `problem` variable as the first argument to the `RVEA` instance. This is followed by two Boolean arguments: `interact=True` and `use_surrogates=True`. The first argument enables the use of an interactive version of RVEA as presented in [43]. The second argument enables RVEA to use the surrogate models as objectives in place of analytical functions. Details about other arguments which control various aspects of the evolutionary method can be found in the documentation of *desdeo-emo*.

```
from desdeo_emo.EAs import RVEA

evolver = RVEA(
    problem,
    interact=True,
    use_surrogates=True
)

(_, _, refp_request, _), _ = evolver.requests()

# references given to refp_request

(_, _, refp_request, _), _ =
    evolver.iterate(refp_request)
```

SOURCECODE 10. Using interactive RVEA to solve the surrogate problem.

We begin the interactive solution process by providing the first reference point to `evolver`. This is done by first calling the `request` method of `evolver`, which returns `refp_request` and additional requests, irrelevant in this example, matched to underscores. This is similar to the requests returned by the `start` method of `nimbus` in the previous subsection. The `refp_request` variable accepts preferences as a reference point. Similar to `classification_request` in the previous subsection, this object also has a `content` method and a `response` attribute. The comment on line 11 in Source code 10 signifies the DM providing preferences to `refp_request`. As mentioned in the previous subsection, this can be achieved by a command-line interface, a graphical UI, or by using a console environment, like IPython or Jupyter Notebook.

In Console 4, we show how preferences can be defined. The `content` attribute of `refp_request` can be shown to the DM to describe the acceptable ranges of the preferences (here, ranges for aspiration levels as components of the reference point) and how the preference information will be utilized in the method used. The DM then provides preferences to the `response` attribute of `refp_request` using a Pandas data frame. The names of the columns of this data frame have to be the same as the objective function names, and the values contained in the data frame reflect the preferences of the DM in the form of a reference point. The preference information can then be submitted to the `iterate` method of the evolver object to run one iteration of interactive RVEA. This involves running the evolutionary method for a number of generations. This number can be changed by the user using arguments of the `RVEA` class, and the details can be found in the documentation.

```
>>>print(refp_request.content["message"])
Provide a reference point worse than or equal to the
↪  ideal point:
f_1        -6.34
f_2      -3.4391
f_3     -7.69844
f_4     -11.1186
f_5    0.0495768
Name: ideal, dtype: object
The reference point will be used to focus the reference
↪  vectors towards the preferred region.
If a reference point is not provided, the previous state
↪  of the reference vectors is used.
If the reference point is the same as the ideal point,
↪  the reference vectors are spread uniformly in the
↪  objective space.

>>>refp_request.response = pd.DataFrame(
...     [[-5.7, -2.8, -6.9, -3.0, 0.1]],
...     columns=["f_1", "f_2", "f_3", "f_4", "f_5"])

>>>requests, _ = evolver.iterate(refp_request)
```

CONSOLE 4. Checking the contents of the preference request object and saving the DM's preferences using a console environment.

After each iteration, the solutions generated can be accessed through the `individuals` and `objectives` attributes of `evolver.population`. The former contains the decision variable vectors of the set of solutions, whereas the latter contains the corresponding set of objective vectors. The solutions received after one iteration of RVEA are shown in Figure 3 in the parallel coordinates plot. As can be seen, many solutions were found that follow the reference point of the DM (denoted in green color) closely. If, however, the DM is not satisfied with the results or wants to see solutions in a different region of the objective space, the steps shown in Console 4 can be repeated as many times as desired with different preference information.

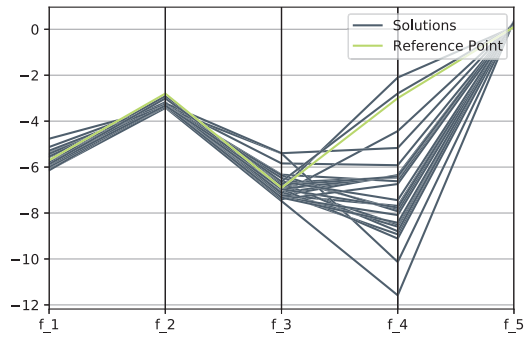


FIGURE 3. Solutions obtained for the data-driven river pollution problem after using RVEA for one iteration.

D. USE CASE 3: COMPUTATIONALLY EXPENSIVE PROBLEM

Multiobjective optimization problems can involve expensive function evaluations. In such cases, computing new solutions in each iteration of an interactive method is not feasible because of the long periods of time a DM would have to wait to see new solutions. Instead, we can use an interactive multiobjective optimization method that works on a computationally less expensive surrogate problem based on a pre-computed representation of Pareto optimal solutions. To get a representation, we can use, e.g., some a posteriori methods like EMO methods. Here we use this simple, yet quite effective, way as an example of combining methods from the *desdeo-mcdm* and *desdeo-emo* packages.

```
from desdeo_emo.EAs import NSGAIII

evolver = NSGAIII(mo_problem, interact=False)

while evolver.continue_evolution():
    evolver.iterate()

individuals, pareto_set = evolver.end()
```

SOURCECODE 11. Generating a representation of Pareto optimal solutions for problem (2) using the *desdeo-emo* package.

We first generate a representative set approximating Pareto optimal solutions for the river pollution problem (2) as shown in Source code 11. For this, we apply NSGA-III (activated by using the `interact=False` argument) as an a posteriori method to get solutions for the problem as implemented in Source code 4. The method is run until a pre-determined termination criterion is met (the default being 1000 generations). After this, we can use the `end` method of the evolver object to extract a representation of the Pareto front (i.e., non-dominated solutions) from the population as `individuals` (decision vectors) and `pareto_set` (objective vectors).

We then apply the E-NAUTILUS method [36] with the generated set of nondominated solutions as its input. It also needs estimates of the ideal and nadir points, typically estimated from the available solutions. However, because we have previously computed (in Section IV-B) the ideal and (estimated) nadir points, we apply them. As mentioned in Section II-D, the solution process starts with an inferior solution and gradually approaches the Pareto front.

In Source code 12, we set up the E-NAUTILUS method using the `ENautilus` class from the *desdeo-mcdm* package. We invoke the `start` method as we did in the case of NIMBUS in Section IV-B to start the solution process. We can get a hint on how to progress by printing the message stored in the request as done in Console 5.

```
from desdeo_mcdm.interactive import ENautilus

# 'pareto_set' stores the set of solutions computed using NSGA-III
method = ENautilus(pareto_set, ideal, nadir)

enautilus_request = method.start()
```

SOURCECODE 12. Initializing the E-NAUTILUS method using the set of solutions computed using NSGA-III and the previously computed ideal and nadir points of problem (2).

```
>>>print(enautilus_request.content["message"])
Please specify the number of iterations as
↪  'n_iterations'
to be carried out, and how many intermediate points to
↪  show as 'n_points'.
```

CONSOLE 5. The help message returned by starting the E-NAUTILUS method.

```
response = {"n_iterations": 5, "n_points": 3}

enautilus_request.response = response

enautilus_request = method.iterate(request)
```

SOURCECODE 13. Specifying the number of iterations to be carried out and the number of points to be shown in each iteration of the E-NAUTILUS method.

A response to the request returned by the `start` method is then defined in Source code 13. We choose five iterations and want to see three intermediate points after each iteration. We then continue iterating and get a new request from invoking the `iterate` method, which contains the message displayed in Console 6.

```
>>>print(enautilus_request.content["message"])
Please select the most preferred point by index as
↪  'preferred_point_index'
```

CONSOLE 6. The help message in a request returned from iterating the E-NAUTILUS method after it has been started.

To address the message shown in Console 6, we first inspect the intermediate points and bounds of the reachable solutions computed in the first iteration of E-NAUTILUS in Console 7.

In Source code 14, we define a response to the current request and continue iterating by invoking the `iterate` method. Subsequent iterations are carried out as shown in

```
>>>print("Points:", enautilus_request.content["points"])
Points:
 [[ -4.985, -2.911, -1.562, -3.470, 0.347 ]
 [  -5.012, -2.853, -1.460, -2.229, 0.325 ]
 [  -5.067, -2.915, -0.395, -3.489, 0.349  ]]
>>>print("Upper bounds:",
↪  enautilus_request.content["upper_bounds"])
Upper bounds:
 [[ -4.756, -3.100, -0.691, -2.094,  0.35 ]
 [  -4.808, -2.927, -3.204, -1.967,  0.35 ]
 [  -4.752, -3.100, -0.406, -2.094,  0.35 ]]
>>>print("Lower bounds:",
↪  enautilus_request.content["lower_bounds"])
Lower bounds:
 [[ -6.315, -3.430, -7.443, -11.158, 0.196 ]
 [  -6.280, -3.347, -7.425, -7.785,  0.009 ]
 [  -6.338, -3.440, -7.401, -11.511, 0.196 ]]
```

CONSOLE 7. Printing the intermediate points, upper bounds, and lower bounds computed in the first iterations of E-NAUTILUS.

```
response = {"preferred_point_index": 2}

enautilus_request.response = response

enautilus_request = method.iterate(enautilus_request)
```

SOURCECODE 14. Expressing preference to be the second point shown in Console 7 and iterating.

Source code 14. We show an example of this using a graphical UI in the next subsection.

This way of exploring an existing representation of a Pareto front is well suited for computationally expensive problems since no expensive function evaluations are needed when the DM is involved. Even though the problem in this example is not really computationally expensive, the process of first using an EMO method to compute a representation of the Pareto optimal front, and then exploring it using the E-NAUTILUS method is identical in a computationally expensive case.

E. USE CASE 4: SWITCHING METHODS

As interactive multiobjective optimization methods vary in the type of preference information they require from a DM and the type of information they provide to the DM, it is sometimes desirable to switch between iterations to a method that is better suited to the changing needs of the DM. The DESDEO environment enables this kind of a switch even between different types of methods. To illustrate this, we consider an example, where we have finished iterating with the E-NAUTILUS method as described in Subsection IV-D and arrived at the solution `[-6.27116931, -2.80042652,-3.46795271, -6.57327201, 0.31967811]`. Since this method uses a set of solutions approximating the Pareto front, we can improve the solution by utilizing the synchronous NIMBUS method and considering the original problem (2) in its analytical form. We can also think that we have applied E-NAUTILUS as a trade-off-free method to find a good starting point for NIMBUS and avoided anchoring at a randomly selected starting point. From NIMBUS, we then switch to applying the reference point method [33], that is, change the preference information type from classifying the objectives to providing a reference point.

To begin, we instantiate a `NIMBUS` object with the solution we arrived at with E-NAUTILUS. We use this solution to derive a Pareto optimal solution that NIMBUS starts with in Source code 15. This solution is shown in Console 8. Small improvements were made in the values of the third and fourth objectives while the other objective values remained unchanged. This is also an example of a possible realization of the post-decision-making stage (mentioned at the beginning of Section IV) to assure the Pareto optimality of the solution found using E-NAUTILUS since EMO methods (used here to generate the input set for E-NAUTILUS) cannot guarantee Pareto optimality.

```
nimbus_method = NIMBUS(mo_problem,
↪  starting_point=np.array([-6.27116931,
↪  -2.80042652, -3.46795271, -6.57327201,
↪  0.31967811]))
classification_request, _ = nimbus_method.start()
solution =
↪  classification_request.content["objective_values"]
```

SOURCECODE 15. Instantiating a NIMBUS object with a specified starting point and starting the method.

```
>>>print("Improved starting point:", solution")
Improved starting point: [-6.27116931 -2.80042652
↪  -3.46795286 -6.57327759  0.31967811]
```

CONSOLE 8. Printing the solution to be classified in the first iteration of synchronous NIMBUS in Source code 15.

```
response = {
    "classifications": [">=", ">=", "=", "<=", "<="],
    "levels": [-6.0, -2.78, 0, -5.5, 0.28],
    "number_of_solutions": 4,
}

classification_request.response = response

save_request, _ =
↪  nimbus_method.iterate(classification_request)

alternatives = np.array(save_request["objectives"])
```

SOURCECODE 16. Providing classification to synchronous NIMBUS and iterating the method further.

Next, we take an iteration with the synchronous NIMBUS using the classification shown in Source code 16. Based on this preference information, the method provides four new Pareto optimal solutions shown in Console 9. While inspecting the solutions, we find the first to our liking, but we would next like to provide a reference point instead of a classification. Thus, we switch to using the reference point method in the *desdeo-mcdm* module. We initialize the reference point method by instantiating a `ReferencePointMethod` object as done in Source code 17.

We provide the best (i.e., the solution we like the most) NIMBUS solution (`[-6.06739, -2.79173,`

```
>>>print("New solutions based on the classification:",
↪  alternatives)
New solutions based on the classification:
[[-6.06739 -2.79173 -5.96145 -6.57333 0.30863]
 [-6.17792 -2.79803 -5.09184 -5.39729 0.28469]
 [-6.17191 -2.79781 -5.15768 -5.38138 0.28427]
 [-6.15075 -2.79704 -5.36753 -5.33890 0.28314]]
```

CONSOLE 9. **Printing the new solutions computed in Sourcecode 16.**

```
1 from desdeo_mcdm.interactive import
  ↪  ReferencePointMethod
2
3 rp_method = ReferencePointMethod(mo_problem,
  ↪  mo_problem.ideal, mo_problem.nadir)
4 initial_request = rp_method.start()
```

SOURCECODE 17. **Instantiating a reference point method object and starting it.**

`-5.96145, -6.57333, 0.30863]`) as the reference point in Source code 18 by defining the reference point as a part of the `response`. We get the solutions shown in Console 10 and since the first one is so similar to the reference point, we stop the solution process and select the first solution as the final one.

```
1 response = {
2     "reference_point": np.array([-6.06739, -2.79173,
      ↪  -5.96145, -6.57333, 0.30863]),
3 }
4
5 initial_request.response = response
6
7 rp_request = rp_method.iterate(initial_request)
8
9 alternatives =
  ↪  np.array(rp_request.content["additional_solutions"])
```

SOURCECODE 18. **Iterating with the reference point method by providing a reference point.**

```
>>>print("Alternative solutions:", alternatives)
Alternative solutions:
[[ -6.06738  -2.79173  -5.96151  -6.57720   0.30869]
 [ -6.06739  -2.78816  -5.96146 -11.62453   0.34999]
 [ -6.06827  -2.79173  -5.95658  -6.59358   0.30895]
 [ -6.06739  -2.79173  -5.96146  -6.57853   0.30871]
 [ -6.06810  -2.79162  -5.95754  -6.67149   0.31016]]
```

CONSOLE 10. **Printing the alternative solutions computed by the reference point method after providing the reference point as done in Source code 18.**

If we were not satisfied yet, we could continue iterating by providing a new reference point in a similar way to what was done in Source code 18. We could also switch back to the synchronous NIMBUS method by initializing a `NIMBUS` object once more, as was done in Source Code 15. In principle, we could also switch to an EMO method, for example, by providing one of the solutions as a reference point to RVEA similar to how it was done in Source code 10, while also switching back to the surrogate version of problem (2). But in this case it makes no sense to switch from Pareto optimal to approximated solutions. Naturally, we are not limited to the interactive methods considered in the use cases, but any method in DESDEO is applicable. Without DESDEO, we would be forced to resort to switching our whole working environment, which may require needless repetition, for example, redefining the same problem multiple times (and possibly in a different syntax). Thanks to DESDEO, we have all the methods, problems, and other relevant information (e.g., solutions computed with different methods) in the same environment, which allows to readily switch methods and re-use already created information.

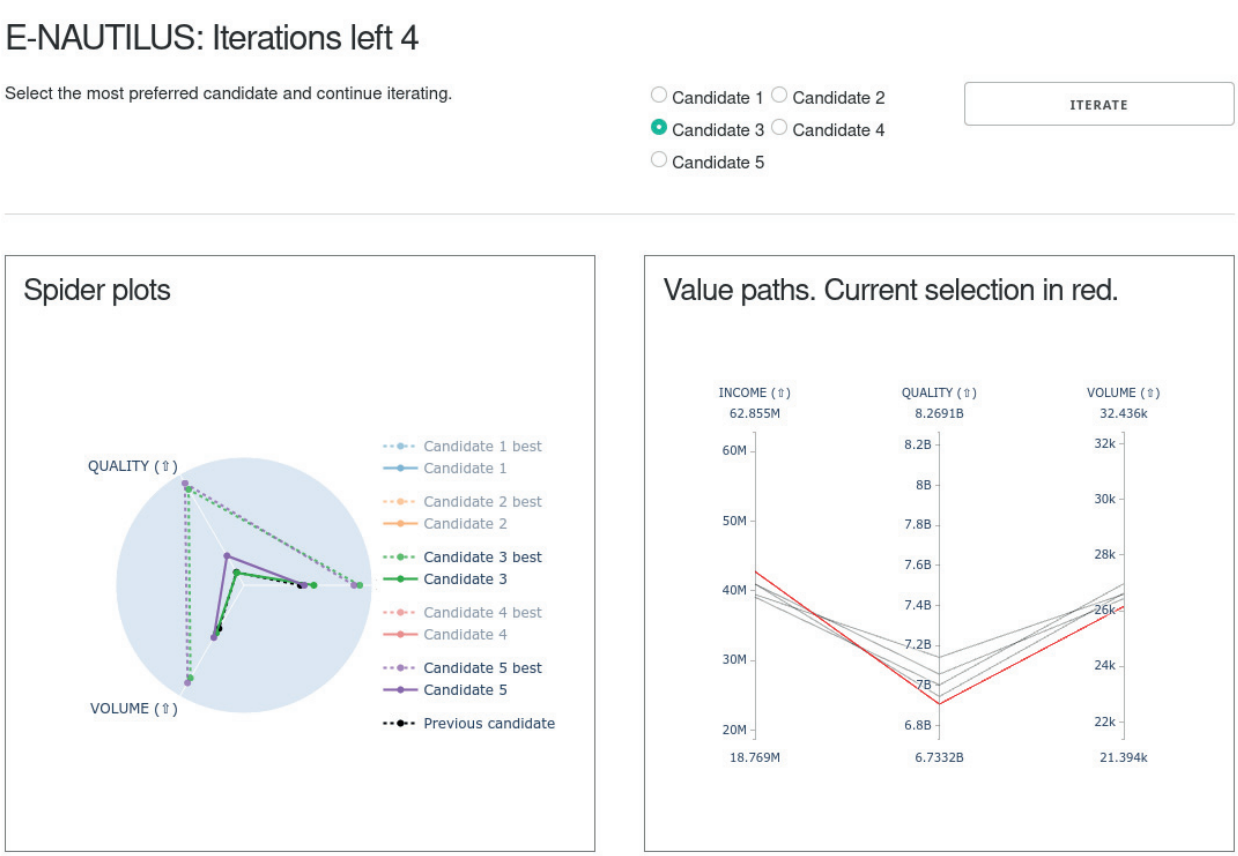
F. SOFTWARE APPLICATIONS BUILT UTILIZING DESDEO

So far, we have not really discussed UIs in the DESDEO framework apart from using a console environment. However, DESDEO is easy to extend to build more advanced software applications, such as graphical user interfaces (GUIs), which facilitate interaction between DMs and interactive methods. In this section, we explore an example of such a GUI implemented for a method in the *desdeo-mcdm* package. It is naturally possible to implement similar GUIs for methods in the *desdeo-emo* package as well.

We consider an interface implemented for E-NAUTILUS. We have furthermore chosen a web interface because they are accessible to anyone through any modern web browser. The interface has been developed using the Python libraries plotly and plotly-dash (https://plotly.com/) due to their ease of use and versatility for developing web interfaces. However, there is a significant lack in support for interactive visualizations in these libraries, which we had to circumvent, leading to a lack in general usability.

The web interface for E-NAUTILUS can be seen in Figure 4. At the top of the interface, we have controls for the DM to engage with E-NAUTILUS: the DM can choose the most preferred intermediate point (labeled as *'candidate'* in the figure) by using the radio buttons and click on the 'ITERATE'-button to continue iterating. Below the controls, there are three different ways to visualize information about the intermediate points calculated: at the top left, a spider plot showing the intermediate points (solid lines) and the best reachable objective function values from each point (dashed lines); at the top right, a parallel coordinate plot showing only the intermediate points with the currently selected point (using radio buttons) being highlighted in red; and at the very bottom, the values of each intermediate point and their best reachable values in a table with the currently selected intermediate point highlighted in blue.

The spider plot in Figure 4 is worth a closer look. First, each intermediate point can be explored by clicking the respective point on the legend to the right of the plot. Second, the plot also shows in black the intermediate point chosen by the DM *in the previous iteration*. Showing the previously chosen point was desired by a real DM in a practical application where this interface was used. This is an example of a subjective need that may arise when interacting with real DMs. Lastly, it is worth comparing the information in Figure 4 to the information outputted in Console 7 to see that the information



Tabled candidate objective values

Candidate	INCOME (⇧)	QUALITY (⇧)	VOLUME (⇧)
1	41,011,177.17	6,944,671,947.74	26,627.11
2	40,988,980.81	7,056,474,462.39	26,431.18
3	42,788,884.27	6,906,564,678.96	26,151.17
4	39,107,181.60	7,002,406,061.35	26,967.98
5	39,494,600.68	7,140,768,731.69	26,594.86

Tabled candidate best reachable values

Best reachable	INCOME (⇧)	QUALITY (⇧)	VOLUME (⇧)
1	57,829,414.62	8,145,971,745.04	30,867.17
2	57,486,543.70	8,145,971,745.04	30,867.17
3	58,521,063.17	8,059,687,621.67	30,623.93
4	57,049,818.26	8,145,971,745.04	31,147.81
5	56,558,622.26	8,145,971,745.04	31,105.08

Back to method index

FIGURE 4. The GUI of the E-NAUTILUS method implemented in plotly-dash. A toy multiobjective optimization problem with three objectives (INCOME, QUALITY, VOLUME) to be maximized is shown.

shown for a single iteration in the web interface and the console are virtually the same. In practice, the presented interface simply handles the `requests` and `responses` (discussed in Section III-B) as was done in Section IV-D. In the E-NAUTILUS GUI, we have a different multiobjective optimization problem with three objectives to be maximized

instead of problem (2). We have chosen a problem with fewer objectives for simplicity. Note that the arrows after the function names remind of the maximization.

The interface described for E-NAUTILUS is available online (https://desdeo.it.jyu.fi/dash) alongside an interface implementation for NAUTILUS Navigator as well. The source code for the interface shown is available on GitHub online.[3] To test the interfaces, we have provided the interested reader with toy data online.[4]

V. POTENTIAL OF THE DESDEO FRAMEWORK

Because the DESDEO framework contains various interactive methods, it enables versatile ways of applying them. As said, the DM can conveniently switch the method during the solution process. This can be desirable if (s)he wants to change the type of preference information in the middle of the solution process or get different types of information about the problem. This opens up vast possibilities when the DM is not forced to stick with a single method to be applied but can select methods that best suit the different phases of the solution process (e.g., learning and decision phases [54]). This potential has been considered in [55], where a generic multiagent architecture for interactive methods was proposed to support DMs in selecting the most suited interactive method based on preferred preference type and their needs in different phases during the solution process. Without a framework like DESDEO, switching the method is inconvenient; the problem to be solved must be connected to individual multiobjective optimization methods separately, and the solution history with the previous method is not easily available.

DESDEO has clear potential in allowing researchers to hybridize EMO and MCDM methods in novel ways. This potential is not just limited to the example seen in Section IV, where an EMO method was used to compute a representation of a Pareto front, which was then explored using an MCDM method. More innovative and advanced ways of combining not just methods but also their individual components are possible. This is because of the modular fashion in which the various multiobjective optimization methods have been implemented in DESDEO. Combining individual components enables the development of new interactive methods, which can be also included in DESDEO extending the framework further. The IOPIS algorithm, described in [46], is an example of such a method.

Moreover, DESDEO offers a promising basis for implementing new interactive multiobjective optimization methods that are not based on combining existing components. Due to the modular structure, a developer can easily reuse already implemented components and only add those that are not yet available (if needed). For example, the *desdeo-tools* package has a wide variety of different tools ranging from achievement scalarizing functions to fast nondominated sorting, which can prove useful in implementing new methods. In addition, experimenting with new methods and ideas in multiobjective optimization is also made easy thanks to DESDEO and the reusability of its components. DESDEO can also encourage and lower the threshold for researchers to implement their methods as open source code, contributing to the openness of the research conducted in multiobjective optimization. This way, DESDEO has the potential and is on a good track to becoming a central hub for open implementations of interactive multiobjective optimization methods.

Apart from being interesting from an academic perspective, DESDEO can naturally be utilized for modeling and solving real-life problems from any field as long as the problem can be modeled as a multiobjective optimization problem. Depending on the type and requirements of the problem, DESDEO might still lack certain features necessary for modeling and solving the problem, which is also one of the current limitations of DESDEO. However, due to DESDEO's modular structure and open source nature, implementing these missing features is possible by anyone. For example, the underlying optimization methods for single-objective optimization problems arising in various interactive methods in DESDEO can be changed to better account for the type of problem being solved. Similarly, the crossover and mutation operations in EMO methods can also be customized if need be. Lastly, in modeling a data-driven multiobjective optimization problem, almost any surrogate model can be implemented and used. Obviously, existing features in DESDEO can be combined with new features as well allowing practitioners to save time and help them focus on solving the problem at hand. In this way, DESDEO can be extended to account for any kind of multiobjective optimization problem from any field while decreasing the potential workload for practitioners.

Being a software framework, DESDEO has a learning curve to it, which means that a certain level of proficiency in Python and multiobjective optimization is to be expected from the user. This clearly limits the size of the potential user base of DESDEO and is, therefore, one of the framework's major limitations at the present time. We already offer a written documentation of DESDEO's features, but to make DESDEO even more accessible, we plan on including more topical guides in the documentation on how to use DESDEO (such as the ones presented in Section IV) and consider producing tutorial videos on how to use DESDEO in the future. This should help broaden DESDEO's user base and allow users to extend DESDEO to meet their individual needs. All of this will help DESDEO grow further as a software framework.

Comparison and identifying the best suited method for various needs are important. DESDEO does offer very promising opportunities for comparing and validating different interactive methods. This is vital and demanding because the DM plays an important role in the solution process and conducting experiments with human participants is challenging. To be able to compare interactive methods, their performance needs

[3] https://github.com/industrial-optimization-group/desdeo-dash
[4] https://github.com/industrial-optimization-group/DESDEO/blob/master/docs/notebooks/data/toy_data.csv

to be evaluated and validated using appropriate quality indicators. To the best of our knowledge, no quality indicators for interactive methods have been proposed. For such quality indicators, the desirable properties that qualify interactive solution processes should be defined. In [56], a systematic literature review of the assessments of interactive methods is provided along with desirable properties for interactive methods. This can be considered as the initial step towards developing quality indicators for interactive methods. Moreover, there has been some interest in comparing interactive methods with so-called artificial DMs in the literature (e.g., [57]–[59]). Within the DESDEO framework, an artificial DM has recently been proposed to compare reference point-based interactive EMO methods [60]. DESDEO provides an excellent platform for comparisons because it involves various interactive methods within the same framework. To utilize the opportunities available, we need artificial DMs capable of handling different types of preferences and methods.

VI. CONCLUSION

In this paper, we fill a gap in the optimization software available. We introduced DESDEO: an open source multiobjective optimization framework implemented in Python. DESDEO makes interactive multiobjective optimization methods openly available for both users and developers. We introduced the modular structure of DESDEO and its different packages and their modules. We also described the purpose of each package and its dependencies and the framework's external dependencies. Besides, with a five-objective optimization problem, we demonstrated how to use the DESDEO framework to define different types of problems (i.e., with analytical expressions, data-driven, and computationally expensive problems) and solve them by applying and hybridizing interactive multiobjective optimization methods of MCDM and EMO types.

The modularity of DESDEO eases developing new methods and offers a convenient possibility of comparing different interactive methods. Furthermore, implementing different types of methods in the same framework, as done in DESDEO, will start a new era in hybridization and allows the DM to switch between methods in various iterations of the solution process.

We also noted that for efficient interaction with the DM, there is a need for interactive visualization tools and suitable (graphical) UIs in multiobjective optimization, which is lacking in the literature. We are addressing this practical concern by actively developing a D3 (https://d3js.org/) based Typescript library of interactive visualization components, such as interactive parallel coordinate plots within DESDEO. Our primary goal with this library is to provide the multiobjective optimization community with new and needed tools to build their own interfaces for interactive multiobjective optimization; similar to the example seen in Section IV-F. To facilitate the use of the packages in DESDEO to be extended to other software, such as web based interfaces, we are also working on a web API (application programming interface) through which we can expose interactive methods in DESDEO to enable their use in a variety of applications. The interested reader can follow the latest developments of DESDEO via its homepage (desdeo.it.jyu.fi). The realization of this vision should make interactive multiobjective optimization methods much more accessible in the future, not just for researchers developing them, but also for the needs of applications in various fields.

ACKNOWLEDGMENT

The authors would like to thank all of those who have contributed to DESDEO in the past. Especially, they would like to thank Giomara Lárraga, Johanna Silvennoinen, Pouya Aghaei Pour, Juuso Pajasmaa, Stefan Otayagich, and Antti Luopajärvi. They would also like to thank Vesa Ojalehto for his pioneering work in developing the old version of the DESDEO framework. This work is a part of the thematic research area Decision Analytics Utilizing Causal Models and Multiobjective Optimization (DEMO, jyu.fi/demo) at the University of Jyväskylä.

REFERENCES

[1] C.-L. Hwang and A. Masud, *Multiple Objective Decision Making–Methods and Applications: A State-of-the-Art Survey*. Berlin, Germany: Springer, 1979.

[2] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making: Theory and Methodology*. New York, NY, USA: Elsevier, 1983.

[3] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston, MA, USA: Kluwer, 1999.

[4] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed. New York, NY, USA: Springer, 2007.

[5] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, U.K.: Wiley, 2001.

[6] K. Miettinen and M. M. Mäkelä, "On scalarizing functions in multiobjective optimization," *OR Spectr.*, vol. 24, no. 2, pp. 193–213, May 2002.

[7] F. Ruiz, M. Luque, and J. M. Cabello, "A classification of the weighting schemes in reference point procedures for multiobjective programming," *J. Oper. Res. Soc.*, vol. 60, no. 4, pp. 544–553, Apr. 2009.

[8] K. Miettinen, J. Hakanen, and D. Podkopaev, "Interactive nonlinear multiobjective optimization methods," in *Multiple Criteria Decision Analysis: State of the Art Surveys*, 2nd ed., S. Greco, M. Ehrgott, and J. Figueira, Eds. New York, NY, USA: Springer, 2016, pp. 931–980.

[9] M. Luque, F. Ruiz, and K. Miettinen, "Global formulation for interactive multiobjective optimization," *OR Spectr.*, vol. 33, no. 1, pp. 27–48, Jan. 2011.

[10] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds., *Multiobjective Optimization: Interative and Evolutionary Approaches*. Berlin, Germany: Springer, 2008.

[11] B. Xin, L. Chen, J. Chen, H. Ishibuchi, K. Hirota, and B. Liu, "Interactive multiobjective optimization: A review of the state-of-the-art," *IEEE Access*, vol. 6, pp. 41256–41279, 2018.

[12] V. Ojalehto and K. Miettinen, "DESDEO: An open framework for interactive multiobjective optimization," in *Multiple Criteria Decision Making and Aiding*, S. Huber, M. J. Geiger, and A. T. de Almeida, Eds. Cham, Switzerland: Springer, 2019, pp. 67–94.

[13] G. Rossum, "Python reference manual," NLD, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, Tech. Rep., 1995.

[14] R. Benayoun, J. de Montgolfier, J. Tergny, and O. Laritchev, "Linear programming with multiple objective functions: Step method (STEM)," *Math. Program.*, vol. 1, no. 1, pp. 366–375, Dec. 1971.

[15] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, "An adaptive Bayesian approach to surrogate-assisted evolutionary multi-objective optimization," *Inf. Sci.*, vol. 519, pp. 317–331, May 2020.

[16] S. N. Qasem, S. M. Shamsuddin, S. Z. M. Hashim, M. Darus, and E. Al-Shammari, "Memetic multiobjective particle swarm optimization-based radial basis function network for classification problems," *Inf. Sci.*, vol. 239, pp. 165–190, Aug. 2013.
[17] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, Feb. 2006.
[18] M. Li, G. Li, and S. Azarm, "A kriging metamodel assisted multi-objective genetic algorithm for design optimization," *J. Mech. Design*, vol. 130, no. 3, pp. 1–10, Mar. 2008.
[19] H. Aytuğ and S. Sayın, "Using support vector machines to learn the efficient set in multiple objective discrete optimization," *Eur. J. Oper. Res.*, vol. 193, no. 2, pp. 510–519, Mar. 2009.
[20] G. Kourakos and A. Mantoglou, "Development of a multi-objective optimization algorithm using surrogate models for coastal aquifer management," *J. Hydrol.*, vol. 479, pp. 13–23, Feb. 2013.
[21] K. Mitra and S. Majumder, "Successive approximate model based multi-objective optimization for an industrial straight grate iron ore induration process using evolutionary algorithm," *Chem. Eng. Sci.*, vol. 66, no. 15, pp. 3471–3481, Aug. 2011.
[22] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *J. Mach. Lang. Res.*, vol. 13, pp. 2171–2175, Jul. 2012.
[23] A. Garrett, *Inspyred: Bio-inspired algorithms in Python*. Accessed: Nov. 19, 2020. [Online]. Available: https://github.com/aarongarrett/inspyred
[24] X. Gandibleux, G. Soleilhac, A. Przybylski, and S. Ruzika, "vOptSolver: An open source software environment for multiobjective mathematical optimization," in *Proc. 21st Conf. Int. Fed. Oper. Res. Societies (IFORS)*, 2017, pp. 17–21.
[25] D. Hadka. *Platypus: Multiobjective Optimization in Python*. Accessed: Nov. 19, 2020. [Online]. Available: https://platypus.readthedocs.io
[26] D. Hadka. *MOEA Framework: A Free and Open Source Java Framework for Multiobjective Optimization*. Accessed: Dec. 1, 2020. [Online]. Available: http://moeaframework.org/
[27] D. Izzo and F. Biscani. *PyGMO: Python Parallel Global Multiobjective Optimizer*. Accessed: Nov. 19, 2020. [Online]. Available: https://esa.github.io/pygmo
[28] F. Biscani, D. Izzo, and C. H. Yam, "A global optimisation toolbox for massively parallel engineering optimisation," 2010, *arXiv:1004.3824*.
[29] A. Benítez-Hidalgo, A. J. Nebro, J. García-Nieto, I. Oregi, and J. Del Ser, "JMetalPy: A Python framework for multi-objective optimization with metaheuristics," *Swarm Evol. Comput.*, vol. 51, Dec. 2019, Art. no. 100598.
[30] J. J. Durillo and A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011.
[31] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in Python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020.
[32] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 73–87, Nov. 2017.
[33] A. P. Wierzbicki, "A mathematical basis for satisficing decision making," *Math. Model.*, vol. 3, no. 5, pp. 391–405, 1982.
[34] K. Miettinen and M. M. Mäkelä, "Synchronous approach in interactive multiobjective optimization," *Eur. J. Oper. Res.*, vol. 170, no. 3, pp. 909–922, May 2006.
[35] K. Miettinen and F. Ruiz, "NAUTILUS framework: Towards trade-off-free interaction in multiobjective optimization," *J. Bus. Econ.*, vol. 86, nos. 1–2, pp. 5–21, Jan. 2016.
[36] A. B. Ruiz, K. Sindhya, K. Miettinen, F. Ruiz, and M. Luque, "E-NAUTILUS: A decision support system for complex multiobjective optimization problems based on the NAUTILUS method," *Eur. J. Oper. Res.*, vol. 246, no. 1, pp. 218–231, Oct. 2015.
[37] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 773–791, Oct. 2016.
[38] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Apr. 2013.
[39] K. Miettinen, P. Eskelinen, F. Ruiz, and M. Luque, "NAUTILUS method: An interactive technique in multiobjective optimization based on the nadir point," *Eur. J. Oper. Res.*, vol. 206, no. 2, pp. 426–434, Oct. 2010.
[40] K. Miettinen, D. Podkopaev, F. Ruiz, and M. Luque, "A new preference handling technique for interactive multiobjective optimization without trading-off," *J. Global Optim.*, vol. 63, no. 4, pp. 633–652, Dec. 2015.
[41] A. B. Ruiz, F. Ruiz, K. Miettinen, L. Delgado-Antequera, and V. Ojalehto, "NAUTILUS Navigator: Free search interactive multiobjective optimization without trading-off," *J. Global Optim.*, vol. 74, no. 2, pp. 213–231, Jun. 2019.
[42] M. Hartikainen, K. Miettinen, and K. Klamroth, "Interactive nonconvex Pareto navigator for multiobjective optimization," *Eur. J. Oper. Res.*, vol. 275, no. 1, pp. 238–251, May 2019.
[43] J. Hakanen, T. Chugh, K. Sindhya, Y. Jin, and K. Miettinen, "Connections of reference vectors and different types of preference information in interactive multiobjective evolutionary algorithms," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.
[44] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. London, U.K.: Springer, 2005, pp. 105–145.
[45] A. P. Wierzbicki, "On the completeness and constructiveness of parametric characterizations to vector optimization problems," *Oper.-Res.-Spektrum*, vol. 8, no. 2, pp. 73–87, Jun. 1986.
[46] B. S. Saini, J. Hakanen, and K. Miettinen, "A new paradigm in interactive evolutionary multiobjective optimization," in *Parallel Problem Solving From Nature—PPSN XVI*, T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann, Eds. Cham, Switzerland: Springer, 2020, pp. 243–256.
[47] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011.
[48] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, and S. J. Van Der Walt, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, Feb. 2020.
[49] W. McKinney, "Data structures for statistical computing in Python," in *Proc. 9th Python Sci. Conf.*, S. van der Walt and J. Millman, Eds. SciPy, 2010, pp. 56–61.
[50] I. Steponavičė, S. Ruuska, and K. Miettinen, "A solution process for simulation-based multiobjective design optimization with an application in the paper industry," *Comput.-Aided Des.*, vol. 47, pp. 45–58, Feb. 2014.
[51] S. C. Narula and H. R. Weistroffer, "A flexible method for nonlinear multicriteria decision-making problems," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 4, pp. 883–887, Jul. 1989.
[52] K. Miettinen and M. M. Mäkelä, "Interactive method NIMBUS for nondifferentiable multiobjective optimization problems," in *Multicriteria Analysis*, J. Clímaco, Ed. Berlin, Germany: Springer, 1997, pp. 310–319.
[53] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.
[54] K. Miettinen, F. Ruiz, and A. P. Wierzbicki, "Introduction to multiobjective optimization: Interactive approaches," in *Multiobjective Optimization: Interative and Evolutionary Approaches*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds. Berlin, Germany: Springer, 2008, pp. 27–57.
[55] B. Afsar, D. Podkopaev, and K. Miettinen, "Data-driven interactive multiobjective optimization: Challenges and a generic multi-agent architecture," *Proc. Comput. Sci.*, vol. 176, pp. 281–290, Jan. 2020.
[56] B. Afsar, K. Miettinen, and F. Ruiz, "Assessing the performance of interactive multiobjective optimization methods: A survey," *ACM Comput. Surveys*, vol. 54, no. 4, p. 85, 2021.
[57] C. Barba-González, V. Ojalehto, J. M. García-Nieto, A. J. Nebro, K. Miettinen, and J. F. Aldana-Montes, "Artificial decision maker driven by PSO: An approach for testing reference point based interactive methods," in *Proc. 15th Int. Conf. Parallel Problem Solving Nature—PPSN XV*, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Eds. Cham, Switzerland: Springer, 2018, pp. 274–285.
[58] S. Huber, M. J. Geiger, and M. Sevaux, "Simulation of preference information in an interactive reference point-based method for the bi-objective inventory routing problem," *J. Multi-Criteria Decis. Anal.*, vol. 22, nos. 1–2, pp. 17–35, Jan. 2015.
[59] V. Ojalehto, D. Podkopaev, and K. Miettinen, "Towards automatic testing of reference point based interactive methods," in *Parallel Problem Solving From Nature—PPSN XIV*, J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, Eds. Cham, Switzerland: Springer, 2016, pp. 483–492.

[60] B. Afsar, K. Miettinen, and A. B. Ruiz, "An artificial decision maker for comparing reference point based interactive evolutionary multiobjective optimization methods," in *Evolutionary Multi-Criterion Optimization*, H. Ishibuchi, Q. Zhang, R. Cheng, K. Li, H. Li, H. Wang, and A. Zhou, Eds. Cham, Switzerland: Springer, 2021, pp. 619–631.

G. MISITANO received the M.Sc. degree from the University of Jyväskylä, in 2020, where he is currently pursuing the Doctoral degree with the Multiobjective Optimization Group. His research interest includes the interpretability aspects of interactive multiobjective optimization. This includes, but is not limited to, researching new ways to make interactive multiobjective optimization methods less opaque to the decision maker and analyst alike. In addition, he is interested in studying how to apply interpretable and explainable artificial intelligence to multiobjective optimization in general. He is also one of the main contributors to the DESDEO Framework.

B. S. SAINI received the M.Tech. degree from IIT Kharagpur, in 2018. He is currently pursuing the Doctoral degree with the Multiobjective Optimization Group, University of Jyväskylä. His research interests include multiobjective optimization, data visualization, data-driven optimization, and development of evolutionary algorithms. He has worked on many open source implementations of the methods from the aforementioned topics with a focus on modularity and interpretability. He is also one of the primary contributors to the DESDEO Framework.

B. AFSAR received the Ph.D. degree in computer engineering from Ege University, Izmir, Turkey, in 2014. He is currently a Postdoctoral Researcher with the Multiobjective Optimization Group, University of Jyväskylä. His main research interests include multiobjective optimization, data-driven multi-criteria decision-making, evolutionary computation, interactive multiobjective optimization methods and their applications, and multi-agent systems. He is also working on assessing the performance of interactive multiobjective optimization methods with both artificial and human decision-makers.

B. SHAVAZIPOUR received the Ph.D. degree in operations research from the University of Cape Town, Cape Town, South Africa, in 2018. He is currently a Postdoctoral Researcher with the Multiobjective Optimization Group, University of Jyväskylä. His principal research interests include multiobjective optimization and multi-criteria decision-making both in theory and applications, mathematical programming, data analysis and impacts upon the data envelopment analysis, scenario planning, and decision-making under (deep) uncertainty.

K. MIETTINEN received the Ph.D. degree in mathematical information technology from the University of Jyväskylä (JYU), Finland. She is currently a Professor of industrial optimization with JYU. She heads the Research Group on Multiobjective Optimization and is the Director of the thematic research area Decision Analytics utilizing Causal Models and Multiobjective Optimization (DEMO, jyu.fi/demo) at JYU. With her group, she develops an open source software framework for interactive multiobjective optimization methods (desdeo.it.jyu.fi). She has authored about 190 refereed journals, proceedings, and collection papers; edited 17 proceedings, collections, and special issues; and written a monograph on nonlinear multiobjective optimization. Her research interests include theory, methods, applications, and software of nonlinear multiobjective optimization. She is a member of the Finnish Academy of Science and Letters, Section of Science, and the Steering Committee of Evolutionary Multi-Criterion Optimization. She has been the President of the International Society on Multiple Criteria Decision Making (MCDM). She has received the Georg Cantor Award of the International Society on MCDM for developing innovative ideas. She belongs to the editorial board of seven international journals.

PVI

INTERACTIVE DATA-DRIVEN MULTIOBJECTIVE OPTIMIZATION OF METALLURGICAL PROPERTIES OF MICROALLOYED STEELS USING DESDEO

by

Bhupinder Singh Saini, Debalay Chakrabarti, Nirupam Chakraborti, Babooshka Shavazipour, Kaisa Miettinen

Submitted to a journal

Interactive Data-driven Multiobjective Optimization of Metallurgical Properties of Microalloyed Steels using the DESDEO Framework

Bhupinder Singh Saini[1,*], Debalay Chakrabarti[2], Nirupam Chakraborti[3,2], Babooshka Shavazipour[1], Kaisa Miettinen[1]

[1]University of Jyvaskyla, Faculty of Information Technology P.O. Box 35 (Agora), FI-40014 University of Jyvaskyla, Finland
[2]Department of Metallurgical and Materials Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, 721302, India
[3]Faculty of Mechanical Engineering Czech Technical University Prague, Czech Republic
* Corresponding author: Bhupinder Singh Saini, bhupinder.s.saini@jyu.fi

August 31, 2022

Abstract

Solving real-life data-driven multiobjective optimization problems involves many complicated challenges. These challenges include preprocessing the data, modelling the objectives, getting a meaningful formulation of the problem, and supporting decision makers to find preferred solutions in the existence of conflicting objectives. In this paper, we tackle the problem of optimizing the composition of microalloyed steels to get good mechanical properties such as yield strength, percentage elongation, and Charpy energy. We formulate a problem with six objectives based on data available and support two decision makers in finding a solution that satisfies them both. To enable two decision makers to make meaningful decisions for a problem with many objectives, we create the so-called MultiDM/IOPIS algorithm, which combines multiobjective evolutionary algorithms and interactive scalarization functions in novel ways. We use the DESDEO framework, an open-source Python framework for interactively solving multiobjective optimization problems, to create the MultiDM/IOPIS algorithm. We provide a detailed account of all the challenges faced while formulating and solving the problem. We discuss and

use many strategies to overcome those challenges. Overall, we propose a methodology to solve real-life data-driven problems with multiple objectives and decision makers. We successfully obtained microalloyed steel compositions with excellent mechanical properties using this methodology.

Keywords: Data-driven evolutionary computation; Multiobjective optimization; Surrogate-assisted optimization; Multiple decision makers; Interactive optimization; Open-source software.

1 Introduction

Digitalization sheds light on new data collection and information sharing methods, leading the world toward a data-centered era and opens up many opportunities for novel data-based methodology developments in data analytics and decision-making. However, various elements and challenges are involved in any decision-making process, starting from data.

Decision makers (DMs), in many real-life problems, often need to consider multiple criteria, called objectives, simultaneously when making decisions. In a decision-making process involving data, they first need to identify the objectives to be optimized with the help of the data available and the independent variables that control them. This process may require the data to be preprocessed. A multiobjective optimization problem (MOP) that considers the objectives important to the DMs can then be formulated. A DM is expected to be a domain expert. An analyst, who is an expert in multiobjective optimization, typically coordinates the formulation of the MOP with the DMs and supports in solving it.

One of the ways to solve data-driven MOPs is to use surrogate-assisted optimization algorithms [1, 2]. These algorithms use regression models, called surrogate models, to mimic the behavior of the objectives as recorded in the data. In this, we assume that the data has been obtained from phenomena that can be treated as objectives to be optimized. The choice of surrogate modelling techniques to model the objectives and the methods of training and validating the models significantly impact the solutions found by the optimization method.

MOPs generally do not have a single optimal solution. Instead, due to potentially conflicting objectives, there exist many so-called Pareto optimal solutions that reflect the trade-offs between the various objectives [3]. Many optimization algorithms aim to find a representative approximation of the Pareto optimal solutions, see, e.g. [3, 4]. However, such algorithms can return upwards of a thousand solutions to the DMs. Comparing many solutions, especially in problems with many objectives, can be a cognitively challenging task. Thus, without assistance, DMs may find it difficult to make decisions in such problems.

One of the ways to tackle this challenge is to use interactive methods for optimization [3]. Such methods incorporate the preferences of a DM during the optimization to focus the search of Pareto optimal solutions to be limited to ones that a DM prefers. Focusing the search in a smaller area helps interactive methods find Pareto optimal solutions quicker and reduces the number of

alternative solutions that a DM must consider at a time. Once the DM sees the solutions discovered using their preferences, they can select a solution they like as the final solution. Alternatively, if they do not like the solutions, they can provide new preferences to the method, signifying interest in a different set of trade-offs. The interactive method then provides them with new solutions that reflect the new preferences. This process enables the DM to learn about the possible trade-offs in the MOP in manageable and iterative steps, making finding preferable solutions easier.

We call the process of problem formulation, optimization and decision-making a seamless chain. We show a simple schematic of the steps involved in data-based decision-making in Figure 1. The figure presents a simple, linear pathway from data and modelling to multiobjective optimization and decision-making. However, solving real-life MOPs can be a more complicated process. It must involve lengthy deliberations between the DM and the analyst to formulate a meaningful MOP. The data may introduce constraints limiting which objectives the DMs can consider in their MOP and how the analyst can model such objectives. The optimization and decision-making steps may reveal significant issues in the problem formulation of modelling phases. This would require going back and fixing the issues in the earlier steps and conducting optimization and decision-making again. The presence of more than one DM can also complicate the consideration, as most interactive methods are designed for a single DM.

In this paper, we formulate and solve an MOP to obtain alloy compositions that optimize multiple physical properties of a microalloyed steel. To achieve this, we follow the seamless chain structure[1], described in Figure 1, to make the most efficient use of the data available. We discuss in detail the challenges we faced during the modeling and the solution process, and the steps we took to overcome them. We used many established and some new and novel techniques. In particular, we developed the interactive MultiDM/IOPIS algorithm, an extension of the IOPIS algorithm [5], to support two decision makers to find a solution that meets their different preferences. The new method can be applied for group decision making.

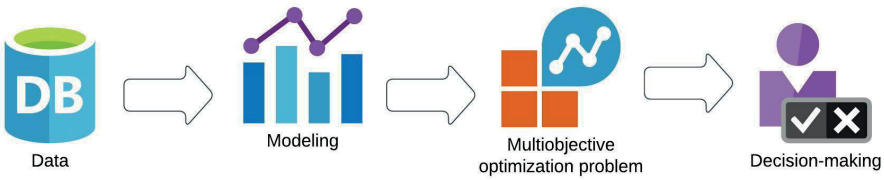


Figure 1: A schematic of the seamless chain from data to decision-making (adapted from `http://jyu.fi/demo`)

In Section 2, we establish the core background concepts. In Sections 3-6, we describe the various steps involved in formulating and solving the MOP.

[1]adapted from `http://jyu.fi/demo`

We start in Section 3 by describing the dataset which contains information about alloys of microalloyed steels. More specifically, the dataset contains the compositions and the corresponding values of various metallurgical properties of the alloys. We then discuss the tools and strategies used to preprocess the data to make it suitable for use in MOPs. Following this, in Section 4 we test a large number of surrogate modelling techniques to find the ones that work best with the data. In Section 5, using the results of the previous steps, and with the help of the DMs, we formulate meaningful MOPs to be solved. Finally, in Section 6 we use the open-source software framework DESDEO[2] [6] to implement the MultiDM/IOPIS algorithm and solve the MOPs interactively with two DMs. We also use two non-interactive optimization methods and compare the results.

It should be noted that the process of problem formulation and solution is usually an iterative one: we update the methodologies used in earlier steps based on the results obtained in the later steps. However, in Sections 3-6 we provide a linear narrative of the steps involved for the benefit of the reader. In Section 7, we discuss the effectiveness of the various steps we took to solve the MOP, including steps that did not succeed. We also discuss the insights gained via the seamless chain process. In doing so, we provide a general framework and a guideline to solve data-driven MOPs. Finally, in Section 8, we conclude and mention some future research directions.

2 Background

2.1 Multiobjective optimization

This paper considers the following form of a multiobjective optimization problem:

$$\begin{aligned} \text{minimize} \quad & \{f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ \text{subject to} \quad & \mathbf{x} \in S \subset \mathbb{R}^n, \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$ represents *decision vectors* (vectors of decision variables) in the feasible region S of the decision space $\mathbb{R}^n$. The number of objective functions is k and the vectors of objective function values, denoted by $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T$, are defined as *objective vectors* belong to the *objective space* $\mathbb{R}^k$.

Because the analytical form of functions is not available in data-driven optimization problems, the decision variables and their corresponding objective function values are often collected from simulators, real-life processes, or experiments and are only available in a dataset format. In such problems, approximation models, also called surrogates (or metamodels), are created, using the available data to approximate objective function values. Then, these surrogates are utilized to perform the optimization process. Surrogates may also be utilized as replacements for computationally expensive functions or simulators to reduce the evaluation time and save computations resources [7, 8]. When collecting new

[2]https://desdeo.it.jyu.fi

data is not possible during the optimization process, as in the case of this paper, the problem is called an offline data-driven optimization [9] problem. It means that the constructed surrogates cannot be updated with some new data.

Typically, in MOPs, identifying a single optimum is not possible because of the existing conflict between the objective functions. Instead, multiple (can be infinitely many) so-called Pareto optimal solutions exist, where improving any objective function value is impossible without impairment in at least one of the other objective function values. The set of Pareto optimal objective vectors in the objective space is called a Pareto front. DM needs to compare Pareto optimal solutions, study the existing trade-offs between objectives, and choose the most preferred one based on their preferences. Besides the DM, an analyst (or a group of analysts) is responsible for performing the analyses, computations, and supporting the interactive decision-making process. Generally, an analyst can be a human or a computer program [10].

Based on when preference information is incorporated, multiobjective optimization methods can be classified as a priori, a posteriori, and interactive methods [3]. DM provides their preferences before and after the solution process, respectively, in a priori and a posteriori methods. Providing unrealistic preferences is the major shortcoming of the a priori method, particularly if the DMs do not have prior deep insight into the problem. On the other hand, computational cost of generating a representative set of Pareto optimal solutions and heavy cognitive loads of many comparisons are the main difficulties in using the a posteriori methods, especially when there are many objectives [11, 12].

When the DM actively directs the solution process by providing preferences iteratively, the multiobjective optimization method is called interactive [3, 10]. In this way, the DM can learn about the interdependencies of the objectives in the problem and the feasibility of their preferences. Furthermore, these methods limit both cognitive and computational load since only a limited amount of information needs to be analyzed at a time, and only solutions reflecting the DM's preferences need to be generated, respectively. The DM can pursue the interactions by adjusting the preferences until they get satisfied and converge to the most preferred solution (see [10] for more details).

There are different ways to solve MOPs (see, e.g., [3, 4, 10]); among them, one widely used approach is to transform the MOP into an equivalent single-objective problem utilizing a so-called scalarization function while incorporating DM's preferences [3, 13, 14]. The scalarized MOP can then be solved using an appropriate single objective optimizer, resulting in one or more Pareto optimal solutions that reflect the preferences to satisfy the DM.

Another way to solve MOPs is to use multiobjective evolutionary algorithms (MOEAs) [4, 12]. They are metaheuristic approaches which use a "population" of solutions simultaneously to mimic the process of evolution. Popular MOEAs such as RVEA [15] and NSGA-III [16] have proven to be successful at generating a representative set of Pareto optimal solutions for MOPs with many objectives. However, MOPs become exponentially more difficult to solve with an increasing number of objectives [11]. The DESDEO framework provides open-source and

modular Python implemenetations of RVEA and NSGA-III (and many other MOEAs), as well as many scalarization functions [6].

The IOPIS algorithm [5] provides a middle ground between using scalarization functions (which optimize in a single dimension) and MOEAs (which generally optimize in the objective space with many dimensions). The IOPIS algorithm incorporates a DM's preferences using multiple scalarization functions (typically fewer than the number of objectives in the original problem). Together, these functions form a new space called a preference incorporated space (PIS). An appropriate MOEA is then used to optimize in this new space, making the MOEA interactive (since preferences are incorporated). An analyst can therefore control the number of dimensions in which the MOEA optimizes by changing the number of scalarization functions, which form the PIS.

The IOPIS algorithm is of note as it enables easy and modular creation of interactive MOEAs. Moreover, as shown in [5], it guarantees that the interactive MOEAs will have the beneficial properties of optimality (the solutions found by the MOEA are Pareto optimal), preferability (the solutions found by the MOEA follow the preferences of the DM), and searchability (the MOEA enables the DM to find any Pareto optimal solution by changing the preferences). However, the IOPIS algorithm was originally designed for solving MOPs with a single DM.

2.2 Microalloyed steels

As mentioned, we consider a data-driven problem of microalloyed steel. Steels used for structural, linepipe and naval applications must meet strict performance standards and withstand mechanical stresses imposed in such applications without failure. Microalloyed steels [3] exhibit the required high strength, toughness, ductility, and weldability capacity [17]. Entities such as the British and European standards (BS EN standards), the American petroleum institute (API standards), and the US naval sea systems command (MIL standards) have published standards for usage of microalloyed steels in various domains. Each published standard includes one or more grades of microalloyed steel which set the minimum requirements and maximum bounds that should be met by the steel for specific applications.

Yield strength (YS) and ultimate tensile strength (UTS) are two important measures of the strength of materials. The YS measures the maximum stress (force per unit area) a material can sustain before undergoing permanent (plastic) deformation. Below the YS, the material deforms elastically, i.e., it reverts to its original shape and size after removing the external force. The UTS measures the maximum stress a material can sustain before undergoing a fracture. Percentage elongation (ELON) measures the ductility as the fractional increase in the length of a specimen that undergoes fracture upon reaching the UTS.

We can use Charpy impact tests to measure the toughness of materials at various temperatures. The test measures the energy required to fracture a standard specimen made from the material using an impact. The Charpy impact

[3] A subcategory of high-strength low-alloy steels.

energy generally decreases at lower temperatures as ductile materials (such as steels) start showing brittle behavior at such temperatures. Good toughness at low temperatures is required for steels used at such temperatures. The result of these tests can be reported as Charpy energy values at specific temperatures or impact transition temperature (ITT) values at specific Charpy energy values.

In minute quantities, alloying elements such as titanium, molybdenum, and vanadium generally positively affect the strength, ductility, and toughness of microalloyed steels. However, they can hinder the ability of structures made from such steels to be welded. The weldability of steels is correlated with the carbon equivalent (C_{eq}), which can be calculated as [18]:

$$C_{eq} = \%C + \frac{\%Mn + \%Si}{6} + \frac{\%Cr + \%Mo + \%V}{5} + \frac{\%Cu + \%Ni}{15}. \quad (2)$$

The terms on the right-hand side of (2) measure the concentration of the corresponding alloying elements as a mass percentage. Various steel grades set different upper limits to the acceptable levels of C_{eq} to account for weldability needs.

3 Data Description and Preprocessing

The starting point of the consideration is a set of data. The raw dataset, compiled from a database and available in the DESDEO framework, contains details about the metallurgical properties of microalloyed steels. There are 736 rows and 51 columns in the dataset. Each row denotes information related to microalloyed steel of a specific composition, which we later refer to as a sample. The first twenty columns contain information about the concentration of various alloying elements. The rest of the columns contain information about various metallurgical properties of the samples. The first three among these are YS, UTS, and ELON. The next ten columns denote the ITT at ten different Charpy impact energy levels (ranging from 13 J to 80 J). The next six columns measure the Charpy impact energy value at different temperatures (ranging from -80°C to 19°C). The remaining columns contain properties such as fracture toughness, pearlite content and hardness.

As in may cases with real data, there are many issues with the raw dataset. As the raw dataset is a compilation of data from various sources, the data quality is not consistent across the rows. For example, some cells have exact numbers, whereas others have a range. Moreover, measurements of properties such as the Charpy energy are very noisy. While the raw dataset contains 736 rows of samples, many cells are empty in various columns. Consequently, many rows do not completely contain information about the samples' composition, grain size, and metallurgical properties. For example, there are only 599 samples that measure YS, 537 UTS measurements, and 296 ELON measurements. The number of rows that contain information about other metallurgical properties is much lower. Moreover, these measurements are spread across the data points such that the overlap in rows that measure two different metallurgical properties

is very small. This means that the different properties are measured for entirely different alloy compositions, with minimal overlap.

3.1 Preprocessing the Data

We cleaned and divided the raw dataset into multiple sets to be used in later steps. Firstly, all empty cells in the alloy composition columns were assigned a value of zero. As the dataset was collected from various studies, empty cells signify that those alloying elements were not a part of the study, and hence did not exist in the alloy. Secondly, we merged the columns named "Nb" (niobium) and "Cb" (columbium) as they refer to the same alloying element. Finally, we replaced the cells in the alloy composition columns that were represented as a range of values by the average value of the range. At this step, all cells in the alloy composition columns had numerical values.

We then divided the dataset into multiple sets such that each set contained all the alloy composition columns but only one metallurgical property. For this, we only considered rows which documented the respective metallurgical property. This led to, for example, a YS dataset with 599 samples. We repeated the process for UTS and ELON. None of the columns documenting ITT or the Charpy energy had more than 300 samples. Hence, instead of breaking these columns into multiple small sets, we combined the 10 ITT columns and the 6 Charpy energy columns into just two columns documenting the temperature and the corresponding Charpy energy[4]. This process led to a Charpy dataset with 781 samples. We used the Pandas Python package to carry out the aforementioned tasks [19].

4 Surrogate Modelling

4.1 Model selection and training

To begin the modelling process, we first analysed the dataset to identify which alloying elements significantly impacted the various metallurgical properties. We compared the results of the feature importance analysis against metallurgical literature to confirm the reliability of the data. In brief, we conducted the following tests to identify significant alloying elements for the YS, UTS, Elongation, and Charpy datasets using the `scikit-learn` Python package [20]:

- Principal component analysis (PCA): This method can help us find "features" (alloying elements in our case) that have the most variance in the dataset.
- Cross decomposition: We use the PLSCanonical, PLSSVD, PLSRegression, and Canonical Correlation Analysis (CCA) algorithms to find out

[4]This process is known is data "melting" and converts "wide" data (more columns, fewer rows) to "tall" data (fewer columns, more rows)

which features lead to the most variance in the various metallurgical properties.

- Random forest regression: We calculate feature importance (FeatImp) and permutation importance (PermImp) using random forest models trained on the dataset.
- ANOVA tests: We rank various features according to their F-values.
- Mutual importance (MI): We rank the various features according to their MI values.

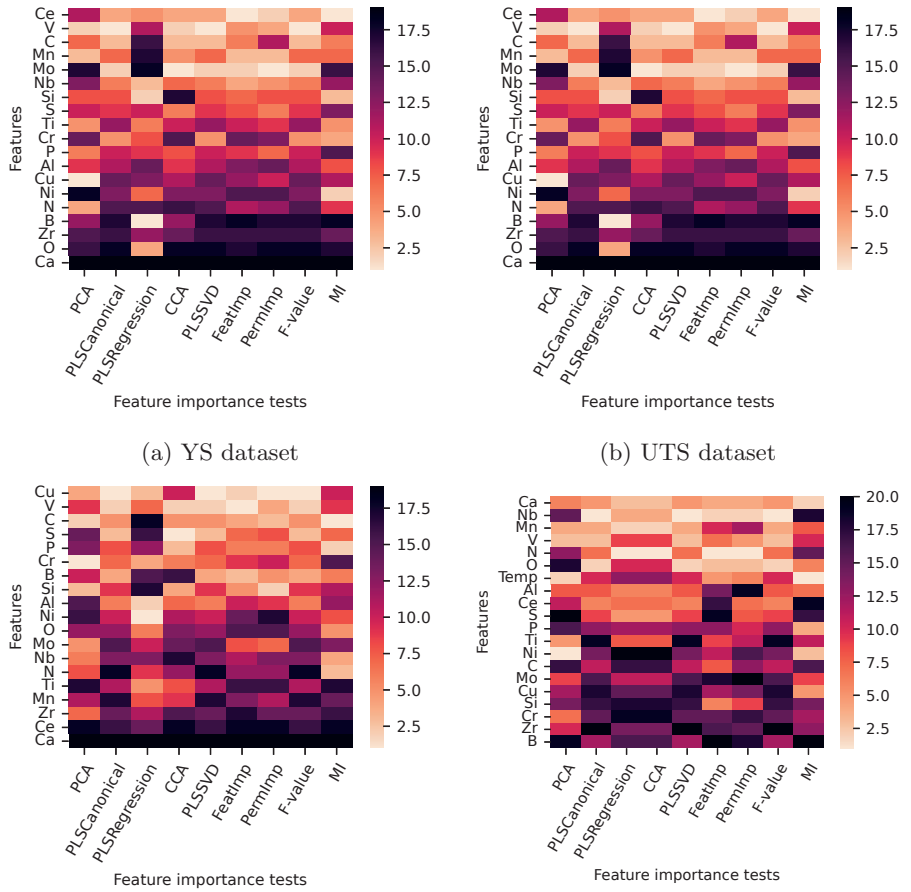
The results of the tests are presented in Figure 2 as heatmaps. The x-axis in each of the sub-figures denotes the aforementioned tests, whereas the y-axis represents the alloying elements (and temperature values for the Charpy dataset). The color of the heatmap represents the relative importance (calculated as a rank) of the alloying element as measured by each test. Elements with lighter hues achieved a better rank and are considered more important by the tests.

There are various reasons why a test may consider a certain alloying element unimportant for a given mechanical property. The alloying element may have no effect, or a mixed effect on the mechanical property. For example, nitrogen can form nitrides with titanium, which can lead to grain refinement, which leads to better YS. However, in the presence of aluminium, it can form AlN, which can lead to embrittlement of the steel, which lowers the YS. Hence, nitrogen can have a mixed effect on YS, and is ranked low by the tests. The dataset may also have a very skewed distribution of the values of alloying element concentrations. In the UTS dataset, only 25 (out of 537) samples contained Zirconium. The effect of such elements may not be adequately represented in the dataset, which can lead to a low rank. Finally, certain alloying elements may show a mixed effect on the mechanical properties because of noise in the dataset.

Based on these tests, we removed unimportant or noisy columns from the datasets. This can increase the overlap in the ranges of alloying element compositions for which the various properties are measured and lead to better modelling and optimization results. We discuss this further in Section 5.

We trained metamodels for YS, UTS, ELON, and Charpy energy based on their respective datasets using many surrogate modelling algorithms and compared their training accuracy using the R^2 value. We considered the following surrogate modelling algorithms: neural networks [21], support vector machines [22], Gaussian process regression [23, 24], and various ensemble modelling techniques. These surrogate modelling algorithms have been used extensively and successfully in data-driven multiobjective optimization [2]. We conducted K-fold cross-validation to choose the best performing surrogate modelling technique for each metallurgical property. Details of the test, along with a Python implementation, can be found in the DESDEO framework.

The results of the test are shown in Table 1. In general, ensemble techniques worked better than other surrogate modelling techniques tested. The extra trees regression [25], gradient boost regression [26] and random forest regression [27]



(a) YS dataset

(b) UTS dataset

(c) Elongation dataset

(d) Charpy dataset

Figure 2: Relative ranking of the features calculated for all datasets. A lower rank value (lighter color in the heatmap) represents higher importance.

(from Python package `scikit-learn`), XGBoost [28] (from Python package `xgboost`, and LightGBM [29] (from Python package `lightgbm`) performed the best. As the Charpy dataset was much noisier than the other datasets, the surrogate models for Charpy energy performed much worse. The model with the highest median R^2 value was chosen for each mechanical property, as highlighted in bold in Table 1.

4.2 Model validation

In the absence of the ability to create and test new alloys, we can validate the models by comparing the effect of changing the alloying element compositions on

Surrogate Modelling Techniques	YS		UTS		Elongation		Charpy Energy	
	Median R^2	Standard Deviation	Median R^2	Standard Deviation	Median R^2	Standard Deviation	Median R^2	Standard Deviation
ExTR	**0.746**	**0.059**	0.8380	0.0709	**0.673**	**0.124**	0.166	0.129
Ada	0.604	0.0818	0.7403	0.154	0.575	0.126	0.293	0.0724
Bagging	0.716	0.0492	0.8358	0.0839	0.612	0.108	0.275	0.109
GradBoost	0.712	0.0619	0.8437	0.0629	0.666	0.177	**0.436**	**0.0843**
XGBoost	0.742	0.0694	**0.8440**	**0.0781**	0.58	0.128	0.178	0.117
XGBRF	0.642	0.0571	0.7678	0.0932	0.651	0.136	0.423	0.0765
LightGBM	0.732	0.0511	0.7833	0.0587	0.642	0.141	0.305	0.103
RandomForest	0.726	0.037	0.8307	0.0679	0.64	0.12	0.317	0.0994
Kriging	-204	1920	-85.18	2030	-509	835	-6.5	3.6
SVM1	-0.0253	0.0435	-0.03087	0.0352	0.172	0.0707	-0.0803	0.11
SVM2	-0.232	0.14	-0.3843	0.143	0.0344	0.134	-0.0244	0.184
NN	-6.79	1.12	-11.35	7.29	-0.452	0.259	-0.0519	0.113

Table 1: The K-fold cross-validation performance (R^2 score) of various surrogate modelling techniques on YS, UTS, Elongation, and Charpy datasets.

the models' responses. We can do so using individual conditional expectation (ICE) plots [30]. They plot the changes in the output of a surrogate model based on changes in one of the input variables, which are the alloying element concentrations in our problem. The effect of alloying elements on the mechanical properties is presented in Table 2.

Alloying element	Effect on YS	Effect on UTS	Effect on ELON	Effect on Charpy energy
C	positive	positive	negative	mixed
Si	mixed	positive	negative	-
Mn	positive	positive	mixed	positive
P	negative	negative	negative	negative
S	mixed	negative	negative	negative
Mo	positive	positive	negative	positive
Ni	positive	positive	negative	mixed
Al	negative	negative	positive	positive
N	-	-	mixed	negative
Nb	positive	positive	positive	positive
V	positive	positive	negative	negative
B	-	-	mixed	-
Ti	positive	positive	negative	mixed
Cr	positive	positive	negative	negative
Ce	positive	positive	-	-
Cu	negative	positive	negative	-
Zr	negative	negative	mixed	-

Table 2: Effect of alloying element composition on mechanical properties calculated using ICE plots. The alloying elements with no reported effects were not considered for the model.

In the table, we divide the effect of changing alloying element concentrations into four categories. A positive effect means that the alloying element benefits the property modeled by the surrogate model, whereas a negative effect implies a negative correlation. A mixed effect implies that the alloying element has a complex relationship with the modeled property. It can have a beneficial or a detrimental effect based on other criteria, for example, the concentration of other alloying elements. In contrast, a nil effect (signified with "-") means that the alloying element has no effect on the surrogate model, or its effect could not be detected. We present a complete discussion of the effect of all alloying elements in Appendix A. The discussion is based on an extensive review [31–54] covering the effects of alloying elements on strengths (YS and UTS), ductility (ELON), and impact toughness (Charpy impact energy absorption at different test temperatures) in a variety of low-carbon steels used for structural, linepipe, automotive, naval, and pressure vessel applications.

The DMs concluded that the metallurgical theory and literature back a majority of ICE plot results. The models can be deemed valid and be used to study steels' mechanical behaviour. We can see in Table 2 that many elements have conflicting effects on YS, UTS, Elongation, and Charpy energy. This is ultimately the source of trade-offs in MOPs that consider such objectives.

5 Problem Formulation

Based on the consideration described in the previous section, the selected surrogate models (ExTR for YS and ELON, XGBoost for UTS, and GradBoost for Charpy energy), can predict the values of YS, UTS, ELON, and Charpy energy value of vanadium microalloyed steels as functions of their compositions (and additionally the temperature for the Charpy energy). Thus, the metallurgical properties form the objectives of an MOP to be optimized with the alloy composition as the decision variables. Besides the surrogate models, two more objectives were added to the problem formulation: carbon equivalent value and the cost of materials. These objectives can be numerically calculated as functions of the steel composition. Equation (2) is used to calculate the carbon equivalent value. The cost of materials is calculated as a linear combination of the alloy composition values weighted by the cost of the elemental components as stated in Table 3.

We formally define the MOP (MOP-I) as:

$$\begin{aligned} \text{maximize} \quad & \texttt{YS}(\mathbf{comp}) \\ \text{maximize} \quad & \texttt{UTS}(\mathbf{comp}) \\ \text{maximize} \quad & \texttt{ELON}(\mathbf{comp}) \\ \text{maximize} \quad & \texttt{Charpy}(\mathbf{comp}, temp) \\ \text{minimize} \quad & C_{eq}(\mathbf{comp}) \\ \text{minimize} \quad & Cost(\mathbf{comp}) \\ \text{subject to} \quad & \mathbf{LB} \leq \mathbf{comp} \leq \mathbf{UB}, \end{aligned} \tag{MOP-I}$$

where `YS`, `UTS`, `ELON`, and `Charpy` are the surrogate models for YS, UTS, ELON,

Alloying Element	Lower bound (MOP-I)	Upper bound (MOP-I)	Lower bound (MOP-II)	Upper bound (MOP-II)	Cost (USD per Kg)
C	0.009	0.34	0.006	0.4	0
Si	0.01	0.55	0	0.6	0.122
Mn	0.28	1.63	0.28	1.63	1.7
P	0	0.035	0	0.035	1.82
S	0	0.035	0	0.042	2.69
Mo	0	0.53	0	1.0	0.0926
Ni	0	3.12	0	3.12	40.1
Al	0	0.06	0	0.06	13.9
N	0	0.024	0	0.024	1.79
Nb	0	0.086	0	0.45	0.140
V	0	0.2	0	0.2	72
B	0	0.001	0	0.001	3.68
Ti	0	0.18	0	0.31	11.5
Cr	0	1.24	0	1.5	9.4
Ce	0	0.015	0	0.015	4.6
Cu	0	0.312	0	0.312	6
Zr	0	0.008	0	0.008	237

Table 3: The upper and lower bound for the concentrations (in percentage weight) of the alloying elements and their cost as used in MOP-I and MOP-II.

and the Charpy energy, respectively. The decision variable **comp** is the vector of the concentration of 17 alloying elements in the steel bounded by lower and upper bounds **LB** and **UB**. The function C_{eq} refers to the carbon equivalent value, and the $Cost$ function refers to the cost of materials. The temperature $temp$ input for the `Charpy` surrogate model is kept constant at $-80°C$. The bounds **LB** and **UB** are calculated as the bounds of the intersection of the datasets used for all four surrogate models and the values are given in Table 3. Expanding this range makes the search space larger, which increases the likelihood of finding suitable alloy compositions. This is why removing unimportant or noisy columns was a crucial step in surrogate modelling, as described in Section 4.

We create a simpler version of MOP-I (named MOP-II) by removing the Charpy energy objective from MOP-I. We consider this version since, as mentioned, the `Charpy` surrogate model had a worse accuracy than the other surrogate models. Removing this objective also led to an expansion of the bounds of the decision variables (calculated as the bounds of the intersection of the remaining datasets). The upper and lower bound values for the various alloying elements can be seen in Table 3.

6 Optimization

The open-source software framework DESDEO [6] provides a variety of interactive and some non-interactive methods to solve multiobjective optimization problems We begin by solving the two problems formulated in the previous section with two non-interactive evolutionary algorithms: RVEA and NSGA-III, to generate an approximate representation of Pareto optimal solutions. These method have been reported to work well with problems with more than four objectives. The details of the optimization and the results are discussed in Subsection 6.1. We conducted optimization using non-interactive MOEAs and presented visualizations of the results to the DMs to help them form their initial preferences.

In Subsection 6.2, we outline the details of the interactive MOEA called MultiDM/IOPIS, a method developed to solve the MOP interactively with more than one DM. We then describe the interactive optimization process itself and the results obtained in Subsection 6.3.

6.1 Non-interactive optimization

We ran the RVEA and NSGA-III algorithms for 150 generations each to solve MOP-I and MOP-II. There was no improvement in the objective values attained by the solutions after around 120 generations. The other parameters of the algorithms were otherwise unchanged from the values suggested in original publications which proposed RVEA and NSGA-III. We combined the final solutions from both RVEA and NSGA-III and removed the ones where some objective function value was worse and other values not better than in some other solution. Figures 3 and 4 show the remaining solutions for MOP-I and MOP-II, respectively, in parallel coordinates plots. Interactive versions of these plots can be seen at `https://desdeo.it.jyu.fi`.

The two methods found 1055 solutions for MOP-I and 1599 solutions for MOP-II. The ideal, i.e., best possible, values for the objectives of MOP-I were (726 MPa, 1577 MPa, 34%, 103 J, 0.087, 43 USD per kg). The ideal values for the objectives of MOP-II were (752 MPa, 1593 MPa, 34%, 0.055, 42.9 USD per kg). The methods were able to find slightly better solutions for MOP-II than for MOP-I. This may be because optimization becomes exponentially more difficult with an increasing number of objectives. As the parameters of the methods were the same for solving MOP-I and MOP-II, the results were slightly worse in the version with one more objective.

Despite the differing performance, certain similarities can be visually observed in the solutions of MOP-I and MOP-II. For example, in both cases, there is a cluster of solutions with very high `UTS` values, followed by a discontinuity, and then a different cluster of medium and low `UTS` values. There are also three clusters in the *Cost* objective. In both problem formulations, *Cost* did not have significant trade-offs with the `ELON` objective. The solutions represented high values for `ELON` for both low *Cost* and high *Cost* alloys. However, *Cost* was very strongly correlated with `UTS`. Low *Cost* alloys could only achieve `UTS`

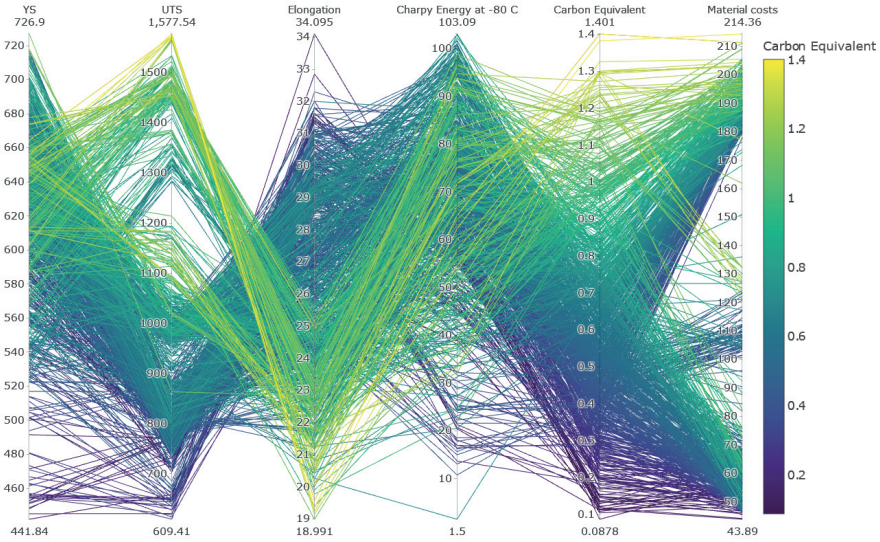


Figure 3: Solutions of MOP-I presented as a parallel coordinates plot. The solution traces are coloured based on the Carbon equivalent value.

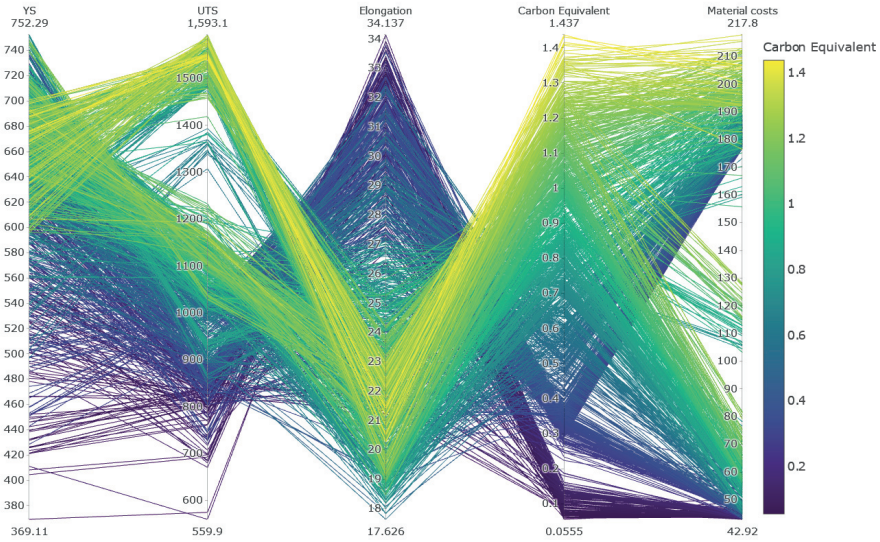


Figure 4: Solutions of MOP-II presented as a parallel coordinates plot. The solution traces are coloured based on the Carbon equivalent value.

values of up to 1200 MPa. Only the highest *Cost* alloys could achieve the best `UTS` values. Increasing *Cost* also generally increased the `YS` values. However, the low *Cost* alloys could still achieve the best `YS` values. Among the solutions of MOP-I specifically, the `Charpy` objective was weakly conflicting with all other objectives.

Standard	Grade	Solutions (MOP-I)	Solutions (MOP-II)
BS EN-10025-4	S275ML	2	47
BS EN-10025-4	S355ML	2	47
BS EN-10025-4	S420ML	5	45
BS EN-10025-4	S460ML	5	17
BS EN-10149-2	S315MC	7	132
BS EN-10149-2	S355MC	8	149
BS EN-10149-2	S420MC	9	226
BS EN-10149-2	S460MC	7	182
BS EN-10149-2	S500MC	7	175
BS EN-10149-2	S550MC	3	133
BS EN-10149-2	S600MC	2	84
BS EN-10149-2	S650MC	1	58
BS EN-10149-2	S700MC	0	22
BS EN-10025-6	S460QL1	5	39
BS EN-10025-6	S500QL1	1	28
BS EN-10025-6	S550QL1	1	12
BS EN-10025-6	S620QL1	0	1
API - 5L - 2018	X42M	1	2
API - 5L - 2018	X46M	1	2
API - 5L - 2018	X52M	1	2
API - 5L - 2018	X56M	1	1
API - 5L - 2018	X60M	1	0
API - 5L - 2018	X65M	1	0
API - 5L - 2018	X70M	1	0
API - 5L - 2018	X80M	1	0
MIL-S-24645A(1984)	HSLA-80	0	0
MIL-S-24645A(1989)	HSLA-100	0	0

Table 4: Number of solutions of MOP-I and MOP-II that match structural (BS EN), linepipe (API), and naval (MIL) steel standards.

We compared the solutions of MOP-I and MOP-II against structural and pipeline steel grades. We enumerate the solutions that match various steel grades in Table 4. The MOP-II formulation resulted in many more feasible alloy compositions for structural steels than the MOP-I formulation. This is because, as mentioned earlier, MOP-II is easier to optimize than MOP-I. However, for linepipe steels, the MOP-I formulation discovered alloy compositions

to satisfy a more diverse range of grades than MOP-II. The inclusion of the `Charpy` energy objective in MOP-I allowed a more diverse search space. Hence, the formulation was able to satisfy more grades. Neither formulation was able to discover solutions that matched the naval steel grades. This signifies that RVEA and NSGA-III were not able to find solutions in all regions of the Pareto front.

6.2 MultiDM/IOPIS

As mentioned earlier, optimization with MOEAs to approximate the entire Pareto front becomes increasingly more challenging with more objectives. Interactive MOEAs resolve this issue by using the preferences of a DM to narrow down the search to solutions that are of interest to the DM, thus, focusing computational resources in a smaller region. As mentioned, in this study, there were two domain experts who acted as DMs, whereas one of the authors acted as an analyst to guide them through the interactive decision making process. As the number of objectives is not an issue for the interactive MOEAs, only MOP-I was solved.

As mentioned, IOPIS was originally proposed for a single DM. We developed and implemented a new variant of the IOPIS algorithm [5], called MultiDM/IOPIS, to support collaborative interaction with multiple DMs simultaneously, that is, group decision making. The new algorithm provides this support while maintaining the beneficial properties of the original IOPIS algorithm, i.e., the guarantees of optimality, preferability, and searchability. The original IOPIS algorithm uses multiple scalarization functions, which take the same preference information, to convert an MOP to a new MOP with a lower number of objectives. The new MOP, with the preference information as a fundamental building block, allows non-interactive MOEAs such as RVEA and NSGA-III to focus on the region of interest of the DM and, thus, converts these non-interactive methods as interactive ones.

MultiDM/IOPIS reverses this concept by using copies of the same scalarization function, taking different preferences as input. The multiple preferences can come, for example, from multiple DMs, as in our case. MultiDM/IOPIS, thus, creates a new MOP with the same number of objectives as the number of DMs, regardless of the number of objectives in the original MOP. In our case, the number of DMs (two) is much smaller than the number of objectives (six), making the new MOP formed by MultiDM/IOPIS much easier to solve.

In brief, the interactive optimization process with multiple DMs applying MultiDM/IOPIS involves the following steps: 1) The DMs provide individually their own reference points reflecting their desired values for the objectives. 2) The new MOP is formed, where the objective functions are scalarization functions containing the preferences of one of the DMs. 3) The MOP is solved with an MOEA to get a set of solutions to be shown to the DMs. 4) The DMs analyze the solutions to see how their own and the preferences of the others are reflected. If a satisfactory solution is found, stop. Otherwise, go to 1).

Let us elaborate the behaviour of MultiDM/IOPIS in the case of two DMs (for simplicity). It has the following modes of operation:

1. Case 1. The DMs provide extremely different preferences: The MOEA converges towards the two solutions that best satisfy the two preferences individually, as well as many solutions that present a compromise between the preferences of the DMs. The inclusion of compromise solutions provides necessary information to the DMs to collaborate on a compromise.

2. Case 2. One DM explores the objective space by changing preferences but the other DM stays anchored: The MOEA converges further and provides better solutions near the region of interest of the DM that did not change preferences. At the same time, the MOEA discovers new solutions to reflect the new preferences given by the first DM. The MOEA also provides compromise solutions between the two regions of interest. This mode allows for quick but controlled exploration of potential solutions: discovery of new solutions in new regions of interest and the corresponding trade-offs with the anchored preferences. Thus, even the anchored DM can find new, potentially favourable solutions.

3. Case 3. The DMs provide preferences close to each other: The DMs are expected to arrive at a favourable compromise during the interactive optimization process. The MOEA returns solutions in a narrower region as the preferences draw closer, providing many solutions with minor variations in the objective values. This variety allows the DMs to choose a finely-tuned solution for their application.

To solve our problem, we applied MultiDM/IOPIS with the scalarization function from the STOM method [55] in our implementation of MultiDM/IOPIS (it was already available in DESDEO) and NSGA-III to solve the resulting biobjective optimization problem. NSGA-III generally performs better than RVEA in MOPs with a lower number of objectives [56]. The STOM scalarization function takes a DM's preferences as a reference point: a vector of objective values (for all objectives) that the DM aspires to achieve. In the interactive optimization process, the DMs can change the reference points at any time, resulting in MultiDM/IOPIS creating a new MOP which reflects the new preferences. However, the population from the MOEA continues its evolution from the previous MOP. Thus, no progress on optimization is lost.

We set the population size of NSGA-III to be 50. While this is an inadequate size for a problem with six objectives, we found that it is sufficient for the modified MOP generated by MultiDM/IOPIS. Moreover, we only ran the MOEA for 30 generations between iterations, i.e., asking the DMs for preferences and providing near-Pareto optimal solutions to analyze and update their preferences. In our experiments, the small number of generations and population size led to a near-instantaneous return of solutions and high computational efficiency.

6.3 Interactive Optimization Process

We started the process of interactive optimization by first showing the solutions of the non-interactive optimization methods to the DMs (Figures 3 and 4) in interactive plots that allowed brushing and filtering of solutions. This process enabled them to explore the approximated Pareto optimal solutions to learn about the trade-offs and form their initial preferences. The analyst introduced MultiDM/IOPIS to the DMs by describing how they can steer the interactive optimization process with their preference information (and the three modes of operation).

Preferences in the first iteration	DM 1: (460, 550, 17, 27, 0.47, 50) Similar to structural grade S460QL1 DM 2: (690, 770, 14, 27, 0.65, 80) Similar to structural grade S690QL1
Preferences in the second iteration	DM 1: (552, 600, 20, 81, 0.6, 50) Similar to naval grade HSLA-80 DM 2: (690, 770, 14, 27, 0.65, 80) Similar to structural grade S690QL1
Preferences in the third iteration	DM 1: (552, 600, 20, 81, 0.6, 50) Similar to naval grade HSLA-80 DM 2: (690, 770, 18, 81, 1, 80) Similar to naval grade HSLA-100

Table 5: Preferences given by the two DMs during interactive optimization using MultiDM/IOPIS in the form of reference points. The values of components of each reference point represents the preferences for objectives in the order (`YS`, `UTS`, `ELON`, `Charpy`, C_{eq}, $Cost$).

We show the first reference points given by the two DMs in Table 5. The first DM set the reference point to target solutions to match the structural steel grade S460QL1 and the second DM did the same for S690QL1. During non-interactive optimization, the MOP-I formulation resulted in only five solutions to match the S460QL1 grade. The MOP-II formulation resulted in 39 solutions that matched that grade. Neither formulation had resulted in any solutions for the S690QL1 grade, as it is a stricter grade.

The analyst visualized the solutions generated based on the two reference points to the DMs in a parallel coordinate plot, see green lines in Figure 5. We combine the solutions of all iterations in Figure 5 for the sake of brevity. The solutions of the first iteration vastly exceeded the first DM's preferences. The solutions closely matched the second DM's much stricter preferences, while still giving much better values for the `Charpy` objective compared to the reference point. No solution discovered during the non-interactive optimization could match the solutions found in the first iteration of the interactive optimization in all six objectives.

For the second iteration, the presence of solutions with very good `Charpy`

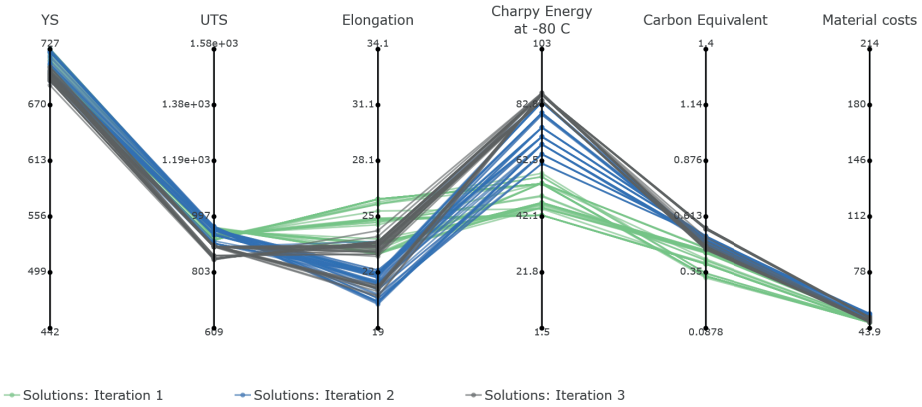


Figure 5: Results of interactive optimization using MultiDM/IOPIS. The results from three iterations are presented in green (iteration 1), blue (iteration 2), and grey (final iteration).

values led to the first DM changing preferences to a much stricter naval steel grade: HSLA-80. The second DM chose not to change preferences to preserve the newly discovered solutions for further consideration. The exact values of the two reference points are presented in the second row of Table 5.

We visualize the solutions of the second iteration in Figure 5 in blue. The most apparent differences between the solutions of the two iterations were in the `ELON` and `Charpy` objective values. The `Charpy` values were much better for solutions of the second iteration, which came at the cost of worse `ELON` values (although still better than the given preferences). This change was because of the much higher value for the `Charpy` objective in the reference point given by the first DM. The equivalent carbon content C_{eq} for the solutions of the second iteration was also higher but still well within the grade specifications. There was a small number of solutions that satisfy the preferences given by the first DM (higher `Charpy` and `ELON`), and a small number that met the preferences given by the second DM (higher `YS` and `UTS`). A majority of solutions represented the trade-offs between the two preferences.

Based on the results of the second iteration, the second DM decided to change the preferences to a much stricter naval steel grade HSLA-100. The reference points for the third iteration (in the third row of Table 5) were very close to each other: the two DMs were coming into agreement regarding their preferences.

The solutions of the third iteration, shown in Figure 5 in grey, had a narrower spread compared to the previous iterations. A significant trade-off this time was present between the `UTS` and `ELON` objectives. This trade-off originated from the differences in the two reference points. Even though the second DM allowed for a much higher *Cost* value of 80 USD/kg, MultiDM/IOPIS achieved satisfactory values for all other objectives at close to half of the *Cost* value. The DMs

were satisfied with the solutions obtained and decided to end the interactive optimization process. The solutions of the third iteration were very similar and the solution with the objective values (700, 846, 24.2, 87, 0.470, 47) was chosen. This solution is better than the preferences provided by either DM as seen in Table 5, which is why it was chosen. The complete set of all solutions found by MultiDM/IOPIS in all three iterations can be found via the DESDEO framework.

7 Discussion

MOPs and their solution processes are often presented straightforwardly in the literature: problem formulation (often given without details of modeling), followed by optimization, visualization and discussion of results. It provides a straightforward approach to give an account of the problem and the steps taken to solve it. However, it hides many complexities of the challenges faced in solving real MOPs, particularly data-driven MOPs. In Sections 3-6, we provided a detailed account of all the steps taken to formulate and solve MOPs in the case considered. However, we did not discuss why some of those steps were necessary. This is because, for example, we took the measures taken during the first steps of solving the MOP (i.e., data preprocessing) to solve issues that arose during the last steps (i.e., optimization). We discovered many fundamental issues while solving the MOP, which required us to rethink our approach, apply fixes to earlier steps, and restart the solution process. An explanation of such issues and their resolutions requires the knowledge of the entire solution process.

After Sections 3-6, having discussed the entire solution process, we can now elaborate on the challenges we faced and justify the choices we made to solve them. In this section, we also talk about approaches that lead to dead-ends and, therefore, did not warrant a mention in the previous sections.

The first issue we faced was converting the data into a usable format. The naive approach of simply training surrogate models on the raw datasets (mostly empty cells) would lead to issues related to extrapolation (models giving bad predictions in areas outside the bound of a given objective but within the bound of the dataset). Sometimes, the models trained in such a manner also led to runtime errors. Thus, cleaning and preprocessing the data were needed. Dividing the dataset into individual datasets for each objective also allowed us to calculate each objective's domains (ranges of decision variable values). This information helped us define the constraints of the MOP during the later steps.

The dataset involving the Charpy energy and ITT experiments required additional treatment as it required combining multiple columns (Charpy energy measured at various temperatures/ITT measured at various energy levels) into just two columns (temperature and Charpy energy). No individual column in the original dataset contained enough information for proper training of corresponding surrogate models. Therefore, initially, we chose not to include Charpy energy in this study. However, the MOEAs could not find solutions to match steel grades requiring high performance at very cold temperatures because of the

lack of Charpy energy in our initial problem formulations. The Charpy dataset we created enabled us to train reasonably well-performing models, which solved the issue.

Creating the MOP required us to find the intersection of the domains of all the objectives. We discovered in our initial calculations that the intersection was a null set. Thus, the models trained on the individual datasets would constantly be extrapolating if used together. This issue was mildly resolved by removing some extraneous columns that were almost entirely empty. This demanded further additions to the data preprocessing steps. However, the intersection was still very restricting. Expanding on the previous idea, we posited that we could remove even more columns to increase the intersection area. Therefore, it was essential to identify which decision variables had little or no impact on the objective values. We discuss the tests conducted for this task at the beginning of Section 4.

The choice and verification of models are crucial in data-driven optimization and help build trust in the solutions generated by the methods. Ideally, such models should be verified by confirming their predictions from experiments (creating and testing alloys, in our case). However, those resources were not available to us during this study. Therefore, we conducted rigorous testing of many surrogate modelling techniques to determine the best choice for each objective. We also confirmed that the predictions of the models matched current metallurgical literature. For modelling Charpy energy specifically, we tried some novel approaches of combining metallurgical knowledge (such as approximate functions which predict Charpy energy based on temperature) and surrogate modelling techniques. However, because the dataset contained many different kinds of steels, this approach did not work as well as simply using the surrogate model with the Charpy dataset.

During interactive optimization, we encountered the issue of supporting multiple DMs. We could not find implementations of interactive optimization methods that could solve our MOP while incorporating the preferences of multiple DMs. Therefore, we created a new method using DESDEO, based on the IOPIS algorithm. We elaborated on MultiDM/IOPIS in Subsection 6.2, which uses reference points as the mode of receiving preferences. However, we had created multiple variants of IOPIS which use different forms of preferences. The DMs in question felt most comfortable giving preferences in the form of reference points, so we only discuss MultiDM/IOPIS in this paper. The creation of these new methods was greatly quickened by the modularity of the DESDEO framework, which enabled us to reuse the components already present in the framework instead of implementing everything from scratch.

The ease of use of the DESDEO framework also allowed the analyst to quickly switch between and demonstrate different MOP formulations to the DMs. This, along with being actively involved in the interactive optimization process, inspired the DMs to suggest the addition of the carbon equivalent value as one of the objectives. Thus, MultiDM/IOPIS and DESDEO led to the formulation of a better MOP.

To sum up, we faced many interesting challenges while solving (or even for-

mulating) our MOP. We made additions to the DESDEO framework to resolve the challenges successfully. We implemented the novel MultiDM/IOPIS method that helped us arrive at a better problem formulation. It also provided better results than RVEA and NSGA-III while being computationally much more efficient, requiring fewer generations with a smaller population size.

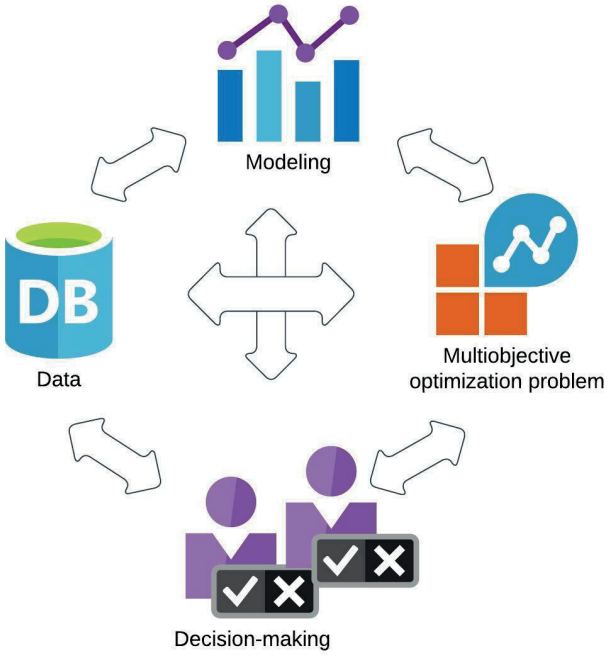


Figure 6: A realistic depiction of the seamless chain.

As we have established, the linear schematic of a seamless chain presented in Figure 1 does not capture the complexity of solving real-life data-driven problems entirely. We present a new depiction that reflects our experience of solving the MOPs presented in this study in Figure 6. The decisions made to preprocess the data, model the objectives, formulate the MOP, and supporting one or multiple DMs are interconnected and interdependent. The data limits which objectives can be considered for the MOP. The modeling and MOP formulation steps can reveal further shortcomings and issues with the data, forcing the analyst to change the preprocessing strategy. The expertise of the DMs is crucial not just in decision making, but also in understanding the data, validating the surrogate models, and formulating meaningful MOPs. An analyst's role is to channel the expertise of DMs to successfully tackle the issues that arise during all steps of the seamless chain process.

To resolve such issues, an analyst needs to use a variety of tools, many of which we have described in this study. While other real-life data-driven problems

may not have the same challenges we faced, we document the entire process as a general guideline of the seamless chain method to solve MOPs. We make all methods, procedures, implementations, data, and visualizations developed or created during this study openly available via the DESDEO framework [6].

8 Conclusions

We considered many challenges in formulating and solving real-life data-driven multiobjective optimization problems. The challenges covered all steps of a seamless chain from data to decision-making that we introduced. The concrete steps related to microalloyed steels demonstrated the reasoning. We processed the raw dataset significantly to enable using it. We identified the essential decision variables, and trained and tested the best surrogate models to emulate the objectives derived from the data. We validated those models by comparing their response to changing decision variables to known metallurgical literature. With the surrogates as objectives, we formulated multiobjective optimization problems. We introduced interactive optimization to two domain experts who acted as DMs, which inspired them to create better problem formulations. We developed a novel interactive evolutionary method called MultiDM/IOPIS to simultaneously support the two DMs (who had different preferences) in interactive optimization. The solutions obtained were very satisfactory to the DMs and the solution process had a low computational cost.

We utilized many popular open-source tools in the process of formulating and solving our problems. The DESDEO framework was instrumental as it enabled us to experiment quickly with different versions of the problems. It also allowed us to implement the MultiDM/IOPIS method quickly by utilizing its modular implementations of scalarization functions (implemented via the GLIDE-II framework [57]) and MOEAs. We discussed the steps taken to solve the MOPs in great detail and established a methodology and guidelines for solving real-life data-driven MOPs. We provide all tools, data, and methods created or used openly via the DESDEO framework, enabling their usage by others.

MultiDM/IOPIS enables supporting multiple DMs simultaneously to solve multiobjective optimization problems. It worked very well in our study and found solutions that non-interactive evolutionary algorithms RVEA and NSGA-III could not find. While our study had only two DMs, MultiDM/IOPIS can support any number of DMs (by having as many scalarization functions in its formulation). The properties of such a formulations and the application of MultiDM/IOPIS with more than two DMs need further studies. Another interesting area of study is the application of MultiDM/IOPIS in online data-driven MOPs, i.e., data-driven MOPs with the option to conduct further function evaluations to increase the accuracy of the surrogates. Moreover, solving such problems requires tackling challenges specific to online data-driven problems, and a guideline to do so will be of great use.

Acknowledgements

This research was partly funded by the Academy of Finland (grant 322221). The research is related to the thematic research area Decision Analytics utilizing Causal Models and Multiobjective Optimization (DEMO), jyu.fi/demo, at the University of Jyvaskyla.

References

[1] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, "A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms," *Soft Computing*, vol. 23, no. 9, pp. 3137–3166, 2019.

[2] Y. Jin, H. Wang, and C. Sun, *Data-Driven Evolutionary Optimization*. Springer, 2021.

[3] K. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[4] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds., *Multiobjective Optimization: Interative and Evolutionary Approaches*. Springer, 2008.

[5] B. S. Saini, J. Hakanen, and K. Miettinen, "A new paradigm in interactive evolutionary multiobjective optimization," in *Parallel Problem Solving from Nature, PPSN XVI, 16th International Conference, Part II*, T. Back, M. Preuss, A. Deutz, H. Wang, M. Doerr C. and. Emmerich, and H. Trautmann, Eds., Springer, 2020, pp. 243–256.

[6] G. Misitano, B. S. Saini, B. Afsar, B. Shavazipour, and K. Miettinen, "Desdeo: The modular and open source framework for interactive multiobjective optimization," *IEEE Access*, vol. 9, pp. 148 277–148 295, 2021.

[7] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, "A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 129–142, 2016.

[8] M. Tabatabaei, J. Hakanen, M. Hartikainen, K. Miettinen, and K. Sindhya, "A survey on handling computationally expensive multiobjective optimization problems using surrogates: Non-nature inspired methods," *Structural and Multidisciplinary Optimization*, vol. 52, no. 1, pp. 1–25, 2015.

[9] H. Wang, Y. Jin, C. Sun, and J. Doherty, "Offline data-driven evolutionary optimization using selective surrogate ensembles," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 203–216, 2018.

[10] K. Miettinen, F. Ruiz, and A. P. Wierzbicki, "Introduction to multiobjective optimization: Interactive approaches," in *Multiobjective Optimization: Interative and Evolutionary Approaches*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds., Springer, 2008, pp. 27–57.

[11] K. Deb and D. Saxena, "Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems," in *Proceedings of the World Congress on Computational Intelligence (WCCI-2006)*, 2006, pp. 3352–3360.

[12] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008, pp. 2419–2426.

[13] K. Miettinen and M. M. Mäkelä, "On scalarizing functions in multiobjective optimization," *OR Spectrum*, vol. 24, no. 2, pp. 193–213, 2002.

[14] F. Ruiz, M. Luque, and J. M. Cabello, "A classification of the weighting schemes in reference point procedures for multiobjective programming," *Journal of the Operational Research Society*, vol. 60, no. 4, pp. 544–553, 2009.

[15] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[16] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[17] N. J. Kim, "The physical metallurgy of HSLA linepipe steels—a review," *JOM*, vol. 35, no. 4, pp. 21–27, 1983.

[18] J. F. Lancaster, *Metallurgy of Welding*, 6th ed. Elsevier, 1999.

[19] J. Reback *et al.*, *Pandas-dev/pandas: Pandas*, version latest, 2020. DOI: 10.5281/zenodo.3509134.

[20] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)- A review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[22] I. Steinwart and A. Christmann, *Support vector machines*. Springer, 2008.

[23] G. Matheron, "Principles of geostatistics," *Economic Geology*, vol. 58, no. 8, pp. 1246–1266, 1963.

[24] M. Emmerich, "Single-and multi-objective evolutionary design optimization assisted by Gaussian random field metamodels," *PhD dissertation, University of Dortmund*, 2005.

[25] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[26] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232,

[27] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[29] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[30] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation," *Journal of Computational and Graphical Statistics*, vol. 24, no. 1, pp. 44–65, 2015.

[31] R. Abbaschian and R. E. Reed-Hill, *Physical Metallurgy Principles-SI Version*. Cengage Learning, 2009.

[32] H. K. Bhadeshia and D. Hansraj, *Bainite in Steels: Theory and Practice*. CRC Press, 2019.

[33] H. Bhadeshia and R. Honeycombe, *Steels: Microstructure and Properties*. Butterworth-Heinemann, 2017.

[34] T. Gladman, *The Physical Metallurgy of Microalloyed Steels*. Maney Pub, 1997.

[35] C. Bae, C. Lee, and W. Nam, "Effect of carbon content on mechanical properties of fully pearlitic steels," *Materials Science and Technology*, vol. 18, no. 11, pp. 1317–1321, 2002.

[36] M.-Y. Chen, D. Linkens, and A. Bannister, "Numerical analysis of factors influencing charpy impact properties of TMCR structural steels using fuzzy modelling," *Materials Science and Technology*, vol. 20, no. 5, pp. 627–633, 2004.

[37] Y. Li, D. N. Crowther, M. J. W. Green, P. S. Mitchell, and T. N. Baker, "The effect of vanadium and niobium on the properties and microstructure of the intercritically reheated coarse grained heat affected zone in low carbon microalloyed steels," *ISIJ International*, vol. 41, no. 1, pp. 46–55, 2001.

[38] E. Rozhkova, M. Garber, and I. Tsypin, "Effect of manganese on the transformation of austenite in white chromium cast irons," *Metal Science and Heat Treatment*, vol. 23, no. 1, pp. 59–63, 1981.

[39] R. Mesquita and H.-J. Kestenbach, "On the effect of silicon on toughness in recent high quality hot work steels," *Materials Science and Engineering: A*, vol. 528, no. 13-14, pp. 4856–4859, 2011.

[40] H. Erhart and H.-J. Grabke, "Equilibrium segregation of phosphorus at grain boundaries of fe–p, fe–c–p, fe–cr–p, and fe–cr–c–p alloys," *Metal Science*, vol. 15, no. 9, pp. 401–408, 1981.

[41] S. Vervynckt, K. Verbeken, B. Lopez, and J. Jonas, "Modern HSLA steels and role of non-recrystallisation temperature," *International Materials Reviews*, vol. 57, no. 4, pp. 187–207, 2012.

[42] S.-H. Kim, C.-Y. Kang, and K.-S. Bang, "Weld metal impact toughness of electron beam welded 9% ni steel," *Journal of Materials Science*, vol. 36, no. 5, pp. 1197–1200, 2001.

[43] L.-Å. Norström and O. Vingsbo, "Influence of nickel on toughness and ductile-brittle transition in low-carbon martensite steels," *Metal Science*, vol. 13, no. 12, pp. 677–684, 1979.

[44] A. DeArdo, "Niobium in modern steels," *International Materials Reviews*, vol. 48, no. 6, pp. 371–402, 2003.

[45] W. Morrison, "Microalloy steels–the beginning," *Materials Science and Technology*, vol. 25, no. 9, pp. 1066–1073, 2009.

[46] W. Yan, Y. Shan, and K. Yang, "Effect of tin inclusions on the impact toughness of low-carbon microalloyed steels," *Metallurgical and Materials Transactions A*, vol. 37, no. 7, pp. 2147–2158, 2006.

[47] F. Wilson and T. Gladman, "Aluminium nitride in steel," *International Materials Reviews*, vol. 33, no. 1, pp. 221–286, 1988.

[48] P. J. Uggowitzer, R. Magdowski, and M. O. Speidel, "Nickel free high nitrogen austenitic steels," *ISIJ International*, vol. 36, no. 7, pp. 901–908, 1996.

[49] T. Baker, "Processes, microstructure and properties of vanadium microalloyed steels," *Materials Science and Technology*, vol. 25, no. 9, pp. 1083–1107, 2009.

[50] A. Ghosh, S. Sahoo, M. Ghosh, R. Ghosh, and D. Chakrabarti, "Effect of microstructural parameters, microtexture and matrix strain on the Charpy impact properties of low carbon HSLA steel containing mns inclusions," *Materials Science and Engineering: A*, vol. 613, pp. 37–47, 2014.

[51] D. Isheim, M. S. Gagliano, M. E. Fine, and D. N. Seidman, "Interfacial segregation at cu-rich precipitates in a high-strength low-carbon steel studied on a sub-nanometer scale," *Acta Materialia*, vol. 54, no. 3, pp. 841–849, 2006.

[52] D. Ye, J. Li, W. Jiang, J. Su, and K. Zhao, "Effect of cu addition on microstructure and mechanical properties of 15% cr super martensitic stainless steel," *Materials & Design*, vol. 41, pp. 16–22, 2012.

[53] Z. Adabavazeh, W. Hwang, and Y. Su, "Effect of adding cerium on microstructure and morphology of ce-based inclusions formed in low-carbon steel," *Scientific Reports*, vol. 7, no. 1, pp. 1–10, 2017.

[54] A. Guo *et al.*, "Effect of zirconium addition on the impact toughness of the heat affected zone in a high strength low alloy pipeline steel," *Materials Characterization*, vol. 59, no. 2, pp. 134–139, 2008.

[55] H. Nakayama and Y. Sawaragi, "Satisficing trade-off method for multi-objective programming," in *Interactive Decision Analysis*, M. Grauer and A. P. Wierzbicki, Eds., Berlin, Heidelberg: Springer, 1984, pp. 113–122, ISBN: 978-3-662-00184-4.

[56] K. Li, R. Wang, T. Zhang, and H. Ishibuchi, "Evolutionary many-objective optimization: A comparative study of the state-of-the-art," *IEEE Access*, vol. 6, pp. 26 194–26 214, 2018.

[57] F. Ruiz, M. Luque, and K. Miettinen, "Improving the computational efficiency in a global formulation (GLIDE) for interactive multiobjective optimization," *Annals of Operations Research*, vol. 197, no. 1, pp. 47–70, 2012.

A Effect of concentrations of alloying elements on metallurgical properties of microalloyed steels

Carbon: Being the primary interstitial solute in steel, C atoms strongly interact with both edge- and screw-dislocations and provide significant solid solution strengthening. Besides, C increases hardenability and promotes the formation of harder phase constituents like pearlite, bainite and martensite which increases the strength. In addition, iron carbides and alloy carbides contribute to precipitation strengthening. An increase in strength by the increase in C content in general hampers ductility, toughness, formability and even weldability. The coarse and brittle carbide particles can act as the crack initiators (or void nucleation sites) and thus, affect impact toughness. On the other hand, fine carbide precipitates pin down the grain boundaries and restrict grain growth. The beneficial effect of microalloy carbide precipitates (say, NbC) on the retardation of austenite recrystallization and the consequent ferrite grain refinement in thermomechanical processed microalloyed steels is well known. Grain refinement is beneficial for impact toughness. As a result, C shows a positive effect on strength, a negative effect on ductility and a mixed response on impact toughness.

Silicon: As a substitutional solute, Si provides solid solution strengthening and shows a positive effect on strength (particularly UTS) and a negative effect on ductility. Si is preferred in steels containing low levels of C and other strengthening elements like Mn. That could result in a mixed response of Si on YS. Si decreases the cohesive strength of the atomic planes helping cleavage crack propagation and renders steel brittle, affecting impact toughness. In contrast, Si restricts the formation of iron-carbides (detrimental to roughness) and contributes to carbide free bainitic microstructures. As a result, Si does not show a clear trend on Charpy energy.

Manganese: Being present at a considerable amount (0.5 to 2.0 wt.%), Mn is a strong solid solution strengthener in steels, which can naturally affect

ductility. Mn is maintained at a higher side typically in low-C steels, having high ductility and impact toughness. Besides, Mn stabilizes retained austenite which contributes to the TRIP (transformation induced plasticity) effect. Mn also suppresses pearlite formation and coarsen the interlamellar spacing in pearlite. Although MnS inclusion is detrimental to toughness, it is not as harmful as iron-sulfide. These combined factors resulted in a mixed response of Mn on ductility and a positive influence on impact toughness.

Phosphorous: Segregation of P at the grain boundaries reduces the boundaries' cohesive strength, and the associated embrittlement seriously affects ductility and impact toughness. Although P is a solid solution strengthener, its content is restricted in commercial grades of steel. It is only allowed when the levels of C and the other alloying elements are very low (like P strengthened interstitial free steel). That is possibly the reason behind P's negative effect on strength, as detected here.

Sulfur: S has a strong negative effect on steel properties, particularly on ductility and toughness. Coarse and elongated MnS inclusions initiate large voids and fissures (promoting ductile fracture), cause anisotropy in properties, and even act as the cleavage crack initiators. Segregation of S at grain boundaries and interdendritic regions and the formation of iron-sulfides can be even more detrimental for properties. Being softer than the steel matrix, MnS can reduce the strength when present at a high fraction. However, occasionally fine MnS particles can offer grain refinement by (i) pinning the grain boundaries, and (ii) VN precipitation on MnS can provide nucleation sites for intergranular ferrite within austenite grains. Hence, S showed a mixed response on the YS and a negative influence on other properties.

Molybdenum: Mo provides solid solution strengthening and precipitation strengthening by forming various precipitates such as Mo2C, (Ti, Mo)C and (V, Mo)C. Mo also increases hardenability and promotes bainite transformation in steels. It refines the interlamellar spacing of pearlite. These aspects can have a beneficial effect on strength but a detrimental impact on ductility. Besides, Mo significantly retards temper embrittlement. In modern structural and linepipe grades of steel (and their weld joints), acicular ferrite microstructure is preferred to achieve high strength and high toughness. Mo can promote such microstructure and improve the impact toughness.

Niobium: As a microalloying element in thermomechanically processed high-strength low-alloy (HSLA) steels, NbC, NbN, and Nb(C, N) precipitate contribute significant ferrite-grain refinement as well as precipitation strengthening. Nb is also a scavenger of C and N from solution, which can improve ductility. Besides, being a substitutional solute, Nb offers solid solution strengthening, increases hardenability and promotes the formation of bainite and acicular ferrite microstructures. Thus, Nb shows a clear trend, i.e., a positive effect on all the investigated properties.

Vanadium: As a microalloying element, the primary contribution of V is precipitation strengthening in steels by the formation of numerous fine VC and V(C, N) precipitates during austenite to ferritic transformation. However, strong precipitation strengthening from V negatively affects ductility and

toughness. Although VN particles in austenite can act as nucleation sites for intragranular ferrite grains, V, in general, is not a strong grain refiner. Thus, V demonstrates a positive response on strength but a negative response on ductility and impact toughness.

Titanium: As a microalloying element, the primary contribution of Ti is to restrict austenite grain growth during soaking, welding, and even conventional hot-rolling by the formation of stable TiN and Ti(C, N) precipitates. Such a grain size control can be beneficial for impact toughness. Similar to Nb, dissolved Ti improves hardenability and provides solid solution strengthening. Fine-scale precipitation of TiC (at relatively lower temperatures) can also offer precipitation strengthening. Recently there has been an emphasis on nanometer-sized (Ti, Mo)C precipitation strengthened ferritic steels for automotive applications. Precipitation strengthening can hamper ductility and toughness. Ti and N contents should be controlled carefully in steels as course and brittle TiN particles are the potent sites for cleavage crack initiation that can seriously hamper impact toughness and ductility. Thus, Ti shows a positive effect on strength, a negative effect on ductility, and a mixed response on impact toughness.

Silicon: As a substitutional solute, Si provides solid solution strengthening and shows a positive effect on strength (particularly UTS) and a negative effect on ductility. Si is preferred in steels containing low levels of C and other strengthening elements like Mn. That could result in a mixed response of Si on YS. Si decreases the cohesive strength of the atomic planes helping cleavage crack propagation and renders steel brittle, affecting impact toughness. In contrast, Si restricts the formation of iron-carbides (detrimental to roughness) and contributes to carbide free bainitic microstructures. As a result, Si does not show a clear trend on Charpy energy.

Manganese: Being present at a considerable amount (0.5 to 2.0 wt.%), Mn is a strong solid solution strengthener in steels, which can naturally affect ductility. Mn is maintained at a higher side typically in low-C steels, having high ductility and impact toughness. Besides, Mn stabilizes retained austenite which contributes to the TRIP (transformation induced plasticity) effect. Mn also suppresses pearlite formation and coarsen the interlamellar spacing in pearlite. Although MnS inclusion is detrimental to toughness, it is not as harmful as iron-sulfide. These combined factors resulted in a mixed response of Mn on ductility and a positive influence on impact toughness.

Phosphorous: Segregation of P at the grain boundaries reduces the boundaries' cohesive strength, and the associated embrittlement seriously affects ductility and impact toughness. Although P is a solid solution strengthener, its content is restricted in commercial grades of steel. It is only allowed when the levels of C and the other alloying elements are very low (like P strengthened interstitial free steel). That is possibly the reason behind P's negative effect on strength, as detected here.

Sulfur: S has a strong negative effect on steel properties, particularly on ductility and toughness. Coarse and elongated MnS inclusions initiate large voids and fissures (promoting ductile fracture), cause anisotropy in properties,

and even act as the cleavage crack initiators. Segregation of S at grain boundaries and interdendritic regions and the formation of iron-sulfides can be even more detrimental for properties. Being softer than the steel matrix, MnS can reduce the strength when present at a high fraction. However, occasionally fine MnS particles can offer grain refinement by (i) pinning the grain boundaries, and (ii) VN precipitation on MnS can provide nucleation sites for intergranular ferrite within austenite grains. Hence, S showed a mixed response on the YS and a negative influence on other properties.

Nickel: Being a solid solution strengthener Ni is expected to improve the strength of steel. Although strengthening can negatively affect ductility and toughness, Ni is particularly beneficial in improving the low-temperature impact toughness, preventing ductile-to-brittle transition in ferritic steels. Being an austenite stabilizer, Ni can also enhance impact toughness through the TRIP effect of retained austenite. Ni may not be as effective in improving room-temperature toughness as low-temperature toughness. Besides being an expensive alloying element, Ni is typically added in special grades of heavily alloyed high-strength steels, which inherently have a low ductility and impact toughness (to restore these properties). Therefore, Ni showed a negative response to ductility and a mixed response to impact toughness.

Aluminium: Al is used for deoxidation and grain refinement in steel by the formation of Al2O3 and AlN, respectively. Al has a weak effect on hardenability, and it stabilizes the soft and ductile ferrite phase. The ability of Al to scavenge N from solution reduces the strengthening effect of N, which can improve ductility, toughness, and formability. The steel also becomes resistant to strain-ageing and the associated yield point phenomenon as desired in formable automotive grades of steel. AlN particles restrict grain growth and help achieve a fine grain size, which is beneficial for impact toughness. In bainitic steels, Al also retards carbide precipitation (iron-carbides are detrimental to toughness) which stabilizes retained austenite and contributes TRIP effect. Hence, Al showed a negative effect on strength but a positive response on ductility and impact toughness.

Nitrogen: Although N is a strong solid solution strengthener, it significantly hampers ductility, toughness, and formability when in solution. Therefore, N in solution is minimized by the scavenging action of strong nitride forming elements such as Ti, Al, Nb and V. The grain refinement of microalloy nitride and carbonitride precipitates, along with the N free matrix, can improve ductility and toughness. Therefore, N does not show any trend on strength, mixed response on ductility, and positive influence on impact toughness.

Boron: B is used at a controlled quantity to increase the hardenability of special grades of steels having bainite or tempered martensite microstructures. In this study, B does not show a clear trend with any of the properties possibly due to the following reasons. (i) B in solution at only tens of ppm can be effective for enhancing hardenability. However, higher addition of B can be detrimental due to the formation of hard and brittle particles like BN, metal-borides and boro-sulfides. (ii) To prevent BN formation, B is shielded by the addition of stronger nitride formers such as Ti, Al and Zr, and those elements also influence

the steel's properties. (iii) Finally, the data available on B containing steels is limited.

Chromium: Cr enhances hardenability, refines interlamellar spacing of pearlite, and provides solid solution strengthening and precipitation strengthening. Hence, Cr has a positive effect on strength. Coarse Cr23C6 precipitates, however, can impose a negative effect on ductility and toughness.

Cerium: Being a rare earth metal, Ce is occasionally added in steels to control the shape and size of sulfide and oxide inclusions. Ce in solution and its grain boundary segregation may impart some strength apart from inclusion refinement, which can benefit toughness. However, Ce is usually added in high-strength steels with low ductility and toughness. Ce addition needs to be carefully controlled and a high Ce level (> 0.03 wt.%) can be detrimental. Data availability on Ce containing steels is also limited. Hence, Ce shows a positive effect on strength but no clear trend on the other properties.

Copper: Cu has low solubility in ferritic steels. It precipitates out as metastable BCC which is coherent with the matrix and then transforms to incoherent FCC, increasing strength. The negative effect of Cu on steels is difficult to explain apart from the fact that being softer than the matrix, Cu precipitates may soften the steel at the onset of plastic deformation when present at a high fraction. High addition of Cu can result in its segregation at grain boundaries and surface regions and can hamper the ductility, particularly at high temperatures, causing hot cracking. FCC-Cu precipitates can restrict the ductile-to-brittle transition at the low test temperature. However, the present study could not detect its beneficial effect on impact toughness.

Zirconium: Being a rare and expensive alloying element, Zr is occasionally used in steels. Its affinity for O, S, and N is the primary reason behind Zr addition: controlling the non-metallic inclusions and scavenging N from solution (say, to protect B). Zr's oxides, sulfides, and nitrides can prevent grain growth at high temperatures. However, the beneficial effects of Zr could not be identified here possibly due to limited data availability of Zr steels.

Some of the elements mentioned in Table 2 are added to steels to improve certain properties beyond the present study's scope. For example, Cr, N and Cu are beneficial for corrosion resistance.