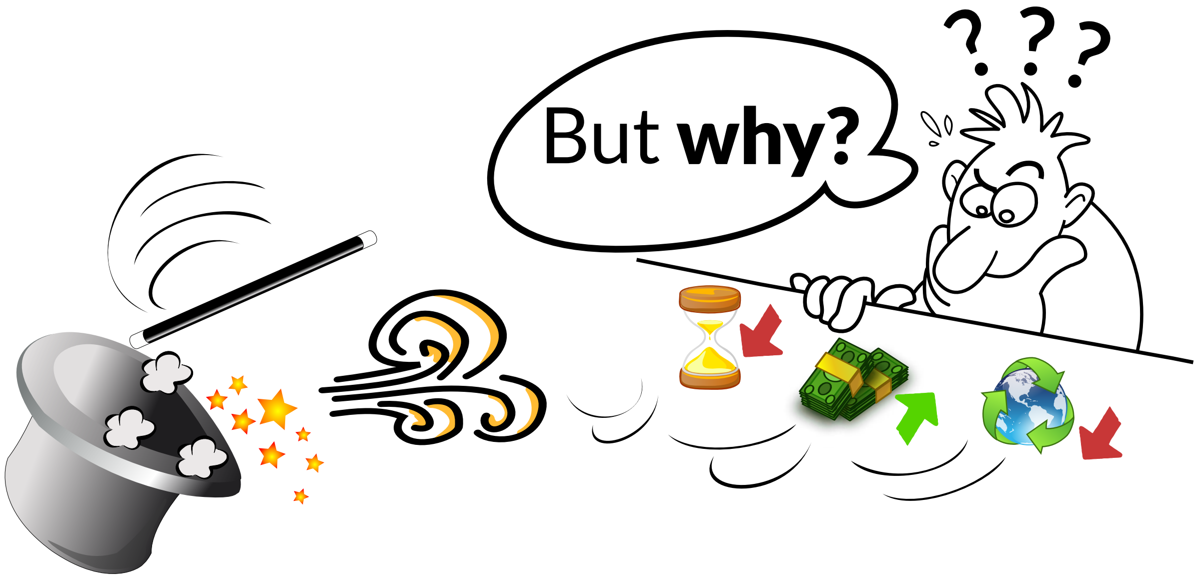
JYU DISSERTATIONS 794

Giovanni Misitano

Enhancing the decision-support capabilities of interactive multiobjective optimization with explainability



UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY

JYU DISSERTATIONS 794

Giovanni Misitano

Enhancing the decision-support capabilities of interactive multiobjective optimization with explainability

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi Ruusupuisto-rakennuksen Helena-salissa (RUU D104)
toukokuun 31. päivänä 2024 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Ruusupuisto, auditorium Helena (RUU D104), on May 31, 2024, at 12 o'clock noon.

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2024

Editors
Marja-Leena Rantalainen
Faculty of Information Technology, University of Jyväskylä
Timo Hautala
Open Science Centre, University of Jyväskylä

ISBN 978-952-86-0183-8 (PDF)
URN:ISBN:978-952-86-0183-8
ISSN 2489-9003

Permanent link to the online version of this publication: http://urn.fi/URN:ISBN:978-952-86-0183-8

Jyväskylä University Printing House, Jyväskylä 2024

ABSTRACT

Misitano, Giovanni
Enhancing the decision-support capabilities of interactive multiobjective optimization with explainability
Jyväskylä: University of Jyväskylä, 2024, 100 p. (+included articles)

Real-life decisions often involve balancing conflicting criteria, which are modeled as objective functions in multiobjective optimization problems. These problems have multiple optimal solutions that cannot be ordered from worst to best without input from a domain expert—a decision maker. To help a decision maker choose their most preferred solution, many multiobjective optimization methods use preferences expressed by the decision maker to identify promising solution candidates. Interactive multiobjective optimization methods are particularly useful as they involve the decision maker in the solution process by allowing them to iteratively provide preferences and review solutions until a satisfactory one is found.

However, when utilizing interactive methods, decision makers often lack in the support they get for providing preferences and understanding the optimal solutions. This thesis explores how explainability, a concept from the field of artificial intelligence, can tackle these issues and enhance the support provided by interactive methods. Explainability can help make complex systems more intelligible, and we demonstrate how the concept can improve interactive methods by making them easier for decision makers to apply and understand.

We introduce three new interactive methods that utilize explainability. The first, R-XIMO, helps decision makers give preferences and understand their impact on the computed solution candidates. The second, INFRINGER, uses explainable rule-based models to effectively capture and model a decision maker's preferences. The third, XLEMOO, uses rule-based models to explain solution candidate characteristics that align with the decision maker's preferences. Lastly, the open source DESDEO software framework supports these methods by enabling their implementation and development, and it is therefore discussed as well.

This thesis lays the groundwork for the new field of explainable interactive multiobjective optimization, offering methods that are openly available for further research and practical use. By enhancing interactive methods with explainability, this novel line of research can lead to more transparent and justifiable decision-support tools in diverse real-life decision-making problems involving decision makers, preferences, and multiple conflicting objective functions.

Keywords: explainable decision-making, multi-criteria optimization, interactive multiobjective optimization, decision-support, explainable artificial intelligence, evolutionary multiobjective optimization, open source software

TIIVISTELMÄ (ABSTRACT IN FINNISH)

Misitano, Giovanni
Interaktiivisten monitavoiteoptimointimenetelmien tarjoaman päätöksenteon tuen tehostaminen selitettävyyden avulla
Jyväskylä: University of Jyväskylä, 2024, 100 s. (+artikkelit)

Monet päätöksenteko-ongelmat edellyttää usein ristiriitaisten kriteerien tasapainottelua. Nämä ongelmat voidaan mallintaa monitavoiteoptimointiongelmina, joissa tavoitefunktiot kuvaavat kriteerejä. Näissä ongelmassa on useita optimaalisia ratkaisuehdokkaita, joita ei voida järjestää paremmuusjärjestykseen ilman asiantuntijan, eli päätöksentekijän, panosta. Erilaisia monitavoiteoptimointimenetelmiä, jotka hyödyntävät päätöksentekijän mieltymyksiä, voidaan käyttää päätöksenteontukena lupaavien ratkaisuehdokkaiden tunnistamisessa. Interaktiiviset monitavoiteoptimointimenetelmät ovat erityisen hyödyllisiä, sillä niissä päätöksentekijä on osa ratkaisuprosessia, mikä sallii heidän ilmaista iteratiivisesti mieltymyksiään ja tarkastella ratkaisuja, kunnes tyydyttävä ratkaisu on löytynyt.

Interaktiiviset menetelmät tukevat kuitenkin harvoin päätöksentekijää mieltymysten ilmaisemisessa ja optimaalisten ratkaisujen ymmärtämisessä. Tässä väitöskirjassa tutkitaan miten selitettävyys, joka on konsepti tekoälyn alalta, voisi vastata tähän tuen puutteeseen ja parantaa interaktiivisten menetelmien tarjoamaa tukea. Selitettävyys voi tehdä monimutkaisista järjestelmistä ymmärrettävämpiä. Väitöskirjassa tutkitaan miten selitettävyys voi parantaa interaktiivisia menetelmiä tekemällä niistä helpommin sovellettavia ja ymmärrettäviä päätöksentekijöille.

Väitöskirjassa esitellään kolme uutta interaktiivista menetelmää, jotka hyödyntävät selitettävyyttä. R-XIMO-menetelmä tukee päätöksentekijöitä mieltymysten ilmaisemisessa ja ymmärtämään miten ne ovat vaikuttaneet laskettuihin ratkaisuihin. INFRINGER-menetelmä käyttää selitettäviä sääntöpohjaisia malleja päätöksentekijän mieltymysten selvittämiseen ja mallintamiseen. XLEMOO-menetelmä käyttää sääntöpohjaisia malleja selittämään niiden ratkaisuehdokkaiden ominaisuuksia, jotka vastaavat päätöksentekijän mieltymyksiä. Myös väitöskirjan aikana kehitettyä avoimen lähdekoodin DESDEO-ohjelmistokehystä käsitellään, sillä se on mahdollistanut esiteltyjen menetelmien kehittämisen.

Tämä väitöskirja luo perustan uudelle selitettävän interaktiivisen monitavoiteoptimoinnin tutkimusalalle. Esitellyt menetelmät ovat vapaasti saatavilla jatkotutkimusta ja käytännön sovelluksia varten. Tehostamalla interaktiivisia menetelmiä selitettävyydellä, tämä uusi tutkimusala voi johtaa läpinäkyvämpiin ja perustellumpiin päätöksenteon tukityökaluihin monissa erilaisissa päätöksentekoongelmissa, joissa tuetaan päätöksentekijää ristiriitaisten tavoitefunktioiden tasapainottelussa heidän mieltymyksiä hyödyntäen.

Avainsanat: selitettävä päätöksenteko, monikriteerioptimointi, interaktiivinen monitavoiteoptimointi, päätöksenteontuki, selitettävä tekoäly, evolutionäärinen monitavoiteoptimointi, avoimen lähdekoodin ohjelmisot

Author	Giovanni Misitano Faculty of Information Technology University of Jyväskylä Finland
Supervisors	Prof. Kaisa Miettinen Faculty of Information Technology University of Jyväskylä Finland
	Dr. Bekir Afsar Faculty of Information Technology University of Jyväskylä Finland
Reviewers	Prof. Juergen Branke Warwick Business School The University of Warwick United Kingdom
	Prof. Jian-Bo Yang Decision and Cognitive Sciences Research Centre The University of Manchester United Kingdom
Opponent	Prof. Serpil Sayın College of Administrative Sciences and Economics Koç University Turkey

FOREWORD

This thesis is, first and foremost, a collection of ideas and their proof-of-concepts. At the time of writing this thesis, these ideas are nothing but freshly planted seeds that are yet to sprout. Ideally, these seeds will grow into something bigger—tools for supporting humans tackle decision-making problems with multiple conflicting criteria in an understandable and justifiable way, without having to blindly trust computers and algorithms. Most importantly, I hope my research can eventually help us keep the human factor alive and present in decision-making despite the rapid advancements we have seen in artificial intelligence in the last few years. I also wish that my work can encourage humans to consider many of the other factors in decision-making, in addition to just maximizing profits—something many policy-makers these days seem to solely focus on. I do not believe—or rather, I do not want to believe—that this narrow-mindedness is out of pure greed. I am naïve enough to be convinced that most people simply do not know better. That is why I want to research and develop tools that would help us consider multiple perspectives and the bigger picture in important decision-making tasks. I am strongly of the opinion that if people have the means to do better, they will.

Naturally, the journey that has led to the conception of this thesis was not traveled alone. First, and foremost, I want to express my deepest gratitude to my thesis supervisor, Kaisa Miettinen, for always taking the time to share her—seemingly endless—wisdom in many academic, and not-so-academic, matters; and for investing her ever more elusive time to provide me with the best constructive feedback I have ever got—and I fear I will ever get. I must also thank Bekir Afsar for his time and patience as well in supervising my thesis work. Somehow, he has often been able to find new perspectives and point out details that even Kaisa might have missed, which only speaks in favor of his great expertise as well. Since this paragraph is clearly dedicated to thanking supervisors, it would be a crime to not mention Jussi Hakanen. He had the chance to mentor me only during the beginning of my PhD pilgrimage, but nonetheless, Jussi has played a critical role in introducing me to the topic of explainability, and its potential application to multiobjective optimization. When it comes to supervision, I had the privilege to be in very good hands.

The Multiobjective Optimization Group has also provided a highly welcoming environment. I felt like I was part of the team since day one when I was but a research assistant. Our research group has a very diverse array of people from different parts of the world, which is evident in the just as diverse points of view and set of opinions that often emerge in the many discussion we have had. I would love to tell some of my personal experiences related to each group member, but it would inflate the contents of this Foreword too much. Therefore, I will simply thank all of our group members, current and past, with whom I had the privilege to collaborate, discuss various matters, and generally interact with. So thank you Bhupinder Sing Saini, Juuso Pajasmaa, Babooshka Shavizapour, Giomara Lárraga, Atanu Mazumdar, Johanna Silvennoinen, Eero Lantto, Risto

Heikkinen, Adhe Kania, MaoMao Liang, Juho Roponen, Michael Emmerich, and Pouya Aghaei pour—in no particular order. There are also many collaborators and acquaintances outside of our research group that I have had the pleasure to meet, collaborate, and share ideas with. Too many to name them all, but I am nevertheless sincerely thankful for these interactions. Lastly, I want to thank the reviewers of this thesis, Jian-bo Yang and Juergen Branke, for taking the time to carefully read and evaluate my thesis.

While I am defending my thesis in the field of computer science, my academic roots reside in the field of physics. It is during my physics studies that I have met many of my current friends, and had many delightful interactions—too many to list all. Therefore, I will limit the following name dropping to my closest posse during my studies, namely, Teemu Loippo, Topi Löytäinen, Niina Mäkelä, and Kirsi Mäki. Thank you for being such an important part of my academic journey, and contributing to the lovely atmosphere of the FYS4 room, where we spent numerous evenings tackling physics problems of many sorts. In fact, my interactions with Kirsi have been so enjoyable that we ended up pursuing a non-platonic relationship, which, for the time being, has lasted quite a few years already. And I truly hope it would last for many more years to come. Apart from Kirsi, I have spent most of my time during my thesis work with our cats: Hiisi, Olio, Örkki, and the late Nadia. These fur balls (and one skin ball) have kept me awake many a night, interrupted my work, and just been generally a nuisance. In other words, they have provided the occasional, but invaluable, respite from my never-ending work. For that, I am thankful.

There is also the rest of my family, who has supported me on my academic endeavors for most of my life. My mother, Tuija, and late father, Giuseppe, have always made sure my basic needs are met, so that I may focus on my studies; and they have always encouraged me in my academic pursues. It has also been very inspiring following my brother's, Giacomo's, journey as a book author—we both seem to have a natural knack when it comes to writing. That said, here is an awkward transition to me acknowledging the financial support I have got during my thesis journey. For this, I wish to warmly thank the Väisälä Fund.

Lastly, there is Lady Luck, to whom I must express my gratitude as well. Being born in a prospering country, with all the necessary resources that have allowed me to pursue my academic interests, is solely thanks to being in Fortuna's favor. So are many other factors that have more or less aided me on my path that ultimately led to the writing of this thesis. While it is mostly up to chance when it comes to the cards we are dealt with in life, we must still play our hand cleverly. There have undoubtedly been a lot of stupid misplays from my part, but in the bigger picture, I think I have been mostly in the black. I will keep making the best plays I can, and I will remind myself daily that I am in a very privileged position being able to strive for my scholarly goals relatively worry free. There are many people less blessed than I, from whom Tyche has turned her gaze afar. It would therefore be foolish—even inconsiderate—of me to waste good cards and fold unnecessarily, when many are still waiting for their very first hand being dealt to them. It is up to us, the prosperous ones, to make sure that there are enough chips

for everyone in the future.

I hope the ideas presented in this thesis will ultimately bud into something substantial. But for that to happen, the eventual saplings will have to be tended to before they may grow into their full splendor. This is not a one-person job, and future collaborations will have to be established, and existing ones will have to be nurtured. I will strive to be part of as many of these cohorts as I can, and I hope to be able to enjoy at least some of the fruits these efforts will eventually bear. Alas, the sweetest of nectars will be enjoyed only by the generations to come, once I will be long gone, for the march of science is a slow thing. But such is life—it is finite. That is why we should make the most of it, and make it as enjoyable as possible for all living beings. But to pursue such undertakings, we, as a society, will have to face many difficult decision-making problems. To be able to tackle these, and to be able to make the best possible decisions, we will need the right tools for the job. This thesis is a small and humble, yet important, step toward achieving such means.

Giovanni Misitano
Jyväskylä 8.5.2024

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

I Misitano G., Afsar B., Lárraga G., Miettinen K.. Towards explainable interactive multiobjective optimization: R-XIMO. *Autonomous Agents and Multi-Agent Systems*, 36 (2), 43, 2022.

II Misitano G.. Interactively learning the preferences of a decision maker in multi-objective optimization utilizing belief-rules. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 133–140, 2020.

III Misitano G.. Exploring the explainable aspects and performance of a learnable evolutionary multiobjective optimization method. *ACM Transactions on Evolutionary Learning and Optimization*, 4 (1), Article 4, 2023.

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

1 INTRODUCTION

Decision-making can be found in many aspects and areas of our society. It is both a daily occurrence in our individual lives and an integral part of the curricula of larger corporations and governments, e.g., online shopping (Stankevich, 2017), city landscape planning (Ordóñez et al., 2019), electric vehicle adoption in the European Union (Biresselioglu et al., 2018), and medicine and healthcare (Kaplan and Frosch, 2005). Some decision-making problems are simple, consisting only of a few non-conflicting criteria. In these simpler tasks, the decision alternatives are readily enumerable and the best solution is unambiguous. However, life is often more complicated than that, and we have to face far more challenging decision-making problems, where there are many conflicting criteria and a large number of decision alternatives—sometimes an uncountable or unknown amount, or both. It is these more challenging types of decision-making problems that researchers have found to be intriguing and worth researching in the field of multiple criteria decision-making (Köksalan et al., 2011), a branch of operations research (Gass and Assad, 2005).

If we think of decision-making as a process to find the best possible solution to a problem, given certain circumstances, then it is not far-fetched to think of it as a form of optimization. In fact, some decision-making problems can be modeled so that the criteria themselves are to be optimized. In such a case, we are dealing with so-called *multiobjective optimization problems* (Nakayama, 1995; Miettinen, 1999). Instead of conflicting multiple criteria, in multiobjective optimization we deal with multiple conflicting *objective functions*. These functions are dependent on *decision variables*. Furthermore, the decision variables can be subject to *constraints*. Different objective function values can be found by altering the decision variable values subject to the constraints.

Consider the example of designing a wooden table as a multiobjective optimization problem, as illustrated in Figure 1. This problem aims to maximize the table's surface area while minimizing the thickness of the tabletop, deviation from an industry-standard height, and manufacturing cost. The objective functions depend on various decision variables, including the amount of wood, different dimensions of the table, and manufacturing quality. Constraints include the mini-

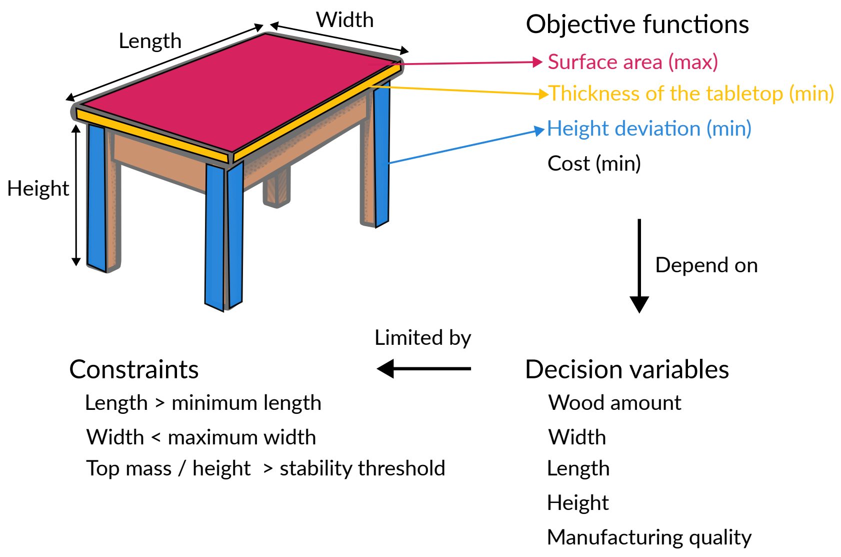


FIGURE 1 Designing a wooden table as a multiobjective optimization problem. There are four objective functions: maximize (max in the figure) the surface area of the tabletop, minimize (min in the figure) the thickness of the tabletop, minimize the table's height from an industry standard, and minimize the manufacturing cost of the table. The objective functions then depend on one or more decision variables that characterize the table and its manufacturing process, which are the amount of wood utilized in manufacturing the table, its dimensions, and the overall manufacturing quality of the table. Lastly, there are constraints on the minimum length and the maximum width of the tabletop, and a stability constraint on the ratio between the table's height and top mass.

mum length and maximum width for the tabletop and a limit on the ratio of its height and mass to assure stability. These objectives outline the decision-making aspects of the table's design, focusing on optimizing the objective functions rather than manipulating the decision variables directly. The relationship between objectives and variables is often complex, influenced by the problem's modeling.

The nature of the objective functions in multiobjective optimization problems often leads to conflicts between them. Continuing with the example in Figure 1, for instance, a larger surface area might necessitate a thicker tabletop and shorter legs for stability, increasing wood usage and cost. Conversely, a shorter table height can reduce costs and improve stability, but deviate from the standard height, an undesirable sacrifice. A feasible solution to this multiobjective optimization problem, and multiobjective optimization problems in general, involves identifying a set of decision variables that meet all constraints and result in some desirable objective function values. But when exactly can we talk about an *optimal* solution to a multiobjective optimization problem? And when can the objective function values of a solution be considered *desirable*? According to *whom*?

The very defining property of multiobjective optimization problems is worth reiterating here: the objective functions are in *conflict*, which means that no solution exists that results in all the objective functions reaching their optimal value simultaneously. This gives rise to the unique property of multiobjective optimization problems: multiple optimal solutions, known as *Pareto optimal* solutions, exist. By definition, a Pareto optimal solution is such that there are no other feasible solutions that would improve any of the objective function values without causing a degradation in at least one other objective function value. For this reason, we often limit our attention to inspecting only the Pareto optimal solutions[1] of a multiobjective optimization problem.

Due to the nature of Pareto optimal solutions, we cannot switch from one solution to another in hopes of improving the value of a particular objective function without deteriorating the value of at least one other—we have to inevitably make some *trade-offs* between the objective function values. For instance, in the example in Figure 1, if we wish to design a table with a very thick and large tabletop, requiring an increased amount of wood, we must be ready to pay more for the table—we cannot expect to find a table that is both cheap and massive. Therefore, there is a trade-off between the cost of the table and the surface area and thickness of the tabletop. Moreover, because the objective function values that characterize Pareto optimal solutions are represented by vectors consisting of numerical values, they cannot be fully ordered, e.g., from worst to best, on a mathematical basis alone. In other words, Pareto optimal solutions are incomparable.

At this point, we might wonder: if multiobjective optimization problems have multiple Pareto optimal solutions, then how can the best solution to a multiobjective optimization problem be determined if the solutions are incomparable? The judge of this is a *decision maker*. The decision maker is an expert with knowledge on the domain of the problem being solved. It is their task to explore the available solutions and make a final decision on which solution is the best or most desirable. This means that the *best* solution is subjective and depends on what the decision maker prefers. Returning once more to the example in Figure 1, one decision maker may prefer a massive and costly table, but does not care much about its height deviation from a standard; while another decision maker could instead be looking for a cheap and light table, and is very adamant that the height deviation from the standard is as small as possible.

Because multiobjective optimization problems often have an uncountable amount of Pareto optimal solutions, it is infeasible to expect a decision maker to explore all the available alternatives. Thus, many methods have been developed to support decision-making in multiobjective optimization problems, aiding the decision maker in exploring the available Pareto optimal solutions (Hwang and Masud, 1979; Sawaragi et al., 1985; Miettinen, 1999; Branke et al., 2008; Brockhoff et al., 2023). These methods aid decision makers in finding their best solution. They do so by utilizing *preferences* expressed by the decision maker.

A decision maker can express preferences in multiple ways and at different times in respect to the optimization process. In fact, multiobjective optimization methods that utilize preferences can be divided into three categories based on

[1] As we will see in Chapter 2, we might also limit our attention to the approximations of Pareto optimal solutions or so-called *non-dominated sets*.

when the preferences are utilized (Miettinen, 1999)[2]. In the first category, preferences are expressed before the optimization process. When decision makers are certain about their preferences, their feasibility, and the nature of the trade-offs between the objective functions, these methods can prove useful. However, if the decision maker is unsure about any of these aspects, providing preferences and finding the best solution to the problem may prove to be challenging. In the second category, preferences are expressed after the optimization when a set of Pareto optimal solutions has been computed. These methods are useful when the problem being solved is simple enough, e.g., has only a couple of objective functions, and a representation of all the Pareto optimal solutions can be readily communicated, for example, visualized, to the decision maker. But if the number of objective functions increases and the set of Pareto optimal solutions becomes very complex—making it hard to communicate the nature of the solutions in an understandable manner to a decision maker—then expressing preferences and understanding the trade-offs between the objective functions becomes challenging.

Instead of the decision maker providing preferences before or after the optimization process, they can also provide them *during* the process. This brings us to the third category of methods, which are known as *interactive multiobjective optimization methods* (Miettinen, 1999; Branke et al., 2008). Interactive methods allow a decision maker to iteratively provide preferences and see what kind of solutions are available. In practice, the decision maker is incorporated in the optimization process. Interactive methods allow the decision maker to express preferences multiple times during the optimization process, which allows the decision maker to also adjust them. Moreover, because the computed solutions depend on the preferences provided, it is possible to show the decision maker smaller samples of the available Pareto optimal solutions that are the most compatible with the preferences. This means that interactive methods do not just allow a decision maker to explore the problem being solved, e.g., the nature of the available solutions and their trade-offs, but also allow the decision maker to adjust their preferences to be more realistic given the available solutions. In other words, interactive methods enable the decision maker to *learn* about the problem being solved and their own preferences. Because in this thesis we are most concerned about how to better support decision makers in solving multiobjective optimization problems, we will focus solely on interactive methods in which the decision maker plays a central role.

Therefore, at its core, multiobjective optimization is about developing methods to support a decision maker in finding their best solution to a multiobjective optimization problem. Because the problems being solved are often real-life problems, the decisions made based on the found solutions have real-life consequences. This means that when developing multiobjective optimization methods, we must consider other aspects than just the efficient search for Pareto optimal solutions

[2] There are also multiobjective optimization methods that omit the decision maker and preferences altogether, known as *no-preference* methods. But because this thesis focuses on the decision-support aspects of multiobjective optimization, we will omit any discussion of these methods.

to the problem being solved. Because of the possible consequences of the decisions made, a decision maker might have to justify their decision at some point to themselves and any possible stakeholders, for instance. While the preferences expressed by the decision maker are utilized to find potential solution candidates, the process of finding optimal solutions can still present itself like a black-box to the decision maker. Needless to say, relying on black-boxes in potentially critical decision-making tasks does not promote confidence in the decision maker or the parties subject to the consequences of the decisions made, which makes justifying any decision challenging. One might also question whether relying on black-boxes in any form of decision-making is good decision-making at all.

A very similar problem with black-boxes has been previously identified in the field of artificial intelligence, where powerful black-box machine learning models have found many practical uses and applications in a wide array of data analysis tasks (MacKay, 2003; Bishop and Nasrabadi, 2006; Kaplan, 2016; Jordan and Mitchell, 2015). It has been realized that when decisions are to be made based on the predictions generated by black-box models, we must be able to understand why and how a particular prediction has been made by the model. Only then we can expect to be able to also justify the decisions to stakeholders and the parties subject to their consequences.

To address the issue of the black-box nature of many machine learning models, the field of *explainable artificial intelligence* was born (Biran and Cotton, 2017; Gunning and Aha, 2019; Arrieta et al., 2020; Kamath and Liu, 2021). While the field has a rich history, beginning in the early to mid 18th century, it is hard to pinpoint its exact conception. However, the introductory chapter of the book by Kamath and Liu (2021) gives a fair historical overview of explainable artificial intelligence. In the field, the concept of *explainability* has been adapted to shine a light on the wide variety of black-boxes found in machine learning applications and research to better understand the models and how they work. This, arguably, can lead to better machine learning aided data analysis and decision-making.

However, explainability is much more than just an academic curiosity. The importance of the explainability of decisions made utilizing algorithms and data-based methods is reflected in the General Data Protection Regulations set by the European Union, especially in Recital 71[3], part of which states that: "In any case, such processing should be subject to suitable safeguards, which should include specific information to the data subject and the right to obtain human intervention, to express his or her point of view, to obtain an explanation of the decision reached after such assessment and to challenge the decision." The focal part in this extract is the *right to an explanation*. As argued by Goodman and Flaxman (2017), while these regulations can pose large challenges for industries heavily relying on black-box data-based machine learning approaches, the regulations can also challenge researchers to develop new algorithms and methods for decision-making (support) that promote explainability.

[3] https://www.privacy-regulation.eu/en/r71.htm, accessed May 9, 2024.

In this thesis, we have taken the initial steps to answer the call to promote and explore explainability in the context of interactive multiobjective optimization to support better decision-making. This brings us to the three research questions of this thesis:

RQ1: **In which ways can explainability aid a decision maker in finding their most preferred solution to a multiobjective optimization problem when applying an interactive multiobjective optimization method?**

RQ2: **Does incorporating explainability in interactive multiobjective optimization methods improve their decision-support capabilities?**

RQ3: **In which ways can explainability improve the justifiability of the decision made when applying interactive multiobjective optimization methods?**

We explore these three research questions from three different perspectives presented in Articles I, II, and III.

To begin, we will discuss the needed background to understand the contents of this thesis in Chapter 2. Then, we will present the first of the three perspectives in Chapter 3, where we will see how explainability can aid a decision maker in providing and understanding the impact of preferences in an interactive multiobjective optimization method. Here, we will see how we can generate suggestions based on explanations, and the wishes expressed by a decision maker, to aid the decision maker in providing new preferences. That is, we leverage explanations to aid the decision maker in modifying the preferences expressed during an interactive method. The preferences modified with the aid of the explanation can lead to solutions that better match the expectations of the decision maker.

The second perspective is presented in Chapter 4, where we will explore the potential of explainability in modeling the preferences of a decision maker in an interactive multiobjective optimization method. Because the availability of preferences is paramount to the success of interactive methods in finding preferred solutions, the modeling of said preferences can be advantageous, e.g., it can allow us to fully rank the available Pareto optimal solutions to a multiobjective optimization problem according to the modeled preferences. A preference model is, however, not easy to elicit from a decision maker, and the model can also be hard to understand. If the preference model could be not only expressed but also primed by the decision maker in terms of understandable "if-then..."-rules, then we could be able to learn explainable preference models. While we do not quite reach this goal in the presented article, we have set the gears in motion towards future studies to explore these possibilities in more depth.

And lastly, we will discuss the third perspective in Chapter 5. We will see how explainable machine learning has the potential to help decision makers better understand the connection between decision variables and objective function values near their preferred solutions. We will see how the vast populations of solutions generated by population-based evolutionary multiobjective optimization methods (c.f., Section 2.1.3) can be exploited to learn an explainable rule-based machine learning model that classifies solutions based on whether they are close to the preferences expressed by a decision maker or not. These rules then describe these solutions in terms of the decision variable values. This information can be especially useful to more technically inclined decision makers, such as engineers tackling design problems, supporting them in providing further preferences, and giving them insights about the formulation and model of the multiobjective optimization problem itself.

All of the three works discussed in Chapters 3, 4, and 5 will present a new interactive multiobjective optimization method that incorporates explainability in different ways to support the decision maker. In developing these methods, one of the fundamental goals has been in not just devising the method but also making its implementation freely available to others. Hence, we will discuss the importance of open source software in developing and applying interactive multiobjective optimization methods in Chapter 6 where we will discuss DESDEO, the open source software framework for interactive multiobjective optimization presented in Article IV. DESDEO has played a critical role in enabling the development and experimentation that has led to the conception of the three interactive multiobjective optimization methods discussed in Chapters 3, 4, and 5. Its open source nature means that it is fully available to others to freely use and build on. Hence, DESDEO will enable future research and exploration of interactive methods incorporating explainability.

Lastly, in Chapter 7, we will discuss the conclusions of this thesis. We will return to the three research questions (RQ1–3) and provide explicit answers to them from the three different perspectives explored in Chapters 3, 4, and 5. In addition, we will discuss the many future research directions and questions the work presented in this thesis has brought forth. To support these directions, DESDEO, discussed in Chapter 6, will inevitably play an important role in the foreseeable future as well. In sum, after the conclusions, we will have an initial picture of the possibilities of explainability for supporting decision makers who apply interactive multiobjective optimization methods, as well as the potential of the concept of explainability in future works to come. Therefore, this thesis begins to establish the foundation of a new field in multiobjective optimization: *explainable interactive multiobjective optimization*, by laying some of its first foundational bricks.

2 BACKGROUND CONCEPTS

In this chapter, we introduce the concepts needed for the rest of this thesis. We begin our discussion in Section 2.1 where we delve into the central theoretical concepts in multiobjective optimization and examine approaches for solving multiobjective optimization problems, particularly scalarization-based and evolutionary, or rather population-based, methods. Then, in Section 2.2, we discuss in more depth the decision-support aspects of multiobjective optimization focusing on the decision maker, preferences, and interactive methods in particular. We then discuss some central ideas of machine learning and explainability in Section 2.3, giving a brief introduction of the topics. Our discussion then naturally flows into Section 2.4 in which we discuss explainability as a concept in the context of multiobjective optimization and give a brief survey on how explainability and similar concepts have been studied in the literature.

2.1 Multiobjective optimization

The theory and concepts discussed in this section are fundamental in multiobjective optimization. The contents of Section 2.1.1, where we discuss the multiobjective optimization problem definition and its central concepts, are based on the books by Miettinen (1999), and Sawaragi, Nakayama, and Tanino (1985). Likewise, the contents of Section 2.1.2 discussing scalarization and scalarization-based methods are also based on the book by Miettinen (1999). The contents in the last section, Section 2.1.3, are based on the collection edited by Branke, Kalyanmoy, Miettinen, and Slowiński (2008). These books, and the collection, have been the most influential elementary texts regarding the work presented in this thesis.

2.1.1 Problem definition

A multiobjective optimization problem can be defined as:

$$\min_{\mathbf{x} \in X} \left[F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})) \right]. \tag{1}$$

In (1), $F(\cdot)$ is a vector valued function with $k \geq 2$ components $f_i(\cdot)$ and $i \in 1, \ldots, k$, which are known as the *objective functions* to be minimized. The objective functions are assumed to be scalar valued and are functions of the *decision vectors* $\mathbf{x}$ with n components $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$ known as the *decision variables*. The collective domain and co-domain of the objective functions are known as the *decision space* and the *objective space*, respectively. The elements of the objective space are known as *objective vectors*, and are represented by Z, i.e., $Z = F(\mathbf{x})$, where $\mathbf{x} \in X$.

The set X in (1) is known as the *feasible set* of the decision variables, and it is defined by the *constraints* of the multiobjective optimization problem. The constraints are either *box-constraints*, which define a lower and upper bound for each decision variable; or *constraint functions*, which consist of functions of one or more of the decision variables, and are expressed as inequality or equality expressions, e.g., $g(\mathbf{x}) \leq 0$ or $h(\mathbf{x}) = 0$, where $g, h : \mathbb{R}^n \to \mathbb{R}$ are constraint functions. When defining a multiobjective optimization problem, it is practical to treat all of the objective functions to be either minimized or maximized. In the case we wish to maximize an objective function instead of minimizing it, we can multiply the function by -1.

To define what an optimal solution to a multiobjective optimization problem (1) means, we employ the concept of *Pareto optimality*. Pareto optimality is defined as follows: a decision vector $\mathbf{x}^* \in X$ is said to be Pareto optimal if, and only if, there does not exist another decision vector $\mathbf{x} \in X$ such that

1. $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all $i = 1, \ldots, k$, and
2. $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one $j \in \{1, \ldots, k\}$.

In other words, $\mathbf{x}^*$ is Pareto optimal if no other decision vector makes all the objective functions better or equal, and at least one of them strictly better. In fact, the preceding definition defines what is known as *strong Pareto optimality*, but we will refer to it as simply *Pareto optimality*. Another type of Pareto optimality is *weak Pareto optimality*, which is defined as: a decision vector $\mathbf{x}^* \in X$ is said to be weakly Pareto optimal if there does not exist another decision vector $\mathbf{x} \in X$ such that $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for all $i = 1, \ldots, k$. In other words, $\mathbf{x}^*$ is weakly Pareto optimal if no other decision vector strictly improves all the objective functions. Other types of Pareto optimality also exist, but for the purpose of this thesis, the concepts of Pareto optimality and weak Pareto optimality are sufficient.

The set of all Pareto optimal solutions is known as the *Pareto optimal set*, and its image—consisting of objective vectors—as the *Pareto optimal front*, or just *Pareto front*. One important characteristic of the Pareto front is that we cannot change from one selected objective vector to another in hopes of improving a certain objective function's value without worsening the value of at least one other objective function. That is, a decision maker has to be mindful of various trade-offs when selecting and comparing different objective vectors on the Pareto front. The full extent of the Pareto front is rarely known, because the number of Pareto optimal solutions is often uncountable, especially if we assume a continuous decision space. In practice, we are generally able to only approximate[1] the Pareto optimal front.

To characterize the Pareto front, the concepts of an *ideal point*, $\mathbf{z}^{\star}$, and a *nadir point*, $\mathbf{z}^{\text{nad}}$, are defined. The ideal point is defined to consist of the best values of each objective function on the Pareto front. Its components can be readily calculated by simply optimizing each objective function individually, while still adhering to the underlying constraint functions. The nadir point is then defined to consist of the worst objective function values on the Pareto front. However, calculating the nadir point is not straightforward because it may not correspond to the image of a feasible solution. The true nadir point would require knowledge on the full extent of the Pareto front, which is generally not available.

The nadir point is therefore often approximated. A *payoff table* is a common method to get a rough estimate of the point (Benayoun et al., 1971). However, the components of the nadir point computed with the payoff table can be inaccurate, e.g., far too low or far too high, as explored by Korhonen et al. (1997) and Weistroffer (1985). Another approach to approximate the nadir point is to utilize heuristics, such as evolutionary metaheuristic (c.f., Section 2.1.3) approaches (Deb and Miettinen, 2009; Deb et al., 2010), or other heuristic approaches (Dessouky et al., 1986; Korhonen et al., 1997), for instance. There are cases in which an exact nadir point can be calculated, but these are often special cases, and require the multiobjective optimization problem to be, e.g., linear (Isermann and Steuer, 1988; Alves and Costa, 2009). However, these exact approaches tend to be much more computationally demanding than heuristic-based approaches, and do not scale very well with an increasing number of objective functions. Computing the nadir point is nonetheless still an active subject of research in the field of multiobjective optimization. And as we shall see, the nadir point plays an important role in the works discussed in the included articles as well.

Another important concept closely related to Pareto optimality is *dominance*, which is defined as follows: an objective vector $\mathbf{z}^1 \in Z$ is said to *dominate* another objective vector $\mathbf{z}^2 \in Z$, if, and only if,

1. $\mathbf{z}^1_i \leq \mathbf{z}^2_i$ for all $i = 1, \ldots, k$, and
2. $\mathbf{z}^1_j < \mathbf{z}^2_j$ for at least one $j \in \{1, \ldots, k\}$,

where the sub-indices of $\mathbf{z}^1$ and $\mathbf{z}^2$ refer to the components of the objective vectors. Utilizing dominance, a *dominance relation* can be established for any two objective vectors $\mathbf{z}^1$ and $\mathbf{z}^2$, which can be notated as $\mathbf{z}^1 \succ \mathbf{z}^2$. Likewise, weak dominance can be defined much like weak Pareto optimality, that is, $\mathbf{z}^1$ is said to *weakly dominate* $\mathbf{z}^2$ if $\mathbf{z}^1_i \leq \mathbf{z}^2_i$ for all $i = 1, \ldots, k$, and can be expressed as $\mathbf{z}^1 \succeq \mathbf{z}^2$.

[1] *Approximate* can come in two types: i. we compute a *representation* of the Pareto front, i.e., we compute some, but not all, of the vectors on the front, which are Pareto optimal; or ii. we compute a set of objective vectors that are an *estimate* of the true Pareto optimal front, but the vectors are not Pareto optimal, instead, they are mutually non-dominating (introduced shorty). In reality, approximations of Pareto fronts are often combinations of both types i. and ii., i.e., we compute a representation of an estimate of the true Pareto optimal front, which we call an approximation in this thesis, unless noted otherwise.

Dominance is an important concept, especially in evolutionary multiobjective optimization discussed in Section 2.1.3, because it allows us to establish a partial ranking of an arbitrary set of objective vectors. That is, we can take any set of objective vectors and establish a subset of the vectors that are not dominated by any of the other vectors in the set. Such a set is then called a *non-dominated set*. Notice that the Pareto front is, by definition, a non-dominated set, but that any non-dominated set is not necessarily a Pareto front. Nevertheless, in practice, non-dominated sets are often used to approximate the Pareto front of a multiobjective optimization problem.

Thus far, we have only discussed the definition of a multiobjective optimization problem (1) and the characteristics of its optimal solution set, i.e., the concepts of Pareto optimality, the ideal and nadir points, and dominance. Next, we will have a look at two of the central approaches when it comes to finding these optimal solutions and solution sets—and their approximations—in Sections 2.1.2 and 2.1.3.

2.1.2 Scalarization-based optimization methods

The quintessential conundrum in solving multiobjective optimization problems (1) lies in the matter that we cannot optimize each of the objective functions simultaneously. To circumvent this, we can transform the original multiobjective optimization problem into a single-objective optimization problem, which we know how to solve. This transformation is known as a *scalarization*, and can be formally defined as a higher-order function $S : (F : X \to Z) \to (X \to \mathbb{R})$. The function S is known as a *scalarization function*. That is, the function S transforms a multiobjective optimization problem with domain X and co-domain Z into a scalar valued function. Multiobjective optimization methods that utilize scalarization are known as *scalarization-based (multiobjective optimization) methods* (Miettinen, 1999).

To solve a multiobjective optimization problem with scalarization, we first need to define a *scalarized problem*

$$\min_{\mathbf{x} \in X} S\left(F(\mathbf{x}); \mathbf{p}\right), \tag{2}$$

where $\mathbf{p}$ is a vector of one or more parameters of the scalarization function S. As we can notice, the scalarized problem in (2) is now a single-objective optimization problem, which can be solved readily with an adequate single-objective optimizer, such as sequential quadratic programming methods (Nocedal and Wright, 1999) and methods based on the Broyden–Fletcher–Goldfarb–Shanno algorithm (Avriel, 2020)[2].

Before we discuss about the properties of the solutions found by solving a

[2] These mentioned methods are adequate for non-linear (un)constrained continuous problems, and they assume the problem to be differentiable, e.g., they are gradient-based. However, these methods cannot guarantee the global optimality of a found solution, unless the problem being solved is convex. To put it simply, it is extremely important that, when solving a scalarized multiobjective optimization problem, a proper single-objective optimization method is chosen, which is suitable given the characteristics of the problem.

scalarized problem (2), we will look at a couple of concrete examples of scalarization and their corresponding scalarized problems. We begin with the *weighted sum* defined as

$$\begin{aligned} \mathrm{S}_{\mathrm{ws}}(F(\mathbf{x});\mathbf{w}) &= \sum_{i=1}^{k} w_i f_i(\mathbf{x}), \\ &\text{where} \quad w_i \geq 0 \text{ and } \sum_{i=1}^{k} w_i = 1; \end{aligned} \tag{3}$$

and its corresponding scalarized problem

$$\min_{\mathbf{x}\in X} S_{\mathrm{ws}}\left(F(\mathbf{x});\mathbf{w}\right). \tag{4}$$

In (3) and (4), the vector $\mathbf{w} = [w_1, \ldots, w_k]$ consists of *weights*, and it is an example of the parameters $\mathbf{p}$ of a scalarization function. We can quickly notice that by varying the values of the weights, we can emphasize the contribution of certain objective functions over the others in the first sum term in (3). Likewise, by altering the weights and solving the scalarized problem, we are able to produce a diverse set of solutions.

However, a solution to (4) is Pareto optimal only if it is unique or the weights are greater than zero. Otherwise, the solutions are weakly Pareto optimal. Beyond these limitations, the weighted sum can only find Pareto optimal solutions that lie on the vertices of the Pareto front, that is, any solution that lies on the edges of the front, cannot be found by the weighted sum. In other words, even in problems with convex co-domains, the weighted sum struggles to find some Pareto optimal solutions. And needless to say, for problems with a non-convex co-domain, the weighted sum is not able to find all Pareto optimal solutions either. For a mathematical discussion on the properties of the weighted sum, see the work of Censor (1977).

Another issue with the weighted sum scalarization (3) is the question of how to set the weights. From a decision-support perspective, the weights can be given by a decision maker, but issues arise in this. Weights are often hard to interpret by a decision maker, which means that they might not be able to express their preferences adequately when supplying weights. But even more serious is the issue with the weights not necessarily producing solutions that reflect the preferences implied by the weights. That is, a weighted sum can produce solutions with the worst value in the objective function that has the greatest weight, which is very counter intuitive as illustrated in the works by Steuer (1986) and Tanner (1991), for instance.

The problems of not being able to generally find all the existing Pareto optimal solutions to a multiobjective optimization problem, and the ambiguity of weights to a decision maker, make the weighted sum scalarization (3) not desirable from a decision-support perspective. But the weighted sum can still be used as an easy-to-understand example of scalarization due to its simplicity and its illustratory power on the potential drawbacks of utilizing a poorly chosen scalarization function in scalarization-based methods. However, it should never be used in any serious application. This motivates the need for better scalarization functions than the weighted sum.

In light of the shortcomings of the weighted sum (3), we can require three properties from scalarization functions, which make them much more desirable from a decision-support perspective (Sawaragi et al., 1985):

1. when optimized, a scalarization function can be used to find *any* Pareto optimal solution;
2. every solution found by minimizing a scalarization function is Pareto optimal; and
3. if *aspiration levels* are used, the solution found by optimizing a scalarization function is *satisfying* given that the aspiration levels are feasible.

In property 3, aspiration levels are values expressed by a decision maker. As the name suggests, the decision maker *aspires* to find a solution to a multiobjective optimization problem with the expressed aspiration levels. Moreover, when the objective function values of a solution found meet, or are better than, the aspiration levels, a solution is deemed satisfying. We made evident that the weighted sum scalarization does not generally meet the first and second properties. As for the third property, the weighted sum does not make use of aspiration levels, and as we discussed earlier, the weights do not do a good job in reflecting the decision maker's preferences.

An example of a scalarization function that can meet the three properties for scalarization functions listed above, is the *achievement scalarizing function* (Wierzbicki, 1980, 1982) defined as

$$S_{\mathrm{AS}}(F(\mathbf{x}); \bar{\mathbf{z}}, \mathbf{z}^{\star}, \mathbf{z}^{\mathrm{nad}}) = \max_{i=1,\dots,k} \left[\frac{f_i(\mathbf{x}) - \bar{z}_i}{z_i^{\mathrm{nad}} - (z_i^{\star} - \delta)} \right] + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{\mathrm{nad}} - (z_i^{\star} - \delta)}, \tag{5}$$

where $\mathbf{z}^{\star} = \left[z_1^{\star}, \dots, z_k^{\star}\right]$ and $\mathbf{z}^{\mathrm{nad}} = \left[z_1^{\mathrm{nad}}, \dots, z_k^{\mathrm{nad}}\right]$ are the ideal and nadir points of a multiobjective optimization problem (1), respectively. Moreover, in (5), $\bar{\mathbf{z}} = [\bar{z}_1, \dots, \bar{z}_k]$ is a *reference point* consisting of aspiration levels $\bar{z}_j$, where $j = 1, \dots, k$, provided by a decision maker, while ρ and δ are small scalar values. The term $(z_i^{\star} - \delta)$ in (5) is sometimes labeled as a *utopian point*, which is an objective vector that is strictly better in each of its components compared to the ideal point, and it is included to avoid division by zero. The sum term in (5) is also known as an *augmentation term*. Thanks to the augmentation term, when the corresponding scalarized problem of (5) is solved, the solutions found can be guaranteed to be Pareto optimal[3]. Examples of other scalarization functions that can fulfill the three desirable properties are the scalarization function used in the STOM[4] method (Nakayama, 1995) and the augmented version of the scalarization function based on the GUESS method (Buchanan, 1997; Miettinen and Mäkelä, 2002). For a

[3] Given that we use an appropriate solver. In fact, the solution found is *properly* Pareto optimal (Miettinen, 1999) since the trade-offs are bounded.

[4] Standing for the *satisfying trade-off method*.

collection of scalarization functions and their varying properties, see the work by Miettinen and Mäkelä (2002).

Thus, the type of scalarization function chosen can guarantee the solutions of a scalarized problem to be (weakly) Pareto optimal. But it is also important to choose an appropriate solver to have a guarantee of the optimality of the solutions found. A successful choice of a solver necessitates knowledge on the properties of the multiobjective optimization problem being solved, such as variable types (e.g., binary, integer, or continuous), and the properties of the objective functions (e.g., (non-)linearity, (non-)convexity, and (non-)differentiability). Additionally, if the problem has constraint functions, the selection of a solver depends on the properties of these functions as well. If we have access to all this information, then utilizing scalarization-based methods is ideal. However, many real-world problems are based on data, and are modeled in a way that the corresponding multiobjective optimization problem is a black-box. For example, this can be the case with simulation-based problems. In such cases, we have hardly any guarantee of the optimality of the solutions found by scalarization-based approaches. This also means that computing only a single solution at a time, which is typical in scalarization-based approaches, makes less sense, since the solution is most likely not even Pareto optimal. This calls for alternative multiobjective optimization methods, which are able to produce more than one solution at a time, and fare well in the absence of exact knowledge on the properties of the multiobjective optimization problem being solved. We will talk about a particular class of such methods in the next section.

2.1.3 Population-based evolutionary multiobjective optimization methods

Whereas scalarization-based approaches are adequate for producing single and accurate solutions, *evolutionary multiobjective optimization methods* (Schaffer, 1985; Ishibuchi et al., 2008; Zhou et al., 2011)—referred to as just *evolutionary methods*—are capable of generating multiple solutions at a time to a multiobjective optimization problem (1). Evolutionary methods are based on metaheuristic, relying on stochastic methods. Therefore, there is no guarantee of the Pareto optimality of the solutions found. As already mentioned, evolutionary methods are often employed in solving multiobjective optimization problems with a black-box nature, such as in data-based, surrogate-based[5], and simulation-based problems, where the exact properties of the objective and constraint functions being optimized are not known. However, evolutionary methods can also be used to solve scalarized problems (2). In such cases, multiple solutions are produced as well, but any guarantee of optimality is lost. Thus, the comparison between scalarization-based methods—where the problem properties are known and a proper solver is utilized to compute an accurate optimal solution—and evolutionary methods, boils down to *quality vs quantity*.

A central concept in evolutionary methods is a *population*, which is the reason

[5] For example, when objective functions are modeled with machine learning.

they are also called population-based methods[6]. A population consists of *individuals* that each represents a candidate solution to a multiobjective optimization problem. An evolutionary method is then a method that tries to iteratively modify, or *evolve*, a population in hopes of improving its individuals in each iteration, or *generation*. How an evolutionary method evolves a population, can be described in terms of six general steps, or *operations* acting on the population, which are

1. *initialization*,
2. *evaluation*,
3. *selection*,
4. *variation*,
5. *elitism*, and
6. *termination*.

Out of the above six steps, initialization and termination are usually done only at the beginning and end of an evolutionary method, respectively. The steps evaluation, selection, variation, and elitism are each iteratively applied to a population once during each generation. Next, we will discuss each of the above six steps in more detail.

Initialization consists of creating an initial population in an evolutionary method. It is important that the initial population has enough variety and that it covers as well as possible the whole domain of the multiobjective optimization problem being solved. This way we can avoid the population converging prematurely. A popular way to initialize the population is utilizing *Latin hypercube sampling* (Mckay et al., 2000). Latin hypercube sampling is a sampling method that ensures a more thorough and representative sampling of the domain of a multiobjective optimization problem than, e.g., random sampling.

Evaluation consists of evaluating each individual in a population to compute its corresponding objective vector and whether the solution represented by the individual is infeasible or not. Moreover, a *fitness function* can be utilized to rank each individual to achieve some ordering for the solutions. We get a *fitness value* for an individual by evaluating the fitness function with the individual. This allows the evolutionary method to choose which individuals of the population continue to the selection step, e.g., by discarding individuals corresponding to infeasible solutions—which is an example of an approach in evolutionary methods to handling constraints[7]—and individuals whose fitness value fails to meet some threshold value.

Selection consists of selecting individuals that are used in the variation step. Selection is often a stochastic procedure that aims to select above-average individuals to continue to an intermediate mating pool during variation. An example of a selection procedure is tournament selection (Miller et al., 1995). In tournament

[6] Other population-based methods can also be utilized for multiobjective optimization, such as particle-swarm and other nature-inspired methods, but these methods are beyond the scope of this thesis.

[7] For more examples of constraint handling in evolutionary methods, see, e.g., the work of Li et al. (2016).

selection, several individuals are chosen at random from the population and the individual with the best fitness value is selected in a tournament-esque setting.

Variation involves creating new individuals, or modifying existing ones, through various processes. Examples of these are *mating*, where individuals from the selection step are combined into new individuals, for example, through *simulated binary cross-over* (Deb et al., 1995), where two individuals are combined to produce offspring by exchanging their decision variable values, mimicking the biological cross-over; and *mutation*, where individuals are randomly modified, for example by altering the decision variable values in the solution represented by the individual. In other words, mating promotes exploitation of individuals with an already good fitness value; while mutation promotes exploration by randomly modifying individuals. Both mating and mutation are stochastic procedures and are often both employed in the variation step. The variation step produces a new population.

Elitism combines the new population generated by the variation step with the older population from the selection step, and then chooses to keep the better individuals from both populations, combining them into a new one. Elitism is very similar to selection—both operations discard individuals, favoring the better ones. The difference is that elitism is about combining two populations, whereas selection is concerned only with a single population. Nevertheless, it is not uncommon to see elitism and selection being combined in evolutionary methods, as done in Article III, for instance. The purpose of elitism, and of selection as well, is to ensure the overall monotonic and non-decreasing increase in the *quality* of the population, i.e., by ensuring that individuals with a good fitness value are not lost.

Termination is the last step and plays an important role in determining when an evolutionary method should stop based on a *stopping criterion*. This criterion can be, for instance, a desired quality for a population—once this quality is achieved by a population, the evolutionary method stops, and the population of the latest generation is the method's output. Another termination criterion is setting the number of generations in advance. When the set number of generations is reached, the method stops, and the final population of the method is its output.

Evolutionary methods can be categorized into three distinct types: indicator- (Zitzler and Künzli, 2004), domination- (Deb et al., 2002), and decomposition-based (Zhang and Li, 2007) methods. However, methods that do not fit into any of these three categories also exist. For more recent reviews on existing evolutionary methods, refer to the works of Antonio and Coello (2017), and Chugh et al. (2019). However, regarding this thesis, indicator-based methods are of most interest to us, therefore they will be described in more detail.

An *indicator-based evolutionary method* relies heavily on an *indicator*, which is a measure that can tell us how "good" a population of solutions or an individual is. Therefore, an indicator plays the role of the fitness function in indicator-based methods. An example of a popular indicator is the *hypervolume indicator*, which computes how good a population is based on the hypervolume spanned by the set of solutions represented by the population. This volume is computed based on a point of reference, for instance, the (approximated) nadir point of the problem. The hypervolume can tell us how much overall variety there is in a population, but it cannot tell us how well the population is spread, i.e., is it mostly clumped into one area or evenly spread across the whole volume. Indicators in the context of evolutionary methods are a widely studied topic in the field of multiobjective optimization. For an extended discussion of various indicators and their mathematical properties, see the works by Zitzler et al. (2003), Pour et al. (2022), and Afsar et al. (2023).

While studies exist where it has been proven that in some cases the convergence to Pareto optimality of the solutions found by evolutionary methods can be guaranteed, e.g., (Deb et al., 2007, 2015), the assumptions made are problematic to generalize to real-life problems, which are often data-driven and may consist of *online data*—that is, new data becomes available in real-time, possibly even during the optimization process. Another issue arising with the assumptions is the required time for an evolutionary method to converge. Elapsed time will be a critical factor especially in the context of interactive multiobjective optimization methods, which we will discuss in more detail in Section 2.2. Generally speaking, when employing evolutionary methods, especially in real-life data-driven problems, it is safer to assume a set of non-dominated solutions produced by an evolutionary method to be always an approximation of the corresponding true Pareto optimal set.

2.2 Decision-support in multiobjective optimization

Thus far, we have only discussed the properties of multiobjective optimization problems and approaches to solve them in Section 2.1. In this section, we discuss what kind of a role the decision maker can play in multiobjective optimization, and consequently, how multiobjective optimization can be utilized as a decision-support approach. We begin our discussion with a more in-depth discourse concerning the decision maker in Section 2.2.1. We then look at how the preferences of a decision maker can be accounted for in multiobjective optimization, first in so-called a priori and a posteriori methods in Section 2.2.2, and then in interactive methods in Section 2.2.3. Lastly, we take a brief dive into the topic of preference modeling in multiobjective optimization in Section 2.2.4.

2.2.1 The decision maker and preferences

As already discussed in Chapter 1, a decision maker is an expert with domain knowledge related to the multiobjective optimization problem being solved. When discussing decision makers in the context of multiobjective optimization and decision-support, a decision maker is assumed to convey some form of *rationality*. The rationality of a decision maker is not a trivial concept and an extensive discussion of it is well beyond the scope of this thesis. For an extended discussion on what rationality might include, see the book by Bermúdez (2009).

That being said, we have made some assumptions when it comes to the rationality of a decision maker. These assumptions are:

1. The decision maker will always prefer a non-dominated solution over a dominated one, or a Pareto optimal solution over a non-Pareto optimal one.
2. The decision maker has preferences that can be utilized to order non-dominated and Pareto optimal solutions alike.
3. The decision maker is able to provide said preferences in some quantifiable form, e.g., as aspiration levels.
4. The decision maker is human and has limited cognitive capabilities.

To further elaborate on these assumptions, let us start from the first point, which is often the main assumption made when a decision maker is assumed to be rational in multiobjective optimization. Since a non-dominated or Pareto optimal solution to a multiobjective optimization problem is strictly better, that is, it has better values for all objective functions than a dominated or non-Pareto optimal solution, there is no trade-off to be made when switching from the latter type of solution to the former. The switch will lead to an objectively better solution. This assumption heavily relies on the fact that the objective function values represent meaningful values to a decision maker, and that the objective function values tell everything the decision maker needs to know about the multiobjective optimization problem when it comes to decision-making. In the case this assumption does not hold, it means that the multiobjective optimization problem has been ill-defined. However, sometimes the "goodness" of a solution may depend on other factors, such as the decision variable values as has been noted in e.g., Article III and (Kania et al., 2022), which can help a decision maker choose between non-dominated or Pareto optimal solutions. This does not contradict our first assumption. However, if a decision variable value causes a decision maker to prefer a dominated or non-Pareto optimal solution over a non-dominated or Pareto optimal solution, then the decision variable value should be modeled as an additional objective function instead. Lastly, the first assumption is also the main reason we can focus on just the Pareto optimal set or a non-dominated set of solutions in multiobjective optimization methods.

The second assumption regarding the decision maker is important for us to be able to order non-dominated and Pareto optimal solutions. This assumption actually highlights the importance and role of preference information: it breaks the stalemate in comparing non-dominated and Pareto optimal solutions in a purely mathematical context without any auxiliary information available on top of the problem formulation. Depending on the preferences available, we may be able to fully order solutions or to further partially order them to narrow down the available solutions to a set of interesting ones. This is crucial when we want to utilize multiobjective optimization as a decision-support tool.

The third assumption made regarding a decision maker is closely related to the second assumption. In fact, it is a necessary prerequisite for the second point to even make sense. We will later discuss the various types of preferences a decision maker can be expected to provide.

The fourth, and last assumption, we make is perhaps the most intriguing one. The decision maker being a human means they have human characteristics. They have a mind of their own with their own experiences and opinions, they can get tired, be inconsistent, and be forgetful. Likewise, they can also be very adaptive and quick to learn, and have a fascinating intuition when it comes to the multiobjective optimization problem being solved. These characteristics are just scraping the surface, and it is well beyond the scope of this thesis to explore the psychology and cognitive abilities of a human decision maker. But it is nonetheless essential to keep these very important facts in mind—decision makers are human beings after all. When it comes to this thesis, we can limit the human nature of decision makers into two observations: i. the decision maker is able to learn, for instance, about the available solutions to a problem and adjust their preferences accordingly; and ii. the decision maker can get tired, e.g., in providing further and further preferences, or when exploring a large amount of available solutions. These observations play a central role and are a motivating factor in interactive multiobjective optimization discussed later in Section 2.2.3.

Lastly, it is beneficial to briefly mention what kind of preference information can be expected from a decision maker. The weights and the reference point we already saw in Section 2.1.2 are types of preference information. Additional types of preference information can come in the form of upper or lower bounds, or both, for desirable objective function values. The decision maker may also express which solutions from a discrete set of non-dominated or Pareto optimal solution they prefer or do not prefer, or they may rank discrete solutions in various ways, such as by comparing them pair-wise. These types of preferences are easily quantifiable, but types of preference that are more qualitative in nature and bound to the domain of the multiobjective optimization being solved, also exist. However, in this thesis, we will focus on strictly quantifiable preference types, which are also the type of preference that are considered in most of the literature on multiobjective optimization.

2.2.2 A priori and a posteriori methods

As already mentioned in Chapter 1, when it comes to multiobjective optimization methods, they can be divided into four categories based on *when* a decision maker provides preferences, or if at all (Miettinen, 1999). Methods that do not incorporate preferences are known as *no-preference methods*. These methods are employed when there are no preferences available, often due to the absence of a decision maker. From the perspective of this thesis, we are not interested in no-preference methods and will not discuss them further. Instead, we will focus on preference-based methods.

The first category of multiobjective optimization methods that incorporates preference information are known as *a priori methods*. As suggested by the name, preferences are incorporated in the multiobjective optimization process *before* optimization takes place. In these methods, a decision maker is first queried for preferences. The preferences are then utilized to find one or more non-dominated or Pareto optimal solutions to the problem being solved, and these solutions then constitute the final solution. From a decision-support perspective, a priori methods come with some innate issues. First, the decision maker is expected to be able to provide preferences without much knowledge on the characteristics of the available solutions—they are expected to rely heavily on their domain expertise alone. A typical example of an a priori method would be solving the scalarized problem (2) of the achievement scalarizing function (5) once with a single reference point provided by a decision maker. The solution found is then the final solution and the optimization process stops. This also shows us another innate problem with a priori methods: the solution or solutions found depend heavily on the method itself. If in our previous example we, for instance, chose to use the GUESS scalarization function (Buchanan, 1997) mentioned in Section 2.1.2, the final solution found will very likely be a different one. One may then ask the question, is it enough to provide preferences once and be content with the solution(s) found? Furthermore, if a found solution in one a priori method produces a different solution than with some other a priori method with the same preference information, then which solution is better? Because of the nature of a priori methods—provide preferences once and then optimize—these questions are very hard to begin answering. However, a priori methods do find uses, for instance, when a decision maker has little time to provide preferences, in which case these methods can prove useful.

The polar opposite of a priori methods are *a posteriori methods*. Again, as the name suggests, in these methods preferences are incorporated *after* optimization. Typically, the aim of evolutionary methods (c.f., Section 2.1.3) is to find a non-dominated set of solutions that approximates the Pareto front of a multiobjective optimization problem being solved. Once a front is available, a decision maker is then expected to select one or more solutions from it. In selecting a solution(s), the decision maker can express various types of preferences, like bounds on the objective function values to help narrow down the number of options to better fit the interests of the decision maker. While a posteriori methods do not expect a decision maker to be content right away with any of the solutions found, they do exert a lot of mental strain on the decision maker. The number of solutions in a computed front can be very large ($\sim 10^3$). It is not feasible to expect a decision maker to inspect all of the available solutions. It is actually safe to assume that most computed solutions in a front are not interesting to a decision maker. This raises another problem with a posteriori methods: a lot of the computed solutions may be not useful from the perspective of the decision maker, therefore wasting computational resources. This problem is hard to avoid because, by their nature, a posteriori methods are required to compute as many solutions as possible because we do not know beforehand what the decision maker prefers. With computationally very expensive multiobjective optimization problems, this can become an incapacitating issue. On the other hand, limiting the number of solutions computed by a posteriori methods is counterproductive—how to limit the search if we do not know the preferences of the decision maker prior to the optimization process? However, in the case a decision maker has very little information about the available solutions, a posteriori methods can play an important role in providing a needed initial insight, which can support the decision maker in providing preferences in the future.

It is clear that a priori and a posteriori methods come with some issues. In the next section, we will discuss the remaining category of multiobjective optimization methods, which aims to address the issues raised in this section.

2.2.3 Interactive methods

The final category of multiobjective optimization methods is *interactive methods*. In interactive methods, a decision maker takes an active role during the optimization process by iteratively providing preferences and inspecting the solutions found. In other words, the decision maker is able to *change* their preferences during the optimization process and *learn* about not only the multiobjective optimization problem and its solutions, but also about the feasibility of their own preferences (Miettinen et al., 2008). This allows for solutions to always be computed based on some preference information and allows the decision maker to change their preferences based on what kind of solutions are available. This process of providing preferences and inspecting solutions continues until the decision maker is happy with a solution found or wants to otherwise stop. Once the decision maker has decided to stop, the interactive method is considered terminated.

To give an example of an interactive method, let us recall the example given in Section 2.2.2 on solving the scalarized problem (2) of the achievement scalarizing function (5) utilizing a reference point provided by a decision maker. Instead of utilizing the method purely as an a priori method, we now allow the decision maker to provide new preferences once they see the solution based on the initial preferences provided. By inspecting the solution and providing new preferences, the decision maker is able to achieve at least two important things. First, the decision maker gets an idea of the feasibility of their preferences. The distance of the solution found from the provided reference point gives the decision maker a chance to see how realistic their preferences are. For instance, if the reference point is far worse than what is available (the reference point is said to be *pessimistic*), the decision maker now knows they can be much more demanding in their subsequent preferences. If, on the other hand, the reference point is way better than what is available solution-wise (the reference point is said to be too *optimistic* or *infeasible*), this can prompt the decision maker to reconsider their preferences and encourage them to think of possible trade-offs when providing subsequent reference points. This brings us to the second important thing the decision maker is able to achieve in an interactive method: learning. The decision maker can learn during an interactive solution process about not just the feasibility of their preferences, but also about the array of available solutions *based on their preferences*. This last point is crucial, because in a posteriori methods this is not the case since solutions have, by definition, been computed in the absence of preference information.

An astute reader may now be asking themselves "Did not the previous example just describe an a priori method that is applied repeatedly?" Before answering the question, let us consider the following point. An interactive multiobjective optimization method is not as straightforward to define because, in theory, any preference-based method can be applied recurrently making the process as a whole seem like an interactive one. However, this misses the whole point of *interaction*. It is not simply repeatedly applying an a priori method, we must, once again, remember that the decision maker interacting with the method is a human who is capable of learning. Therefore, we may confidently answer the questions with a resounding *"No."* because it misses the point on what interaction and interacting is. For this reason, the author of this thesis likes to think of *interactive multiobjective optimization processes*, which are enabled by interactive methods. This is because interactivity is defined by the continuous interaction between a decision maker and an optimization method, during which the decision maker can learn. For a method to be an interactive one, it must support meaningful interaction between a decision maker and the method itself. That being said, to avoid confusion, we will still refer to *interactive (multiobjective optimization) methods*, which at least in this thesis, will be synonymous with the described interactive multiobjective optimization process.

While repetitively applying an a priori method can be mistaken to be a sort of interactive method, there are multiobjective optimization methods that are inherently interactive. These are clearly different from just applying an a priori method repetitively. One example of such a method is the Synchronous NIMBUS method (Miettinen and Mäkelä, 2006), where the decision maker is asked to provide preferences that are relative to an existing solution. The decision maker is asked to classify the objective function values of a solution to either improve, improve until some specified limit, impair until some specified limit, change freely, or stay as they are. In the NIMBUS method, by design, the minimal interaction between a decision maker consists of first inspecting an existing solution, classifying the objective function values of said solution (providing preferences), and then inspecting up to four (the amount is specified by the decision maker) new solution computed based on the preferences. This minimal interaction process does not directly fit into the category of a priori or a posteriori methods. The initial solution in the NIMBUS method has been computed without preference information, as is done in a posteriori methods, but not in a priori methods. Consequently, the preferences provided by the decision maker before the actual optimization process are based on an existing solution, unlike in a priori methods. Based on these observations, we see that the NIMBUS method can be classified as an inherently interactive one.

If the above description did not yet convince the reader that interactive methods are more than just repeatedly applied a priori methods, then consider multiobjective optimization *navigation methods*, such as the NAUTILUS Navigator method (Ruiz et al., 2019). In NAUTILUS Navigator, the solution process starts from the nadir point of the multiobjective optimization problem being solved and ends on a solution on a non-dominated set or the Pareto optimal front, both referred here as the final set of solutions. Proceeding from the nadir point to the final set of solutions is known as *navigating*. During navigation, the decision maker can provide a reference point and bounds on the solution in terms of objective function values, and can change these at will. When navigating, the method approaches the final set of solutions iteration by iteration (through so-called navigation points) and the decision maker is able to see in real-time how the possible set of solutions that are still reachable from each navigation point changes without sacrifices in any objective function values. The decision maker can stop the navigation process at any time and change their preferences. Needless to say, the interaction between a decision maker and NAUTILUS Navigator is critical, and this is true for all navigation methods (Allmendinger et al., 2017) making them inherently interactive. The idea of starting from the nadir point and iteratively approaching the final set of solutions is true for all methods belonging to the NAUTILUS family of methods (Miettinen and Ruiz, 2016), and it is argued that such an approach can help a decision maker avoid anchoring effects.

In addition to the examples of interactive methods discussed thus far, there exist many more in the literature. For surveys of existing interactive methods, refer to the overviews given by Miettinen et al. (2016) and Xin et al. (2018).

However, interactive methods do come with their own challenges. One of the main issues arising in interactive methods, which is also addressed in this thesis, is: how can we support decision makers in providing preferences? This is an open research question (Belton et al., 2008). We will address this issue in three of the included Articles I, II, and III. Another issue comes with comparing and assessing the quality of interactive methods. Since the success of an interactive method as a decision-support tool is very subjective—it depends on how well a method can support a decision maker in a decision-making problem—comparing interactive methods and assessing their performance is still an open challenge (Afsar et al., 2021), but some attempts in this direction have recently been taken (Afsar et al., 2023, 2024).

In this thesis, we will discuss three interactive methods in the later sections presented as novel contributions to the field of interactive multiobjective optimization. Two of these methods are based on the reference point as preference information. The first of these is a scalarization-based method, presented in Article I, and the other is an evolutionary method enhanced by machine learning, presented in Article III. The third interactive methods is based on pair-wise comparisons of solutions, presented in Article II, and it relies on machine learning to find solutions according to the preferences of the decision maker.

2.2.4 Preference modeling

When it comes to the decision-support aspect of multiobjective optimization, we have seen in Sections 2.2.1, 2.2.2, and 2.2.3 how preferences play a key role, especially in interactive multiobjective optimization methods. We can go as far as to say that if we fully knew the preferences of a decision maker, solving a multiobjective optimization problem would be trivial. However, this is seldom the case. The natural question is then: could we model the preferences of a decision maker? We do not need to ask the decision maker for preferences to solve multiobjective optimization problems if we are able to model them.

Indeed, modeling the preferences of a decision maker in multiobjective optimization is not a new topic and has been studied in the literature (Keeney and Raiffa, 1993; Pedro and Takahashi, 2013; Wang et al., 2017). However, modeling preferences comes with many challenges. For instance, we need existing information on the preferences of a decision maker. How to get this kind of data when each decision maker is unique? Our best bet is to gather information on preferences during an optimization process, ideally during an interactive one. Still, if we assume the decision maker to provide preferences, and be able to somehow communicate how much they like each of the computed solutions in each iteration of an interactive method, this still leaves us with very few data points—only 3 to 8 according to some studies (Gardiner and Vanderpooten, 1997). And even if we assume to be able to gather enough data, then we may question how to model and account for the decision maker's learning?

Furthermore, when a decision maker learns about the multiobjective optimization problem and their preferences, their preferences change, which can make the previously gathered information on the preferences completely void at worst. We may even assume an ideal case where we have enough data and have managed to model learning; then the question is: how can we verify our preference model? The ultimate judge is still the decision maker, and we need to somehow be able to present the preference model in such a way that the decision maker can compare it to their own preferences. This can be a very abstract concept for the decision maker, which can then make it challenging for them to tell us whether the model is good or not. In the worst case, presenting such a model can even introduce a bias that may steer the decision maker to prefer solutions that they might not have preferred in the absence of the preference model. Whether such a bias is good or bad is subject to debate.

Nevertheless, a common approach in modeling a decision maker's preferences is to assume a *value function* (or *utility function*) (Keeney and Raiffa, 1993; Branke et al., 2008) defined as

$$V : \mathbb{R}^k \to \mathbb{R}. \tag{6}$$

A value function (6) maps objective vectors into scalar values, which can then be used to order the vectors. The obvious challenge in this approach is figuring out what the value function for a particular decision maker is—they obviously cannot just tell us. As an example, machine learning has been employed successfully to some degree in modeling the preferences of a decision maker utilizing value functions (Battiti and Passerini, 2010; Branke et al., 2014). However, typically value functions assume the preferences of a decision maker to be constant, which makes accounting for memory effects—and by extension, learning as well—challenging (Jarecki and Rieskamp, 2022). Nonetheless, value functions have been utilized as the underlying preference model, and it has been possible to learn the function while applying interactive multiobjective optimization methods (Branke et al., 2014). In fact, this idea has been explored in Article II, where some of the shortcomings of the typical use of value functions could be addressed.

If we compare the value function to the fitness function discussed in Section 2.1.3, we notice that they accomplish a very similar goal. The difference is that a value function is meant to reflect the preferences of a decision maker, but this may not be the case with a fitness function. However, a value function can also act as a fitness function, or even as an indicator in an evolutionary method (Thiele et al., 2009). This approach has also been explored in Article III.

2.3 Machine learning and explainability

As a concept, explainability, or at least how it is understood in this thesis, has its roots in the field of artificial intelligence and machine learning. For this reason, we will first briefly discuss what machine learning is in Section 2.3.1. Then, in Section 2.3.2, we introduce the concept of explainability and how it connects to machine learning.

2.3.1 Machine learning

Since even cognitive scientists struggle to define what *intelligence* is in the context of artificial intelligence (Legg et al., 2007), the question is well beyond the scope of this thesis, and consequently, we will not try to define the concept here either. But the one thing we can confidently agree upon is that whichever way we define intelligence, *learning* is an essential characteristic and necessary prerequisite of it (Legg et al., 2007).

Unsurprisingly, machine learning is seen as a central element of artificial intelligence. In fact, machine learning, as the name suggests, focuses on the study and application of computer algorithms and programs that are able to learn patterns from data (Bishop and Nasrabadi, 2006; Flach, 2012). These patterns can be, for instance, recognizing the contents of an image; learning trends in data, such as predicting prices in the housing market; or learning to classify tumors into benign and malign types in computer tomography scans. From a data analysis perspective, machine learning can be predictive ("What could happen?"), descriptive ("What has happened?"), or prescriptive ("What should we do?") (El Morr and Ali-Hassan, 2019; Roy et al., 2022). Likewise, machine learning methods can be categorized into supervised learning (e.g., neural networks), unsupervised learning (e.g., k-clustering), and reinforcement learning (e.g., evolutionary models) (James et al., 2013; Sutton and Barto, 2018). In this thesis, we have utilized mainly supervised learning and applied it in a descriptive and prescriptive manner in Articles I, II, and III. For this reason, we will focus on supervised learning in the following discussion.

Training a supervised learning model can be described in terms of three main steps: *training*, *evaluating*, and *testing*. Data plays a crucial role in supervised learning so it is no surprise that a data set used for training the model is also further divided into *training data*, *evaluation data*, and *testing data*. We will discuss these three steps and their connection to the training data in further detail next.

Training. The first step, training, consists of supplying a supervised machine learning model sets of input and output pairs. The inputs are often symbolized by X^{train}, and the outputs by Y^{train}. The inputs and outputs are further aggregated into pairs, or tuples, $\left(X_i^{\text{train}}, Y_i^{\text{train}}\right)$, where $i = 1, \ldots, N$; and $N = \|X^{\text{train}}\| = \|Y^{\text{train}}\|$, where $\| \cdot \|$ is the number of elements in a vector. We expect a trained model to be able to *predict* Y_j^{train} given the input X_j^{train} for any $j \in \{1, \ldots, N\}$. In other words, a perfectly trained supervised machine learning model can be seen as a general function

$$M : \mathbb{R}^D \to \mathbb{R}^B, \tag{7}$$

where D is the length of the elements[8] in X^{train} and B the length of the elements in Y^{train}. Therefore, the term *predict* is nothing more than evaluating a function of the characteristics of M. How M maps its inputs into predictions is assumed to be defined by its internal parameter values represented by a vector $\bar{\mathbf{p}} \in P$, where P represents the set of all possible parameter values. The predicted values by M form a *set of predictions* $\bar{Y}^{\text{train}} \in \mathbb{R}^B$.

The goal of the training step is to find a function M for the given training data set. In practice, training consists of minimizing a *loss function*, sometimes also referred to as an *error function*[9]. A general loss function can be defined as

$$L : \mathbb{R}^D \times \mathbb{R}^B \to \mathbb{R}. \tag{8}$$

In other words, the loss function (8) is a function that maps the outputs, often also referred to as the *true outcomes*, and the predictions $\bar{Y}^{\text{train}}$ to a scalar value. The loss function is defined in a way that the smaller this scalar value is, the closer the predicted outcomes are to the true outcomes. This value is referred to as simply the *loss*. Training a supervised machine learning model is then nothing more than finding the internal parameter values $\bar{\mathbf{p}}$ of M such that the loss is minimized. This is an optimization problem that can be defined as

$$\underset{\bar{\mathbf{p}} \in P}{\operatorname{argmin}}\, L\left(M(X^{\text{train}}, Y^{\text{train}}; \bar{\mathbf{p}}), Y^{\text{train}}\right), \tag{9}$$

or in other words, find the parameter values $\bar{\mathbf{p}}$ of M that minimize the loss function L. During training, the value of L, given the optimal parameters $\bar{\mathbf{p}}$, is called the *training loss*.

Evaluating. After the parameter values $\bar{\mathbf{p}}$ that minimize (9) are found, the supervised machine learning model is evaluated. As in training, the evaluation data

[8] We are assuming the dimensions of the elements in the input and output data sets to be all the same. This might not be the case always, but in the context of this thesis, making this assumption is safe.

[9] Not to be confused with the Gauss error function "erf."

consists of inputs $X^{\text{eval}} \subset \mathbb{R}^D$ and outputs $Y^{\text{eval}} \subset \mathbb{R}^B$. The role of evaluating the machine learning model is to assess how well trained it is given data that the model has not "seen" during its training. In other words, how well the machine learning model M can predict the outcomes Y^{eval}, which is again measured by the loss function (8). This loss value is computed with the optimal parameters $\bar{\mathbf{p}}$ found during training, and is known as the *evaluation loss*, or

$$\textit{evaluation loss} = L\left(M(X^{\text{eval}}, Y^{\text{eval}}; \bar{\mathbf{p}}), Y^{\text{eval}}\right). \tag{10}$$

The evaluation loss gives an idea of how well the model M is able to generalize to data outside of the training data. The evaluation loss is often higher (worse) than the training loss, and it is typically more informative than the training loss as a metric of the performance of M.

After evaluating, the model M is trained again, usually with new training and evaluation data. Training and evaluation data form a collective pool from which the training and evaluation data sets are picked before each training and evaluation of the model. Therefore, the training and evaluation of the model consists of an iterative process, which aims to improve the evaluation loss of the model over time. This way, an iterative improvement of the machine learning model can occur either by training multiple models and choosing the model with the lowest evaluation loss, or models can retain information from a previous training and improve upon themselves in subsequent retraining. However, in some cases, it is enough to train a model only once.

Testing. After the evaluation loss of a supervised machine learning model has reached some desirable threshold, the model can be tested. Testing data is utilized to compute a loss similarly to how the evaluation loss is computed (10). It is crucial that testing data consists of data that has never been used to train or evaluate the model. The purpose of testing is to assess how well the model can generalize beyond the data it has "seen" during testing and evaluation. A low loss computed with the testing data indicates a well-trained model, which can be expected to generalize well to new data.

It is important to note that we have made a fundamental assumption when discussing supervised machine learning models. The model in (7) relies on the assumption that M can generalize to any data. This generalizability of machine learning models has been explored in statistical and computational learning theories, such as the Vapnik–Chervonenkis theory (Vapnik, 1999). While an in-depth discussion of these theories is beyond the scope of this thesis, it is nonetheless important to remember that machine learning models are statistical tools, and they should be treated as such. That is, with caution.

The description given in this section for training a supervised machine learning model is a very general one, and does not describe the myriad of nuances that may arise in the process. It is not uncommon to hear researchers and practitioners referring to training machine learning models to be as much an art as it is a science. But still, the description given in this section captures the central essence of supervised learning models: existing data is utilized to train and evaluate the model in hopes of ending up with a trained model that can generalize well to new observations. To put it simply, a machine learning model is nothing else than a statistical mapping from observations to outcomes based on—and deduced from—already known data.

2.3.2 Explainability in machine learning

Given the description of a supervised machine learning model in Section 2.3.1, one might stop and ask: "How does a machine learning model work, and why does it make these predictions based on these observations?" Machine learning alone cannot answer these questions, which, in turn, is a problem. This problem is emphasized even more when machine learning is utilized as an aid in decision-support, where the justifiably of the decisions made can be especially important.

The opaque nature of most machine learning models is perfectly captured by (7)—we are often aware only of a model's input and output, what happens in between can be a kind of a mystery. Indeed, this *black-box* nature of machine learning models has troubled researchers and practitioners to the point where a whole new field has spawned to address this issue: *explainable artificial intelligence* (Confalonieri et al., 2020; Kamath and Liu, 2021). As the name suggests, this field is devoted to researching models that can be explained and understood on a deeper level, going well beyond the information we can gather from an abstraction, such as the one in (7).

The next natural question is then: "What is explainability?" Unfortunately, even researchers in the field of explainable artificial intelligence find it challenging to define what exactly *explainability* is in the context of explainable artificial intelligence (Flora et al., 2022). For the purposes of this thesis, we approach this issue by defining first what *explanations* are, and then defining *explainability* as the property of an explainable (machine learning) model to be able to produce such explanations.

Before we continue our discussion, we must note that explainability and *interpretability* are two common terms that are sometimes used interchangeably, and sometimes they mean completely different things. We will not explore these semantics further, and we will simply understand interpretability and explainability to mean the same thing in the context of this thesis, unless specified otherwise. That said, we give two definitions, adapted from the literature (Lipton, 2018), for what an explanation is in the context of this thesis in Definition 1 and 2.

Definition 1 (local explanations)**.** An explanation is a description of the relationship between certain input values and the corresponding output values. E.g., an explanation can convey to a human information on how the output has been affected by the input, possibly allowing the human to modify the input to achieve a more desirable output.

Definition 2 (simulatability)**.** An explanation is information that can tell a human how a machine learning model works on a general level. E.g., a textual or pictorial representation of the machine learning model that can give a human a general idea of how the model works so that given an arbitrary input, a human could mentally “run through” the model and come to the correct output (according to the model) without ever having to actually evaluate the model.

In fact, the two definitions given for what an explanation is are closely bound to two main types of explainable machine learning models: *inherently explainable models*, or *interpretable models*, and *ad hoc models* (Linardatos et al., 2020)[10]. Ad hoc models aim to produce explanations for any kind of machine learning model by observing the model’s inputs and outputs. The goal is to come up with rules or a description that can tell us what kind of effect the inputs have had on the output, i.e., if we change a particular input by a certain amount, what kind of change can we expect in the output? Often these explanations are only local, which means they are valid for a subset and specific value ranges of the model’s input features. Therefore, the explanations produced by ad hoc models fit Definition 1. Examples of ad hoc models are LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017).

The second type of explainable machine learning models, inherently explainable models, are models that are self-descriptive to at least some degree. For instance, decision trees are a good example of a self-descriptive model. A human can, in practice, take a trained decision tree, visualize it, and quite literally *see* how the machine learning model works. In other words, in terms of explainability, an inherently explainable model is self-sufficient—the explanation is the model. Needless to say, this fits Definition 2 for explainability. There are arguments to be made in favor of utilizing inherently explainable models instead of ad hoc ones (Rudin, 2019). Part of the argument is that ad hoc models can give misleading explanations that might not be true at all, possibly even resulting in serious biases (Slack et al., 2020). But on the other hand, being descriptive is not necessarily enough to build any kind of understanding regarding the machine learning model, as exemplified by the Chinese Room thought experiment (Searle, 1980).

So why even bother with ad hoc models then if inherently explainable models exist? Unfortunately, the predictive power and how understandable a machine model is—understandable in the sense of Definition 2 for explainability—are inversely related. That is, the more predictive power a machine learning model has, the less explainable it is (Gunning and Aha, 2019). However, this concept is not exactly always true (Lipton, 2018). For instance, consider the inherently explainable decision tree: while a shallow tree with a couple dozen nodes can be interpreted by a human, a tree with nodes in the hundreds is already too cumbersome for most humans to even bother looking at in more detail. The same is true also for a machine learning model that is often taken to be not explainable, such as neural networks: while a deep neural network with tens of hidden layers and thousands of neurons is definitely not understandable, a neural network with just a couple of hidden layers and a few neurons can be argued to be explainable.

[10] These are also referred to as *model agnostic* and *model specific* machine learning models, respectively.

This is to say that the inherent explainability, or interpretability, of a machine learning model is not a monolithic concept, as articulated by Lipton (2018).

While there are some dangers to utilizing ad hoc models, they are still valuable because of their model agnostic nature. They can be applied to virtually *any* machine learning model[11]. One just needs to be careful in interpreting the explanations by these models, which also stands true for inherently explainable models, or, in fact, any statistical model.

2.4 Explainability in multiobjective optimization

In this thesis, explainability is explored in the context of multiobjective optimization and interactive methods. Because explainability in interactive multiobjective optimization is a novel concept, and fleshing out this concept is one of the main contributions of this thesis, we leave the more detailed discussion of explainability, and examples of it, to Chapters 3, 4, and 5, where we discuss three of the articles included in this thesis in more detail. Once we have discussed these examples, we will be capable of further elaborating on the premise of explainability in interactive multiobjective optimization in the conclusions of this thesis in Chapter 7. However, before we move onto the following chapters, we will first conclude this Background chapter with a survey on how explainability and explainability-like approaches have been explored in the context of multiobjective optimization by others in the literature in Sections 2.4.1 and 2.4.2, respectively.

2.4.1 Existing explainable approaches

In this section, we will review some of the works in the literature where explainability has been explored in the context of multiobjective optimization. The focus is solely on works where explainability has been utilized to support a decision maker during a multiobjective optimization process. We are not interested in applications of multiobjective optimization in explainable artificial intelligence. Next, we will introduce some of the most relevant works summarized in chronological order.

Wang et al. (2016) propose a diversified movie recommendation framework using a decomposition-based evolutionary multiobjective optimization algorithm. The authors emphasize the importance of diversity, novelty, and interpretability in recommendation quality, acknowledging that these properties often conflict with accuracy. The proposed framework integrates movie content information, such as genre, into a diversity objective function, making recommendations more explainable and relevant to user preferences.

Sukkerd et al. (2018) introduce a method to generate understandable justifications for decisions made in multiobjective probabilistic planning problems. This method emphasizes creating verbal explanations that clarify why certain solutions are chosen over others and how conflicting objectives are balanced. The

[11] Ignoring computational limitations.

approach involves both an explainable planning representation and an algorithm for generating contrastive justifications, effectively communicating the trade-offs and reasoning behind each decision in a format that is understandable by humans.

Zhan and Cao (2019) explore enhancing multiobjective optimization in reinforcement learning by incorporating a novel method for explaining the trade-offs between various objectives. They introduce a correlation matrix to represent the relative significance of each objective function, allowing for a more detailed understanding of how changes in one objective may affect others. This approach provides a clearer insight into the optimization process, particularly in scenarios where objectives can conflict, which can aid in decision-making.

Lin et al. (2023) present an approach to urban tunnel construction optimization under uncertain soil conditions by integrating machine learning with a robust optimization algorithm. They use SHAP values and building information modeling software, which collectively enhance the explainability and interactivity of the optimization process. SHAP values are utilized to explain the outcome of the optimization. The authors claim that this can potentially aid decision makers to better understand and solve design problems related to tunnel construction under uncertain conditions.

Osika et al. (2023) discuss advancements and possible future trends in multiobjective optimization in their recent review. As an emerging research direction, they explicitly mention explainable multiobjective optimization. They also note that the field is still in its infancy.

Yadav et al. (2023) present a new navigation method where they have combined Pareto race (Korhonen and Wallenius, 1988) with self-organized map visualizations. The authors claim the visualizations to be interpretable, thus capable in aiding a decision maker during an interactive navigation process to provide preference information. In the interpretable visualizations, the authors visualize three new metrics that they believe to be useful to a decision maker when exploring the objective vectors of a multiobjective optimization problem.

Shavarani et al. (2023) developed an interactive evolutionary multiobjective optimization method where the preference model of a decision maker is learned with binary classification decision trees. The decision trees are utilized to predict results of pair-wise comparisons from the perspective of a decision maker, which aids the evolutionary optimization process in converging to solutions interesting to the decision maker. Since decision trees are interpretable, the authors postulate that this can help the decision maker in trusting the overall optimization process.

Corrente et al. (2024) introduce XIMEA-DRSA, an interactive method where a decision maker's preferences are captured through "if..., then..." decision rules using the dominance-based rough set approach (Greco et al., 2001). These rules inform the multiobjective optimization process, guiding it towards the most favorable solutions on the Pareto front. The authors claim that this approach enhances the transparency and explainability of the optimization process, as it can demonstrate how the decision maker's expressed preferences influence the algorithm's search direction and outcome.

2.4.2 Some explainability-like approaches in the literature

In this section, we review some other relevant lines of research that explore approaches that are similar to explainability, but do not make an explicit reference to the concept. To begin, Deb and Srinivasan (2006) introduced the concept of *innovization*, where techniques for multiobjective optimization are explored as tools to innovate new designs[12] in problems with conflicting objective functions. Their methodology emphasizes the extraction of knowledge from the optimization process, rather than just focusing on finding a single optimal solution. In other words, innovization is a form of descriptive analysis for multiobjective optimization. By analyzing trade-offs and commonalities among Pareto optimal solutions in multiobjective optimization problems, innovization seeks to uncover underlying principles that can aid a decision maker in design decisions. This approach not only helps in finding optimal solutions for a given problem, but also provides insights into the problem's structure, offering a deeper understanding that can be applied to future design tasks. We can therefore clearly see that innovization can support a decision maker learn about the multiobjective optimization problem from a design discovery perspective. Innovization has been widely applied in different application fields since its conception, such as manufacturing (Dudas et al., 2013) and healthcare (Goienetxea Uriarte et al., 2017) problems.

In their recent thesis, Smedberg (2023) explores the possibilities of datamining for knowledge discovery in interactive multiobjective optimization. Knowledge discovery is a similar concept to innovization, but Smedberg criticizes that the knowledge emerging in innovization is difficult to interpret for humans. To address this, the thesis emphasizes the generation of explicit knowledge about solutions to multiobjective optimization problems, focusing on the relationships between input and output spaces to aid decision makers in understanding complex problems. Despite existing methods like innovization, there is a lack of accessible decision-support systems for practitioners to interactively extract and comprehend this knowledge. Moreover, Smedberg highlights the challenge of real-world problems, which often involve stochastic elements and large scales, leading to time-consuming optimization procedures. To better the efficiency of optimization processes, the thesis proposes incorporating this knowledge directly into algorithms, a process termed *knowledge-driven optimization*, which aims to enhance the search by avoiding less preferred solution regions. While Smedberg does not coin his approach as explainable *per se*, their thesis outlines an approach of combining ideas from data-mining and interpretable machine learning to explicate—as in, make explainable—interactive multiobjective optimization from a perspective very similar to innovization.

While the reviews of existing works in this section, and Section 2.4.1, has by no means been a comprehensive one, they capture some of the most emergent trends when it comes to applying explainability, and similar concepts, to multiobjective optimization to support decision makers. One evident issue with exploring

[12] These are particular sets or descriptions of (approximately, c.f., Section 2.1.3) Pareto optimal decision variables.

the literature for ideas similar to explainability is the vast and varying vocabulary utilized to refer to the explication of a multiobjective optimization process to a human decision maker. It would be a Sisyphean task to try and survey all the existing literature to find every possible little nuance that could be interpreted to be a form of explainability. There is a lack of established vocabulary, which makes searching comprehensively for such literature very hard. This thesis is a small step towards establishing the term *explainable* in the context of multiobjective optimization, which will facilitate making more comprehensive surveys possible in the future.

3 R-XIMO

One of the main issues in interactive multiobjective optimization is how to support a decision maker in providing preferences during an interactive optimization process (Belton et al., 2008). This issue has been addressed with explainability in Article I. The article proposes a new method, R-XIMO, which is able to support a decision maker in providing preferences and understanding the connection between preferences and the computed solutions in a multiobjective optimization context. The preference type considered consists of a reference point. R-XIMO explores a novel perspective on how explainability can be utilized to enhance interactive multiobjective optimization by considering a multiobjective optimization method to be a black-box, such as in (7). Then, the connection of the inputs (preferences) and outputs (computed solutions) of a multiobjective optimization method are explored and explained by an ad hoc type of explainable machine learning model: SHAP values. The SHAP values are then utilized to generate human-understandable text-based explanations that can guide a decision maker in modifying their preferences so that they may achieve more desirable solutions.

We discuss SHAP values in more detail in Section 3.1, and give a detailed outline of the R-XIMO method in Section 3.2. We conclude this chapter with Section 3.3, where we discuss the limitations and advantages of the R-XIMO method, and its potential impact in the field of interactive multiobjective optimization.

3.1 SHAP values

SHAP (SHapley Additive exPlanations) values, based on the SHAP framework (Lundberg and Lee, 2017), offer a way to explain individual predictions of machine learning models. SHAP values are model agnostic, which means they can be trained in an ad hoc fashion to produce explanations for any kind of machine learning model. The SHAP framework itself is based on Shapley values (Shapley, 1953), a concept with roots in game theory (Morgenstern, 1953). Shapley values are based on the idea of assigning a value to each "player" (feature in a machine learning model) based on their contribution to the overall "payout" (prediction of the machine learning model) of a game (the machine learning model).

To better understand the concept of Shapley values, imagine a team sport or game[1], where each player strives to achieve the best possible outcome for the game, for instance, by maximizing a final score. Now, the same game is played multiple times, over and over again. Each time certain players are present—that is, they contribute to the game, and consequently to the final score—while other players are absent, not contributing. After every possible game has been played by iterating over all the possible combinations of present and absent players[2], the marginal contribution of each player to the overall final score of the game can be computed. The marginal contribution reveals to us the quality and quantity of the average contribution each player had to the outcome of the game. Needless to say, this way of computing Shapley values scales poorly with an increasing number of players, i.e., features. As it is often the case in machine learning models, the number of features can be very large. SHAP values are a computationally efficient way to accurately approximate Shapley values, which then tell us the contribution of the input features to the output of a machine learning model.

SHAP values reveal the impact of features on the output of a machine learning model. The measure of the impact for each feature on each of the outputs is quantifiable, which allows for a more nuanced understanding on how the machine learning model works. In addition, SHAP values ensure the theoretical properties of local accuracy (for a specific input, the contribution of each feature to the model's output is accurately reflected by SHAP values); missingness (when a feature is absent in the input to a machine learning model, that feature's SHAP value indicates no impact on the output); consistency (when a feature's impact on the output increases, its corresponding SHAP value will not decrease); and uniqueness (SHAP values provide specific and exclusive explanations).

One of the drawbacks in computing SHAP values is the handling of missing inputs (the "missing players"). Many machine learning models will not be able to handle partial inputs without retraining the model, which is problematic. To address this, Kernel SHAP (Lundberg and Lee, 2017) can be utilized. Kernel SHAP is a specific implementation for computing SHAP values for any type of machine learning model of the type described in (7). It handles the absence of inputs by simulating "missing data", which ensures that the input dimension of the machine learning model stays constant. This eliminates the need to retrain the model each time its input dimension changes. In the next section, we will discuss how the properties, and possibilities, of SHAP values can be leveraged to enhance an interactive multiobjective optimization method with explainability.

[1] For example, a rowing competition with eight sculls, the crew of eight wants to reach the goal as fast as possible.

[2] We assume the players to never get tired.

3.2 The R-XIMO method

In the R-XIMO method, SHAP values are computed by utilizing Kernel SHAP for a reference point based interactive multiobjective optimization method with a reference point as its input and an objective vector (the solution) as its output. The interactive method considered in Article I is simple—it consists of computing a solution based on a given reference point by utilizing different scalarization functions and solving its corresponding scalarization problem (2). In practice, the SHAP values computed for the described method are represented by a square matrix with dimensions $k \times k$:

$$\text{SHAP values} = \begin{pmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1k} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{k1} & \phi_{k2} & \cdots & \phi_{kk} \end{pmatrix}, \tag{11}$$

where each element ϕ_{ij}, with $i = 1, \ldots, k$, and $j = 1, \ldots, k$, represents the average effect of a given aspiration level on the components of a computed objective vector. That is, the effect the jth aspiration level in the input has had on the objective vector's ith component in the output is represented by ϕ_{ij}. The components of (11) are real-valued, where a positive values indicates a positive correlation between the jth aspiration level and the ith objective vector component, and likewise a negative value indicates a negative correlation. A SHAP value of zero would indicate that the aspiration level had no effect on the objective vector's component.

Based on the information contained in the SHAP values (11), it is possible to tell which aspiration levels affect which objective vector components, and how much on average—if at all. This information can be further leveraged to tell a decision maker how their preferences have affected the output of a reference point based multiobjective optimization method. This is the core idea in the R-XIMO method.

Once a decision maker has provided a reference point and has seen the objective vector computed based on the said point, the decision maker can express which component in the objective vector they would like to improve. The component the decision maker desires to improve is termed *a target*. The SHAP values can then reveal us which aspiration level in the reference point has had the most significant positive effect, or negative effect, on the target. This information can then be used to build a text-based explanation that tells the decision maker which aspiration levels have had the most significant effect on the target. As an example, consider the decision maker is interested in improving the first objective function to some multiobjective optimization problem with three objective functions, then, an explanation could be: "In the solution, the first objective function value was most improved by the aspiration level for the second objective function, and most impaired by the aspiration level for the third objective function."

Based on the explanation, we can then give a suggestion to the decision maker on how to modify the reference point to achieve a new solution with a more

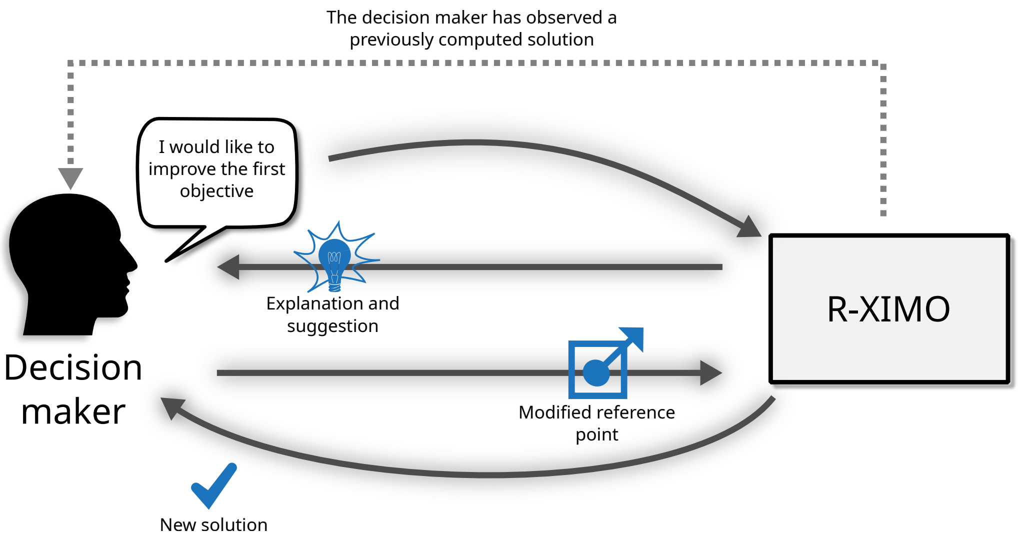

The decision maker has observed a previously computed solution
I would like to improve the first objective
Explanation and suggestion
R-XIMO
Decision maker
Modified reference point
New solution

FIGURE 2 A diagram illustrating the main idea of the R-XIMO method. The sequential steps in the R-XIMO method are illustrated, starting from the decision maker observing the objective vector corresponding to a previously computed solution. Then, the decision maker desires to improve a specific objective function in the observed solution. Next, the decision maker can utilize the explanation and suggestion generated by R-XIMO to modify the reference point. The modified reference point is finally utilized to compute a new solution with a desired improvement in the objective function the decision maker originally wished to improve.

desirable objective function value for the target. In the suggestion, the aspiration level corresponding to the target is suggested to be always improved, while the aspiration level with the most negative effect—or the least positive in the absence of a negative effect—is suggested to be deteriorated. This suggestion is based on the assumption that the objective vectors that have been computed are nondominated, which means that to improve the value of one of the objective functions in the objective vector, at least one other objective function value should worsen. This trade-off is all but trivial to the decision maker since it can be local and change depending in which part of the co-domain of the multiobjective optimization problem the trade-off is being made. In sum, without the suggestions produced by R-XIMO, the decision maker is left guessing which aspiration level they should be worsening in a reference point to achieve a more desirable result. An example of a suggestion derived from the explanation in the previous example, where a decision maker is interested in improving the first objective function, would be: "Try improving the reference point component of the first objective and impairing the component of the third objective." The main idea of the R-XIMO method has been depicted in Figure 2.

The correctness of the suggestions produced by R-XIMO was studied statistically in Article I. It was found that the suggestions will lead to desirable outcomes most of the time. Three different scalarization functions were considered in the tests: the achievement scalarizing function, the GUESS scalarization function, and the STOM scalarization function (c.f., Section 2.1.2). R-XIMO performed similarly with different scalarization functions, which is encouraging in terms of its generalizability to other scalarization functions. Moreover, the suggestions and explanations were also tested in a small scale case study with a human decision maker in a Finnish forestry management problem. The decision maker felt that the suggestions were helpful, but the explanations were too verbose. This led to the conclusions that the viewing of the explanations should be optional. Nonetheless, the suggestions remain explainable since they are directly derived from the explanations. Overall, the tests conducted with R-XIMO showed clear potential in the method and the underlying idea.

3.3 Discussion of the R-XIMO method

The R-XIMO method is an example on how existing methods for explainability can be utilized to produce meaningful explanations in the context of an interactive multiobjective optimization process. The main idea is to treat an interactive method as a black-box with inputs and outputs, and then explaining the connection of the inputs and the outputs. Here, we have leveraged the explanations to support a decision maker in providing preferences in a reference point based interactive multiobjective optimization method. While the idea was explored through a concrete example, SHAP values can be utilized to potentially achieve similar results in any kind of multiobjective optimization method that can be abstracted to resemble a black-box model with the form of (7). One intriguing example of this would be to explain the connection between the given classifications and computed solutions in the Synchronous NIMBUS method (Miettinen and Mäkelä, 2006). We could communicate to a decision maker which classifications have had the most significant impact on a computed solution, and how to modify the classifications to achieve a more desirable solution.

Of course, the research presented in Article I comes with limitations as well. To start, the suggestions are not always 100% accurate, and can lead to undesirable solutions. Moreover, the suggestions do not contain any information on how much an aspiration level in the reference point should change. Another limitation in the suggestions produced by R-XIMO, is that only the most positive (or least negative) SHAP value is considered, while any "second order effects" are ignored (that is, the SHAP value with the second most positive value, or second least negative value). On this note, R-XIMO is able to handle only requests from a decision maker to improve a single objective function in a solution. Generating suggestions for improving two or more objective functions simultaneously would require additional research. While the suggestions were welcomed by the human decision maker in the case study, the produced explanations were deemed to be too verbose. The case study also considered only a single decision maker, which makes the results of the case study anecdotal.

The limitations withstanding, we can still assert with confidence that R-XIMO proposes a novel and promising direction for exploring explainability in the context of multiobjective optimization. The idea should be applied to different types of multiobjective optimization methods. A particularly interesting line of research is the study of how to formulate explanations and suggestions based on SHAP values, and how to best communicate them accurately to a decision maker—e.g., by utilizing specialized visualizations. While a lot of information can be derived from SHAP values, we should be careful as not to overwhelm the decision maker with too much information.

4 INFRINGER

As discussed in Section 2.2.4, modeling the preferences of a decision maker can help us in finding preferred solutions that match the decision maker's preferences. However, the uniqueness of each decision maker, potentially changing preferences—due to learning, for instance—and the often small amount of preference information available, makes the prospect of learning a preference model challenging. Withstanding these difficulties, learning a preference model for a decision maker in interactive multiobjective optimization makes a lot of sense because the decision maker is expected to express their preferences multiple times during the optimization process. Moreover, the decision maker is also available to audit the preference model as it is learned.

When it comes to explainability, it is not far-fetched to argue that an explainable preference model comes with many benefits. For one, explainability could help the decision maker understand the model, which would not only help in any auditing of it, but also increase the trust the decision maker has in any solutions found by utilizing the model. Therefore, exploring the possibilities of modeling preferences with an explainable model is very intriguing.

Following these deliberations, in Article II, we have proposed a novel interactive multiobjective optimization method—the *INFRINGER*[1] *method*—with a learnable preference model. The method utilizes pair-wise comparisons of objective vectors to elicit preferences, and a *belief rule-based system*—utilized as a machine learning model—to model the preferences. Belief rule-based systems were chosen because of their apparent explainable properties. While the method can learn a preference model of a decision maker and help them reach desirable solutions, the explainable aspect of the model could not be exploited because of the complexity of the resulting model. Nonetheless, the work presented in this chapter has set the spark to ignite future research in this direction.

We begin our discussion by introducing belief rules and belief rule-based systems in Section 4.1. Then, we present the INFRINGER method in Section 4.2. We conclude this chapter in Section 4.3, where we discuss the implications of the INFRINGER method to the field of explainable multiobjective optimization.

[1] Derived from *Interactive inference of preferences using belief rules*.

4.1 Belief rules and belief rule-based systems

The belief rules and belief rule-based systems introduced in this section, and discussed for the remainder of this chapter, are ideas based on Bayesian probability, Dempster-Shafer theory of evidence, and fuzzy set theory (Yang et al., 2006). While intriguing topics on their own, discussing them further is way beyond the scope of this thesis. In the following, we will limit our discussion to what is needed to understand the work presented in Article II.

In this section, we first introduce belief rules in Section 4.1.1. Then, we proceed to discuss how belief rules can be utilized to devise belief rule-based systems in Section 4.1.2.

4.1.1 Rules, belief rules, and belief rule bases

A short primer on rules

Before introducing belief rules, a brief discussion on what is meant by *a rule* is needed. In our case, a rule consists of three parts: *a precedent*, *a condition*, and *a consequent*. For example, "If the traffic light is green, then cross the road." In the given example, the traffic light being green is the precedent, crossing the road is the consequent, and the *"if... then"*-part is the condition. The precedent is often one or more observations about the environment or a system, and the consequent is some action to be taken. As for the condition, we can see that it ties together the precedent and the consequent, or the observation and action.

However, in real life, we often make observations that do not simply boil down to be either true or false. For instance, we may posit the following question: "Is the traffic busy?" Unless we have a very specific definition of what "busy" means, we would naturally answer this question with something vague, for example: "not very" or "it is very busy." To cover all the possible magnitudes of *"busy"* would be a futile task. This same argument can also be made for the consequent: "If it is cold outside, then dress warmly."[2] What exactly does *"warmly"* mean? Does it consist of putting on a beanie? A beanie and mittens? Or perhaps it is enough to swap one's shorts into longer pants? Again, listing all possible variants of the respective condition of *"dressing"* in a certain way, would be pointless. To complicate matters even further, rules have often more than just a single precedent, e.g., "If the **traffic is busy** and the **weather is cold**, then leave for work now." Imagine listing all possible variants of the respective conditions of *"busy"*, *"cold"*, and *"later"*, and their possible variations, and we see that this is not a practical task at all.

[2] Of course, *"cold"* is also ambiguous here.

Belief rules

It is quite obvious that simple *if... then*-rules are not very well suited to handle real-world problems. Instead, we can use *belief rules*. Belief rules treat precedents and consequents as collections of discrete (referential) values. For precedents, these values are known as *antecedent values*, and for the consequents, *consequent values*. To tell how well the different antecedent values represent a precedent in a belief rule, and how well a consequent value represents a consequent, *belief distributions* are utilized. As an example of a belief distribution, in case of the amount of traffic, the antecedent values could consist of "*busy*" and "*light*", then, an observation of moderately busy traffic could be represented by the traffic being 70% "*busy*" and 30% "*light*", as an example. The same is true for the belief distributions for consequent values. It is important to note that the antecedent and consequent values are often chosen to be such that they convey a clear meaning. In belief rules, we can now represent the precedent and consequent with belief distributions over discrete, predetermined, and meaningful values. That is, we do not need to be able to list, or even know, all the possible values precedents and consequents may take. This is the main benefit of belief rules. For the rest of this thesis, the antecedent and consequent values are assumed to be numerical.

More formally, a single belief rule can be represented as the following collection of sets:

$$\textit{belief rule} = R = \{I, A, C, F\}, \tag{12}$$

where I is the set consisting of all the *inputs* (observations); A is the set of the antecedent values for each input in I; C is the set of consequent values; and F is the logical operator connecting each condition in the precedent of a belief rule, e.g., the conjunction *and* in the previous example: "If the traffic is busy **and** the weather is cold, then leave for work now." In this thesis, we consider only the *and* conjunction. A summary of a belief rule, and its components, is given in Figure 3.

Furthermore, each antecedent value, consequent value, and the belief rule itself (12), has an associated *rule weight* to them. Similarly, the antecedent values have each an *attribute weight* associated to them. Each consequent value has a *belief degree* associated to it, and each belief rule itself has its own weight. All the weights, the antecedent values, and the consequent belief degrees become important when we build a belief rule base, discussed next, and a belief rule-based system, discussed in the following section.

Belief rule bases

When multiple belief rules are considered, then we have a *belief rule base*. A belief rule base is a set containing belief rules (12):

$$\textit{belief rule base} = \{R^1, \ldots, R^l\} = \langle I^i, A^i, C^i, F^i | i = 1, \ldots, l \rangle, \tag{13}$$

where R^i, $i = 1\ldots, l$; are belief rules, l is the number of rules in the rule base, and $\langle \cdot \rangle$ represents a quadruple containing all the inputs, antecedent values, consequent values, and logical mapping found in the rule base. However, a belief rule base in

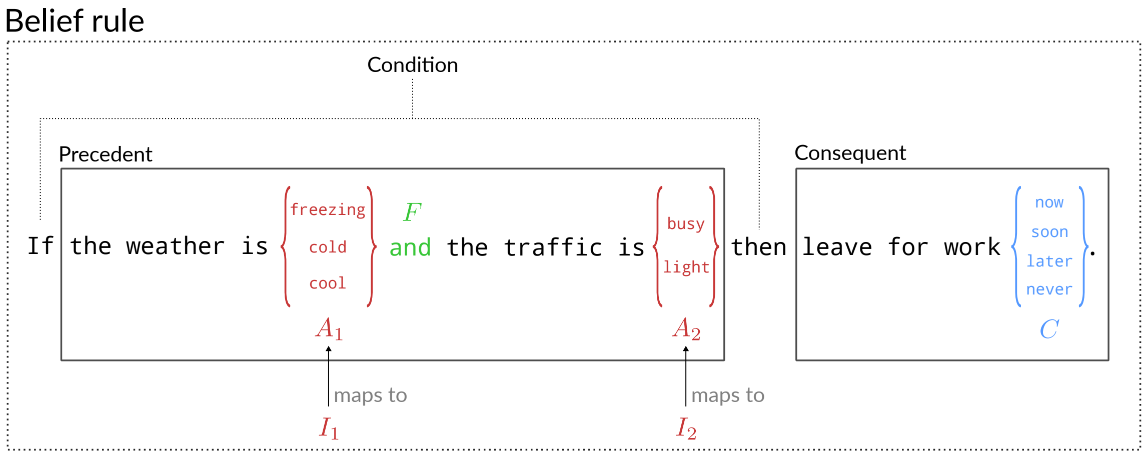


FIGURE 3 An example of a belief rule (12) and its components. $I = \{I_1, I_2\}$ is the set of inputs (observations) of the rule, $A = \{A_1, A_2\}$ is the set of antecedent values, C are the consequent values, and F is the logical mapping between the conditions in the precedent of the belief rule. The antecedent and consequent values are pictured. The precedent, consequent, and condition of the belief rule are highlighted as well.

itself is not that useful. The base is just a collection of rules. In the next section, we will discuss how belief rule bases can be utilized to build belief rule-based systems.

4.1.2 Belief rule-based systems

A belief rule-based system is a type of expert system or decision support system that combines elements of both rule-based systems and belief rules to make decisions or draw conclusions. Belief rule-based systems are adequate for modeling complex nonlinear systems, making effective use of semi-quantitative information, including qualitative knowledge, incorporating uncertainty, and utilizing expert knowledge and small sample data to obtain accurate results (Zhou et al., 2019). Moreover, belief-rule based systems can be interpretable (Cao et al., 2020; You et al., 2022). The belief rule-based system discussed in this section, and the remainder of this chapter, is largely based on the RIMER methodology (Yang et al., 2006). Likewise, the inference and the training of the belief rule-based system is partly based on the work presented in (Chen et al., 2011).

The belief rule-based system in Article II has been divided into five components. These components are: *a belief rule base*, *input transformation*, *an inference mechanism*, *calculation of activation weights*, and *output transformation*. We will discuss each of these components, and how they combine into a belief rule-based system, in the rest of this section. A more concrete example of a belief-rule based system will be discussed in Section 4.2, where we discuss the INFRINGER method.

As already mentioned, a belief rule base is a collection of belief rules (13). The various parameters of a rule base (antecedent and consequent values, attribute weights, rule weights, logical mappings between consequent, and belief distributions over consequent values) can all be either set as learnable parameters, or to constant values. The constant values can be based on expert knowledge—e.g., from a decision maker—or they may be determined otherwise. In Article II, the antecedent values, the belief distributions over consequent values, and the attribute and rule weights, have been left as learnable parameters, while the consequent values and the logical mappings are treated as constants.

Inputs to each belief rule in a rule base are expressed as belief distributions over antecedent values. Computing the belief distributions is done by the input transformation component. The inputs are transformed such that belief distributions are calculated for each rule in the rule base of the belief rule-based system. For each input, the values comprising the input's belief distribution are assumed to sum to unity[3]. As a simple example, consider the antecedent values $\{4, 8\}$ and the belief distribution $\{0.75, 0.25\}$. These would then represent an input of 5, i.e, $0.75 \times 4 + 0.25 \times 8 = 5$. The belief distributions over the consequent values are expressed with similar belief distributions as well. It is worth noting that belief distributions are not necessarily unambiguous.

The inference mechanism calculates the *combined belief degrees* based on the individual belief degrees calculated over the consequent values in the belief rule base. We have assumed that each rule consists of exactly the same (and constant) consequent values. To calculate the combined belief degrees, the inference mechanism makes use of so-called *rule activation weights*, which, in turn, are calculated (by the aptly named component) based on the belief degrees over the antecedent values, attribute weights, and rule weights.

To learn the parameters of a belief-rule based system, a form of supervised learning (c.f., Section 2.3.1) can be utilized. Given known training pairs consisting of inputs (precedents) and outputs (consequents)—that is, known observation-action pairs—a belief rule-based system can be trained to model different systems. In other words, this means finding the optimal values for the parameters of the system by solving an optimization problem similar to (9), so that it achieves a desirable outcome. Concrete examples of an input and output will be discussed in the next section.

Lastly, there is the output transformation component. The output transformation aggregates the combined belief degrees computed by the inference mechanism into a coherent output of the rule base. A *utility function*[4] is utilized to transform each consequent into a form that may be combined. This is useful when the consequent values are expressed in qualitative terms, e.g., *"now"* or *"soon"* (c.f., Figure 3). Then, the combined belief degrees, and transformed consequent values, can be aggregated by, for example, a weighted sum, where each transformed consequent value is weighted by its corresponding combined belief degree. This allows us to express the output of a belief rule-based system as a numerical value. And in turn, allows us to represent, understand, and utilize, a belief rule-based system as a machine learning model (7). A summary of the belief rule-based

[3] This assumption can be relaxed such that when belief distribution values do not sum to unity, the deficit conveys a degree of uncertainty or "unknown knowledge" of the precedent.

[4] The naming is unfortunate here and should not be confused with the function presented in (6).

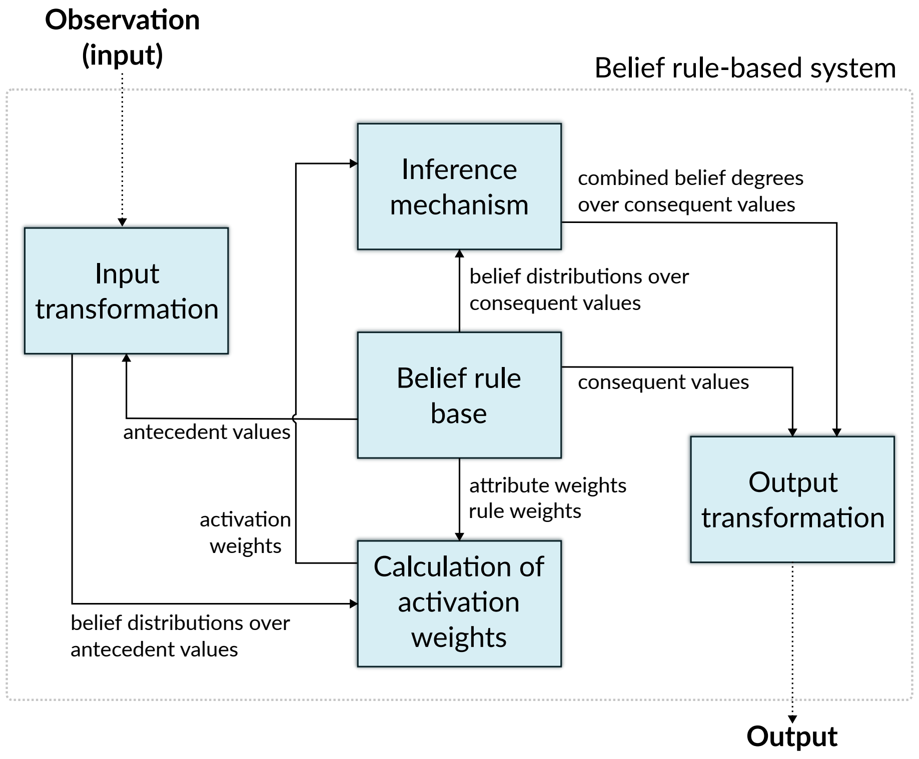

Observation (input)
Belief rule-based system
Inference mechanism
combined belief degrees over consequent values
Input transformation
belief distributions over consequent values
Belief rule base
consequent values
antecedent values
attribute weights rule weights
Output transformation
activation weights
Calculation of activation weights
belief distributions over antecedent values
Output

FIGURE 4 The belief rule-based system considered in Article II. The system's components are represented by boxes. The solid arrows indicate how the different parameters are related to each component, they do not represent any kind of directed flow. The dashed arrows represent the input and output of the whole belief rule-based system. The component the solid arrows originate from are the sources of the parameters, while the component the arrows point to, are dependent on the parameters. The input to the system is a set of observations about the precedents found in the belief rule base. The output is aggregated from the consequent values of the rule base. Notice that the belief rule base component at the center only provides parameter values to the overall belief rule-based system, its parameter values are the result of training the model.

system discussed in this section is given in Figure 4.

4.2 The INFRINGER method

The INFRINGER method is an interactive multiobjective optimization method able to learn a preference model of a decision maker. The input of the method is a representation of the Pareto front of a multiobjective optimization problem (1), and the output is the value function (6) of the decision maker modeled by a belief rule-based system, and one or more of the best solutions on the Pareto front, according to the learned preference model. In addition to Article II, the INFRINGER method has also been discussed in the master's thesis of the author (Misitano, 2020).

At the start of the INFRINGER method, there is no value function. To remedy this, an initial value function is modeled based on a reference point given by the decision maker at the start of the method. In the case the decision maker does not wish to, or cannot, provide a reference point, then, a point with each of their components being the midpoint of the respective components in the ideal and nadir point, is chosen as the reference point to model an initial value function. After initializing the value function based on the reference point, the decision maker then compares objective vector pairs by indicating which of the two vectors they prefer, or if they deem the vectors equally preferable. When making pair-wise comparisons, the decision maker is presented with multiple sets of pairs of objective vectors. The objective vectors utilized to form the pair-wise comparisons are taken from a non-dominated solution set representing an approximation of the Pareto optimal front of the problem being solved.

The vectors chosen for each set of pair-wise comparisons are selected such that the relative difference of the value function value between both objective vectors in each pair exhibit a varying degree of variance, i.e., in some pairs the value function values for both vectors are close to each other, and in other pairs they are farther apart. In other words, the decision maker is prompted to compare pairs of objective vectors, where some of the pairs are clearly distinct in terms of their value function value, and some are less. This way, we posit to be able to elicit both coarser and finer details about the decision maker's preference model.

Based on the pair-wise comparisons conducted by the decision maker, the belief rule-based system of the INFRINGER method is then trained to model the decision maker's preferences as a value function (6). The higher the value function's value, the better the assessed objective vector is regarded to be according to the preferences of the decision maker. The modeled value function is assumed to be monotonically increasing, i.e., when assessing an objective vector, the value function's value increases if any single objective vector component improves without changing the others. This is based on the premise that the decision maker will always prefer less to more, when an objective function is being minimized.

After modeling the value function, the decision maker is shown a visualized representation of the Pareto front, where each objective vectors has been ranked on a continuous scale according to the value function modeled by the belief rule-based system. The "best" objective vector is highlighted as well. This gives the decision maker an impression of their preference model; and whether they agree with the ranking or not, indicates how well their value function has been modeled. Afterward, the decision maker can choose to perform a new set of pair-wise comparisons of objective vectors to further tune the modeled value function, or they can choose to stop. If they choose to stop, the objective vectors with the best value function values in the approximate Pareto optimal front of the problem being solved are shown to the decision maker. This number of objective vectors shown can be specified by the decision maker. From these, the decision maker can then select a final solution, or solutions. Even if the preference model already tells us the best objective vector according to the modeled preferences, we still offer the decision maker the possibility to choose the final solution from the top-ranked vectors in case the decision maker's preferences have slightly changed, or they wish to select multiple solutions. The INFRINGER method can also be set to stop once a desired loss function value (8), e.g., a threshold value, is reached during training of the belief rule-based system.

The precedents in the rules of the belief rule-based system in the INFRINGER method are objective function values, while the consequents are value function values. The number of precedents in each rule is equal to the number of objective functions in the multiobjective optimization problem being solved. As for the consequents, there is only one for each rule. To give an example, consider the belief rule (c.f., Figure 3) for inspecting an objective vector (the input) with two components: "If the value of the first component is $\{0/50/100\}$ and the value of the second component is $\{120/200/225\}$, then the respective objective vector's value is $\{0/0.5/1\}$." In the example, we have two objective functions in the precedent each with three antecedent values. In the consequent of the rule, the value function being modeled is represented by three consequent values. In the belief rule-based system of the method, there are multiple belief rules similar to the example given, but with different (and learnable) antecedent values, rule weights, attribute weights, and belief distributions over the consequent values.

For training the parameters of the belief rule-based system of the INFRINGER method, a loss function (8) is minimized by solving a problem equivalent to (9). The formulation of the loss function is based on three assumptions. First, based on the pair-wise comparisons conducted by a decision maker, we can infer the relative value function values between the compared objective vectors. In other words, if we know that the objective vector $\mathbf{z}_1$ is preferred to $\mathbf{z}_2$, then the value function being modeled should reflect this so that $\mathbf{z}_1$ is evaluated to a greater (better) value by the modeled value function than $\mathbf{z}_2$. Second, the ideal and nadir points of the multiobjective optimization problem being solved should be evaluated such that the modeled value function results in its highest possible value (best) by evaluating the ideal point, and its lowest possible value (worst) by evaluating the nadir point. And third, the modeled value function should be monotonically increasing, as already described. Therefore, we do not have explicit data for outputs (i.e., known objective vector and value function value pairs) when training the belief rule-based system[5], but rather, we have constraints based on the three aforementioned assumptions. While there are no explicit training pairs, we are able to model a loss function nonetheless based on the pair-wise comparisons, making the training of the belief rule-based system of the INFRINGER method a form of supervised learning. An overview of the INFRINGER method is given in Figure 5.

The INFRINGER method was tested with a human decision maker in a case study considering a Finnish forest management multiobjective optimization problem in Article II. Overall, the decision maker thought their preferences were well-represented after a couple of iterations of the method. The final best objective vectors shown to the decision maker made sense from their perspective. During the interactive process, the decision maker felt that the ranking of the objective

[5] A variant where a decision maker scores objective vectors (i.e., expresses the value function values for each vector directly), instead of conducting pair-wise comparisons, was proposed in (Misitano, 2020). However, with such data, it was found that the belief rule-based system was very challenging to train successfully.

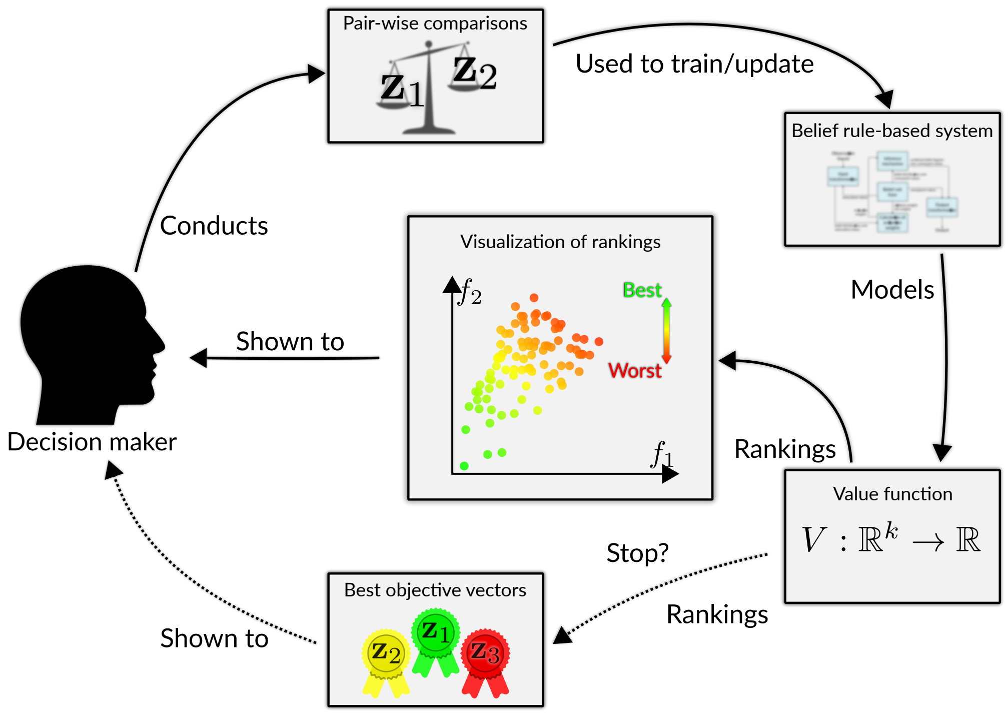


FIGURE 5 An overview of the INFRINGER method. A decision maker provides pair-wise rankings for objective vectors. These ranking are then used to train and update a belief rule-based system, which in turn models the preferences of the decision maker via a value function. The value function is then used to rank the Pareto optimal objective vectors of the multiobjective optimization problem being solved. These rankings are visualized and shown to the decision maker. The decision maker can then choose to either continue to provide new pair-wise rankings, which will lead to an updated model of their value function. Alternatively, the decision maker may choose to stop (indicated by the dashed arrows), in which case the best objective vectors, according to the current value function, are shown to the decision maker, and then the method stops.

vectors shown to them was gradually closing to what they were looking for. The decision maker also noted that their preferences had changed during the course of the method (probably due to the decision maker learning about their preferences), and that the modeled value function was able to, presumably, keep up with these changes. It was also noted by the decision maker that in some of the pair-wise comparisons, they would have preferred to express *"no preference"* between two objective vectors, but due to the lack of this option, they had expressed the two vectors to be equal instead.

4.3 Discussion of the INFRINGER method

The INFRINGER method discussed in this chapter, and introduced in Article II, is able to utilize belief rules and belief rule-based systems to learn a preference model of a decision maker, eliciting it through pair-wise comparisons of Pareto optimal objective vectors. It was shown in a case study that the method has the potential to successfully learn a decision maker's preference model, and also to support them in finding solutions to a multiobjective optimization problem. Because the preference model is retrained between pair-wise comparisons, comparing consequent models can also reveal details about how the decision maker has learned their preferences as well. Furthermore, the produced preference model can also be utilized in other multiobjective optimization methods for solving the same problem, even in absence of the decision maker.

However, the INFRINGER method comes with limitations. Again, as was the case with R-XIMO (c.f., Chapter 3), the case study conducted involved only a single decision maker, which does not allow the results to be generalized. In addition, the INFRINGER method comes with some computational overhead. The training of its belief rule-based system to model the value function can take a considerable amount of time, which, unfortunately, can also vary a lot depending on the results of the pair-wise comparisons. The considered model of preferences, namely, a value function, comes also with known limitations, as discussed in Section 2.2.4. Furthermore, as was mentioned by the decision maker during the case study, the pair-wise comparisons were lacking, as in, there was no option to express indifference between two objective vectors, and the decision maker was forced to choose equality in cases where indifference would have been more appropriate. Lastly, the INFRINGER method requires a precomputed approximation of the Pareto front of the problem being solved, which must be computed by other means, e.g., by an evolutionary method (c.f., Section 2.1.3). Thereby, the solutions found by the INFRINGER method are also approximations of the true Pareto optimal solutions.

Considering the title of this thesis, the reader might worry that we still have not discussed explainability in the context of the INFRINGER method. Given that the modeled value function is based on belief rules, which we saw to be quite understandable, one could assume that these rules would be an easy way to also build explanations explaining the value function to the decision maker. However, the reality is not that simple. There are 27 belief rules in the rule base of the INFRINGER method presented in Article II, which in itself is already a formidable number of rules to a human to interpret. We must also remember that each belief rule comes with a myriad of parameters, complicating the situation even more. Building any comprehensible explanations based on the rules, and their parameters, will require additional research—not just to find a way to explain the belief rule-based system, but to also understand how to reduce the number of parameters and rules without compromising the accuracy and interpretability of the results. Perhaps, by incorporating more expert knowledge from the decision maker in the rule base, we could begin approaching this goal.

The presented work is also a testament to the notion that a machine learning model's explainability is not a monolithic concept (c.f., Section 2.3.2). In other words, utilizing a method that, on the surface, seems explainable, does not guarantee its explainability if the method ends up being too complex. To the best of our knowledge, there are no ways to extract explanations from belief rule-based systems that could help a decision maker in the context of interactive multiobjective optimization and preference modeling. However, this is not to say that belief rule-based systems would not have been used as explainable models in other applications. Quite the contrary in fact, as is evident in the works by Sachan et al. (2020) and Yang et al. (2022), for instance. These works, and others, could inspire future works in interactive multiobjective optimization where the explainable aspects of belief rule-based systems are better leveraged.

While in the INFRINGER method we were not able to produce explanations, we have nonetheless set the stage for future research on utilizing belief-rule based systems as a potentially explainable machine learning model to enhance decision-making in interactive multiobjective optimization. The shortcomings of the INFRINGER method should not discourage the reader regarding the potential of rule-based explanations. As we will see in the following chapter, rule-based approaches can be very promising for enhancing interactive multiobjective optimization with explainability.

5 XLEMOO

When we discussed evolutionary methods in Section 2.1.3, we noted that one of their advantages is their ability to generate multiple, approximate Pareto optimal solutions. However, most of the time only a few of these end up being useful to a decision maker. Does this mean that the neglected solutions are useless? Not necessarily. For instance, machine learning could be utilized to gain additional insights about the solutions generated, namely, insights about the connection between decision and objective vectors. Moreover, we do not need to utilize machine learning in isolation from the evolutionary method. In fact, the two can be combined into its own kind of method.

The work presented in Article III is about exploring the combination of evolutionary methods and machine learning models. The work sets the stage for a new type of interactive evolutionary methods: *learnable and explainable evolutionary multiobjective optimization methods*. The concept is, yet again, studied in an interactive context. In addition, a software framework is provided, which allows others to readily apply and explore learnable and explainable evolutionary multiobjective optimization methods in their own works. The implementation of this framework has been largely enabled by DESDEO, which will be discussed in the following chapter, Chapter 6.

Before discussing the method proposed in Article III, we briefly discuss learnable evolutionary models in Section 5.1, the basis of the work addressed in this chapter. Then, in Section 5.2, we discuss the interactive method proposed in Article III. Finally, we conclude this chapter in Section 5.3, where the implications, limitations, and contributions of Article III are weighted.

5.1 Learnable evolutionary models

Learnable evolutionary models (Michalski, 2000) combine both evolutionary methods and machine learning for solving single-objective optimization problems. The idea is to evolve a population of solutions (c.f., Section 2.1.3), and periodically employ a machine learning model to learn a so-called *hypothesis* based on the best and worst performing solutions found in the population. The best and worst performing solutions are found by dividing the population into *high-performing* and *low-performing* groups—H- and L-groups, respectively. The division into these two groups happens, e.g., by selecting solution percentiles that have the highest (H-group) and lowest (L-group) fitness values. Then, a machine learning model is trained to classify solutions in a population into these two groups. This learned dichotomy by the machine learning model, the hypothesis, is then utilized to generate new high- or low-performing solutions. Lastly, these newly generated solutions are combined with the existing population, which can then be evolved further with an evolutionary method.

In their original work, Michalski (2000) demonstrated that for some optimization problems, the combination of evolutionary methods with machine learning showed a clear boost in the search process for an optimal solution, characterized by sudden improvements[1] in the overall fitness of the population—typically manifesting after machine learning had been employed to generate and introduce new solutions into the population. This apparent boost to the search for an optimal solution makes learnable evolutionary models compelling.

Learnable evolutionary models are characterized by two distinct operating modes: a *Darwinian mode*[2] (evolutionary method) and a *learning mode* (machine learning). The learning mode is interesting because it is used to generate new solutions, which is also referred to as *hypothesis instantiation*. This is a type of descriptive machine learning (c.f., Section 2.3.1), i.e., we do not try to predict new data from observations, but rather, try to learn a description of the data, which can then be used to generate new data—a form of *generative machine learning*. Consequently, when a machine learning model is selected for the learning mode, it should be readily possible to utilize the model to generate new solutions based on the learned classification. That is to say that not all machine learning models are fit for being applied in a learning mode.

As an example of a suitable machine learning model, in Michalski's work (2000), *AQ learning* was employed in a learning mode. AQ learning is a form of inductive learning based on attributional calculus (Michalski, 1983). AQ learning generates rule-sets of attributional rules, which consist of a condition and a decision. The condition is a conjunction of attributional conditions, and the decision is an elementary attributional condition. If the condition is true, then the decision holds. Despite the different terminology, if we return to the example of a rule given in Section 4.1 ("If the traffic is busy and the weather is cold, then leave for work now."), then the condition in AQ learning is synonymous to the precedent in the example, and likewise, the decision is synonymous to the consequent. Continuing with the previous example, *"traffic is busy" and "weather is cold"* are then a conjunc-

[1] Described as *quantum jumps* by Michalski.

[2] In (Michalski, 2000), both the Darwinian and Lamarckian theories of evolution were contrasted, and the proposed model employed specifically a Darwinian type of evolution in the learning mode. An in-depth discussion and comparison of these two theories are, unfortunately, outside the scope of this thesis.

tion of attributional conditions (conjugated with the *"and"*), and *"leave for work now"* is an elementary attributional condition. Unsurprisingly, rule-based machine learning models are a prime candidate to be utilized in a learnable evolutionary model's learning mode, because they are easy to interpret, and therefore, readily utilized to generate new solutions based on a learned hypothesis.

Apart from single-objective optimization problems, the application of learnable evolutionary models to multiobjective optimization has also been studied in the literature, but to a small extent only. For example, in the works of Jourdan et al. (2005), Moradi et al. (2016; 2019; 2020), and Niu et al. (2021), learnable evolutionary models have been employed to solve multiobjective optimization problems from various application fields. A common finding in these works has been that the inclusion of a learning mode to boost an evolutionary search process has had an overall positive effect on the search of optimal solutions.

In Article III, learnable evolutionary models have been applied to create a new interactive multiobjective optimization method. In the method's learning mode, an interpretable rule-based machine learning model, able to generate similar rules as in AQ learning, is used. The work presented in Article III is first of its kind in exploring learnable evolutionary models in the context of interactive multiobjective optimization and explainability. We will discuss the interactive method and its explainable qualities next.

5.2 The XLEMOO method

The interactive multiobjective optimization method proposed in Article III, the *XLEMOO*[3] *method*, is an indicator-based evolutionary multiobjective optimization method (c.f., Section 2.1.3) combined with a rule-based interpretable machine learning model: *skope-rules* (Goix et al., 2020). Skope-rules were chosen over AQ learning because of the readily available implementation of the former. Skope-rules are adept for learning interpretable and precise rules for *scoping* classes in classification problems. They offer a trade-off between the interpretability of decision trees and the modeling power of random forests. As an example of a rule generated by a skope-rules model trained for bi-classification[4], consider the following: "If $a_1 > 0.8$ and $a_2 < 0.3$ then predict `CLASS 1`.", where a_1 and a_2 are attributes of the model modeled by skope-rules; and `CLASS 1` is one of the two classes the rule may classify an input characterized by a_1 and a_2 into. When the two conditions in the precedent of the rule are true, then the input is classified to belong to `CLASS 1`. Otherwise, the input is classified into the other class, e.g., `CLASS 2`. A skope-rule model may generate more than one such rule with varying numbers of attributes in its precedent. Moreover, each rule is characterized by a *precision*, which describes how many of the observed training inputs are described by the rule. E.g., a precision of 0.45 means that the rule describes successfully 45%

[3] Explainable and learnable multiobjective optimization
[4] A machine learning classification problem where there are two possible categorical outcomes.

of the samples in the training data. In other words, the higher the precision, the more descriptive the rule is for describing the training data.

In the XLEMOO method, the skope-rules model is trained to classify decision vectors in a population of solutions to a multiobjective optimization problem into H- and L-groups. Whether a solution belongs to the H- or L-group, is determined by the decision vector's fitness. The fitness is calculated based on a scalarization function (c.f., Section 2.1.2) and a reference point provided by a decision maker. According to the scalarization function, the closer a decision vector is to the reference point, the better its fitness, and consequently, the more likely it belongs to the H-group. Naturally, the farther the decision vector is from the reference point, the more likely it belongs to the L-group. The division of vectors into the two groups is determined based on a top and bottom percentile, e.g., 10% of decision vectors with the best fitness form the H-group, and likewise the 10% of decision vectors with the worst fitness form the L-group. It is important for the two groups to be distinct enough so that a clear-cut classification can be learned, which is also why we do not simply split *all* the decision vectors evenly into the two groups. When instantiating new solutions based on the learned classification, solutions belonging to the H-group are generated and introduced to the population. To summarize, the skope-rules in the XLEMOO method are used to classify decision vectors into two distinct classes based on whether the decision vectors are, in relative terms, closer or farther away from the preferences of a decision maker.

The Darwinian mode of the XLEMOO method is an indicator-based evolutionary multiobjective optimization method inspired by the works of Zitzler et al. (2004) and Thiele et al. (2009). In the XLEMOO method, the same scalarization function from the learning mode is utilized as the indicator of the Darwinian mode as well. Thus, in both modes, the XLEMOO method strives to find solutions that are close to a reference point provided by a decision maker.

An iteration of the XLEMOO method consists of applying its Darwinian mode to the population continually multiple times, and then applying its learning mode once. Each time a decision maker provides a new reference point, the XLEMOO method is iterated a few times before stopping. Once stopped, the output of the method is a population of solutions that are close to the reference point in the objective space, and a list of skope-rules. These rules describe the decision variables of the solutions with the best fitness. Each rule can describe one or more of the decision variables. The rules are then combined to describe the set of solutions that abide to all the rules. If there are no rules describing a decision variable, then that variable's description is extracted from the final population of the method as the variable's minimum and maximum values. Then, a description consisting of the minimum and maximum value of each decision variable in the decision vectors close to the reference point provided by a decision maker, is formulated and acts as an explanation.

The explanations provided by the XLEMOO method can provide valuable information to a decision maker about the connection between the decision space and the objective space near their preferences. This information can be valuable for engineers and technically adept decision makers to whom the decision variable values convey important information, e.g., information about implementing a solution, which can affect decision-making. An analyst, a person supporting the decision maker by modeling the multiobjective optimization problem being solved, can also benefit from the explanations. To the analyst, the explanations can provide important information about the modeling of the problem, which can help in, e.g., troubleshooting and debugging the model, if unexpected relationships are found among the decision and objective vectors. A showcase on how the explanations can support a fictional decision maker[5], with an engineering background, to solve a multiobjective optimization problem optimizing the crash safety of a car has been given in Article III.

The search performance of the XLEMOO method for solutions near a reference point was also studied in Article III. Two parameters unique to the learnable evolutionary part of the method were varied. Namely, the *switching frequency* and the *H/L-split*. The switching frequency determines how many times a Darwinian mode is applied to a population before switching to a learning mode, while the H/L-split determines the percentiles when dividing solutions of a population into the high-performing and low-performing groups. It was shown that the performance of the method (measured based on four different performance metrics) was best with the lower switching frequencies, while the effect of the H/L-split on the search performance was minor at best. The test results were promising and indicated that the inclusion of a learning mode in the indicator-based method presented in the work was beneficial to the overall search performance of the method.

To summarize, the XLEMOO method can find solutions close to a reference point and is able to explain the characteristics of the decision vectors that are closest to the reference point. The inclusion of a learning mode to a simple indicator-based evolutionary method was shown to boost the method's search performance. By iterating the method multiple times with a reference point provided by a decision maker, the method can act as an interactive multiobjective optimization method as well. The interactive solution process, and the XLEMOO method itself, have been summarized in Figure 6.

In addition, a software framework, the *XLEMOO framework* (Misitano, 2023), was provided in Article III for implementing learnable evolutionary multiobjective optimization methods with interpretable machine learning. The framework is implemented in Python, open source with an MIT license[6], fully documented, and first of its kind. The framework allows researchers and practitioners to experiment with learnable evolutionary multiobjective optimization methods and apply them to their own needs.

[5] That is, the author of the article acting as a decision maker.
[6] https://spdx.org/licenses/MIT.html, accessed: May 9, 2024

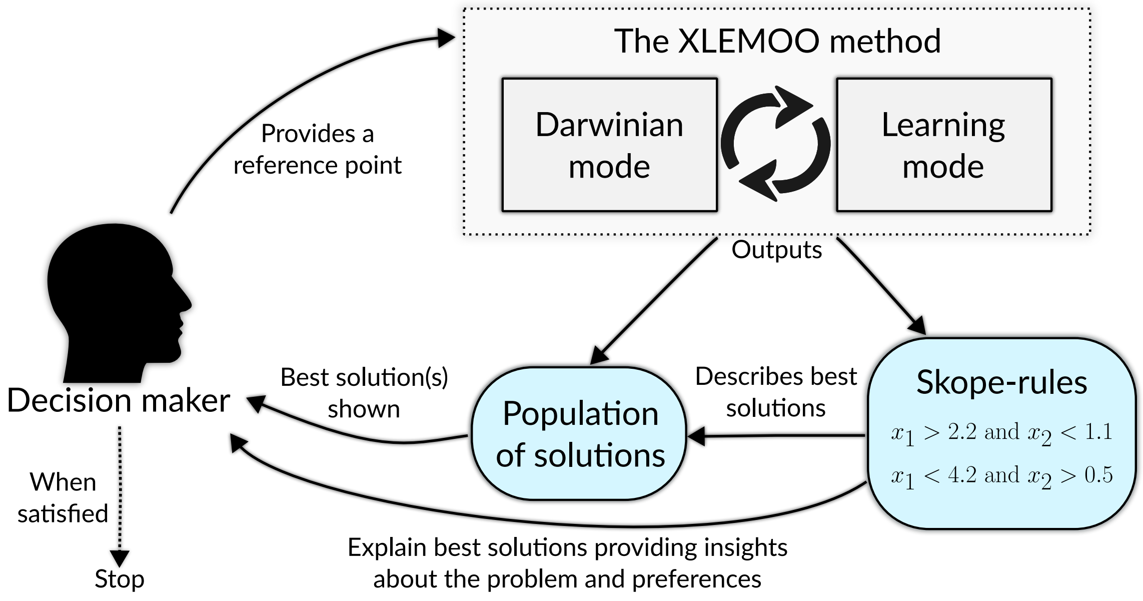


FIGURE 6 Illustration summarizing the XLEMOO method. The decision maker can provide a reference point, which is utilized in the XLEMOO method to compute a population of solutions to a multiobjective optimization problem being solved. The method outputs the population and skope-rules describing the high-performing individuals of the population. One or more of the best solution(s) are then shown to the decision maker (in the objective space) along rules describing all the high-performing solutions. The rules function as an explanation explaining the characteristics of decision variables found in the high-performing solutions. The decision variables x_1 and x_2 are given as an example. The decision maker can then inspect the explanations and best solutions, after which they may provide a new reference point, or stop if they are satisfied with the found solution(s).

5.3 Discussion of the XLEMOO method

The combination of a learnable evolutionary model with an indicator-based evolutionary multiobjective optimization method shows promise in terms of the search performance for optimal solutions. This observation is in line with previous studies (Jourdan et al., 2005; Moradi and Mirzaei, 2016; Moradi, 2019, 2020; Niu et al., 2021). In addition, by employing an interpretable machine learning method in the learnable evolutionary model, explanations can also be generated that can support decision makers. The XLEMOO method presented in this chapter has successfully combined these ideas to enhance the decision-support capabilities of an interactive evolutionary multiobjective optimization method.

However, the work in Article III has also limitations. Firstly, no real decision maker was included in the showcase demonstrating the usefulness of the explanations generated by the XLEMOO method. While there are works with human decision makers in which it is shown that decision variable values can be useful from a decision-support perspective, e.g., as was found in (Kania et al., 2022), the actual usefulness remains subject to interpretation and requires further studies to properly address. Secondly, the XLEMOO method was tested with only one type of evolutionary method in its Darwinian mode, and one type of machine learning model in its learning mode. Again, a study that further delves into the use of different evolutionary multiobjective optimization methods combined with different machine learning methods, is needed. While rule-based explanations are promising, other types of explanations should be explored in the future as well. This is especially true for multiobjective optimization problems with a high number of decision variables. The problems considered in Article III had only a few variables.

The XLEMOO method, and the XLEMOO framework, present a preliminary work in paving the road towards a new sub-field in evolutionary multiobjective optimization. Learnable and explainable evolutionary multiobjective optimization methods leverage one of the great strengths of evolutionary methods: their ability to generate vast amounts of non-dominated solutions to a multiobjective optimization problem. With sizeable populations, machine learning becomes a viable tool in describing the population. These descriptions can then be readily formulated into explanations. Naturally, interpretable machine learning models make this task almost trivial. In addition to enabling explanations, the inclusion of a learning mode can work potentially as a niching operator[7] in evolutionary methods. The XLEMOO software framework provided can make it easier for other researchers to embark on novel studies, encouraging the exploration of the potential of learnable evolutionary models in multiobjective optimization in the future. In light of its encouraging results, and despite its limitations, the work presented in Article III sets a promising avenue for many types of future works in exploring the emerging field of explainable and learnable evolutionary multiobjective optimization.

[7] An auxiliary operator in evolutionary methods utilized to encourage diversity in the population during its evolution, and thus improving the search for optimal solutions.

6 DESDEO

In this chapter, we will discuss the open source software framework DESDEO introduced in Article IV. DESDEO is specifically crafted to support the implementation and research of interactive multiobjective optimization methods—both existing and novel. Consequently, it has significantly supported the integration of explainability with interactive methods. DESDEO has been the main enabling force in researching, developing, and implementing the three interactive methods enhanced with explainability presented in Chapters 3, 4, and 5. Moreover, the author of this thesis is also one of the main developers of DESDEO. For these reasons, DESDEO has been included in this thesis as well.

We begin with Section 6.1, where we briefly introduce DESDEO. Then, in Section 6.2, we elaborate in more detail on the role DESDEO has had in the other works included in this thesis. Lastly, in Section 6.3, we shortly mention DESDEO's impact outside the scope of this thesis to outline its broader impact, discuss its shortcomings, and conclude by discussing some of the future developments of DESDEO to address its current limitations.

6.1 The DESDEO software framework

DESDEO is an open source software framework for interactive multiobjective optimization implemented in Python, as described in Article IV. The framework has been developed in a modular fashion to facilitate further contributions and utilization of specific parts of the framework and interactive methods. Apart from DESDEO, and according to the best of our knowledge, there are no other frameworks that provide implementations of both scalarization-based and evolutionary interactive multiobjective optimization methods.

DESDEO is composed of four *core packages*, each with a distinct role and fulfilling specific needs in interactive multiobjective optimization. The first of the packages is `desdeo-problem`. As the name suggests, the `desdeo-problem` package contains tools and utilities to model multiobjective optimization problems.

In addition to modeling problems with analytical formulations, there is support for data-driven and surrogate-based problems as well. Moreover, `desdeo-problem` provides also implementations of ready-defined problems, such as various test problems, e.g., the real-world engineering test problems (Tanabe and Ishibuchi, 2020).

The second package, `desdeo-tools`, contains utilities and functionalities that are needed across different types of interactive methods. For instance, scalarization functions, and interfaces to single-objective optimizers are found in this package. In addition, `desdeo-tools` provides means to enable interaction between a decision maker and an interactive method, which are handled through *requests* and *responses*. These are custom structures found in DESDEO that have been crafted to meet the various needs of different interactive methods.

Interactive methods are implemented in the `desdeo-emo` and `desdeo-mcdm` packages; evolutionary methods in the former, and scalarization-based methods in the latter. Some of the interactive methods found in `desdeo-emo` include the interactive variants of RVEA (Hakanen et al., 2016) and NSGA-III (Lárraga et al., 2023); and PBEA (Thiele et al., 2009), for instance. Other interactive evolutionary methods are implemented as well. Moreover, the `desdeo-emo` package contains the non-interactive versions of many of the interactive variants, such as the previously mentioned RVEA (Cheng et al., 2016) and NSGA-III (Deb and Jain, 2013) methods. Apart from evolutionary methods, `desdeo-emo` contains implementations of tools to handle populations and their evolution, namely evolutionary operators suitable for multiobjective optimization. In other words, `desdeo-emo` provides many tools to implement further evolutionary methods.

Lastly, the `desdeo-mcdm` package contains implementations of methods, such as synchronous NIMBUS (Miettinen and Mäkelä, 2006), the reference point method (Wierzbicki, 1982), and various methods belonging to the NAUTILUS family of methods (Miettinen and Ruiz, 2016). Besides the listed interactive methods, other methods are also implemented. In addition, the `desdeo-mcdm` package contains utilities that are sometimes important in scalarization-based methods, such as an implementation of the payoff table method (Miettinen, 1999).

Together, the four packages make up DESDEO. Being structured as such, its users can readily find the necessary tools to implement interactive methods to solve multiobjective optimization problems. However, DESDEO is far from perfect, and it has been subject to further developments since the conception of Article IV. Some of these developments, and the plans for the future development of DESDEO, will be discussed in Section 6.3.

6.2 DESDEO's role in this thesis

In this section, we will elaborate more on how DESDEO has enabled the development and research involved in the interactive methods enhanced with explainability presented in this thesis. Beginning with the R-XIMO method discussed in

Chapter 3, DESDEO's role has been in primarily enabling the utilization of various scalarization functions, and providing different multiobjective optimization problems to experiment with. In addition, the evolutionary methods in DESDEO have played a key role in producing the populations that were utilized when generating SHAP values. Because the problems and scalarization functions used were under the same framework, scalarizing different problems was seamless, and did not require, e.g., redefining the problems or the functions each time they had to be reused. The availability of different evolutionary methods allowed the computation of various non-dominated solution sets, which were then combined to produce an approximation of the Pareto optimal front for different problems being studied. By combining the fronts generated by different methods, they can better represent the solution set of a problem[1]. This, in turn, enabled the computation of meaningful SHAP values in a feasible time. Consequently, this also led to the generation of accurate explanations and suggestions to support the decision maker in providing preferences in the R-XIMO method in an interactive setting.

Similar to what was done in R-XIMO, in the INFRINGER method discussed in Chapter 4, the evolutionary methods were used to compute a non-dominated solution set used in training the underlying belief rule-based system. Moreover, various tools found in DESDEO for manipulating and analyzing solution sets were utilized, such as the non-dominated sorting of solutions, and the approximation of the ideal and nadir points of a problem based on the available populations.

Lastly, in the XLEMOO method and its accompanying software framework, which were discussed in Chapter 5, DESDEO has again been in a central role. The availability of various multiobjective optimization problems, scalarization functions, and evolutionary methods in the same software framework, has been crucial. This has enabled the implementation of the XLEMOO interactive method, and the experimentation related to the work. Especially the evolutionary operators for multiobjective optimization, and the tools and utilities to calculate and manipulate solution populations, have been in a key role in developing and researching the XLEMOO method. Namely, the development of the XLEMOO method required the implementation of a learnable evolutionary algorithm for multiobjective optimization. The development of this algorithm was greatly expedited by the availability of the evolutionary operators suitable for multiobjective optimization found in DESDEO.

In essence, the fact that DESDEO provides different readily defined multiobjective optimization problems, scalarization functions, tools to handle solution sets, and various methods for multiobjective optimization in the same framework, has greatly supported the works related to Articles I, II, and III. Without DESDEO, the research and development of the interactive methods enhanced with explainability would have taken a lot more time. DESDEO has demonstrably enabled the re-use of existing software for interactive multiobjective optimization. This last fact alone makes DESDEO an important general asset in the research of other interactive

[1] This is because different evolutionary methods have different strengths and weaknesses, and can therefore generate diverse non-dominated solution sets.

multiobjective optimization methods as well.

6.3 Discussion of the DESDEO framework

While DESDEO has enabled a significant amount of the work related to the interactive methods discussed in this thesis, these works themselves have also outlined some of the flaws of DESDEO, at least in its current form. The current structure of DESDEO is not very suitable for implementing novel ideas as part of the framework. In fact, the software artifacts related to the R-XIMO, INFRINGER, and XLEMOO methods have not been made an integral part of DESDEO. But rather, they are their own entities that utilize DESDEO, i.e., applications of DESDEO. This makes the available software for enhancing interactive methods with explainability spread out and harder to find; even despite the fact that all the software related to the aforementioned works is openly available as open source software. One of the main reasons for why it is currently hard to implement tools that enable explainability in DESDEO, is because each of its packages are in their own repositories, and consequently under their individual version control systems. This makes working on more than one package simultaneously very challenging. Since explainability is an overarching concept, and it is related to multiple packages found in DESDEO, integrating explainability in the current framework is unnecessarily complex. However, it is also true that explainability in interactive methods is still a very novel concept. Therefore, generalizations and core ideas for explainability are yet to mature. This, in turn, makes the implementation of explainability related tools and utilities in DESDEO challenging as well.

Because of the nature of interactive multiobjective optimization methods, a user interface plays an important role in enabling decision makers to interact with the methods. DESDEO, as described in Article IV, was developed to mainly support the implementation of the algorithmic parts of various interactive methods. The inclusion of any user interfaces was left mostly open in the original work. This has made it challenging to experiment with any visualizations or interfaces for any of the interactive methods enhanced with explainability. Thus, the explanations were mostly communicated in a textual format to decision makers, which might not be optimal.

As the development of DESDEO has actively continued after the publication of Article IV, more efforts have been made in including a user interface in DESDEO, and also enabling DESDEO and the interactive methods therein to be accessed from any software application implementing a user interface. In fact, more recent developments in this direction are evident in the articles (Afsar et al., 2023, 2024), where DESDEO has been utilized in subject studies with human participants for comparing interactive methods. In these works, DESDEO has been connected to a custom web-based interface through a web-facing application programming interface, or web API. This enables any interface capable of handling HTTP-requests to connect to DESDEO and utilize the methods found in it. The concept of a web API for DESDEO was first explored by the author of this thesis in (Hakanen et al., 2022). When it comes to explainability, testing the actual benefits of the explanations should be assessed and studied with real decision makers as well. Therefore, having a user interface available can also expedite further studies on enhancing interactive methods with explainability. This will enable many future studies as well.

It is evident that DESDEO needs to be further developed to better meet the needs of explainability in interactive multiobjective optimization. Especially the inclusion of explainability related tools into the core framework, such as user interfaces and visualizations. The overall structure of DESDEO needs to be also rethought so that it will be more welcoming to the needs of explainability. Therefore, the works included in this thesis have also provided valuable insights to direct the future development of DESDEO. These, and other, enhancements will enable many future research directions in enhancing interactive methods with explainability.

In fact, the raised issues concerning DESDEO are already being addressed, and the development work required to address these is currently being spearheaded by the author of this thesis. This overhaul will result in a complete restructuring of DESDEO leading to a more versatile and flexible software framework capable of addressing a much wider assortment of needs in interactive multiobjective optimization, and its more novel branches. After the ongoing overhaul resulting in a *DESDEO 2.0*, a web API will be made an integral part of the framework enabling its connection to other software, and the framework will provide its own user interface, and many visualizations suitable for interactive methods. Moreover, the structure of the framework will be made much more welcoming to novel ideas, such as explainability, but also other emerging ideas, such as group decision-making in interactive methods (Pajasmaa, 2023), for instance. This overhaul will bring many other changes to DESDEO as well, including support for more types of multiobjective optimization problems, such as mixed-integer problems, more interfaces to connect DESDEO to existing single-objective optimizers, and a database (Saini et al., 2023) enabling many further aspects important in interactive methods, such as the archiving and comparison of solutions generated by different interactive methods, and the switching between interactive methods. The availability of these features will promote exploring and researching the potential of enhancing the decision-support capabilities of interactive multiobjective optimization to the next level.

Ultimately, the overhauled version of DESDEO, *DESDEO 2.0*, will be home to the ideas related to explainability discussed Chapters 3, 4, and 5, and much more. While the software artifacts related to the included works are currently freely and openly available, they are scattered. To enable the seamless future research of explainability in interactive methods, some of these existing ideas will be re-implemented in the overhauled version of DESDEO. This will bring explainability related enhancements closer to a myriad of different interactive methods, user interfaces, visualizations, and other tools for interactive multiobjective optimization. But most importantly, it will bring explanations for interactive methods much closer to other researchers and practitioners, and decision makers. As DESDEO continues to be developed further—by the author of this thesis, and others—it will become a central hub for openly available implementations of interactive methods enhanced with explainability.

7 CONCLUSIONS AND AUTHOR'S CONTRIBUTIONS

Thus far, we have seen examples on how explainability can be utilized to enhance the decision-support capabilities of interactive multiobjective optimization methods in Chapters 3, 4, and 5. We also discussed the role and importance of the DESDEO framework in integrating explainability with interactive methods in Chapter 6. In this chapter, we synthesize the main contributions of this thesis, and provide general future research directions for enhancing interactive multiobjective optimization with explainability.

This chapter is structured as follows: we begin with Section 7.1, where we outline the author's contributions in Articles I, II, III, and IV. Then, we shift the discussion to the main research questions of this thesis in Section 7.2, where we also provide a synthesis of the main findings and limitations of our work. Finally, we conclude this chapter—and consequently this thesis—with Section 7.3, where we discuss the future research directions and the role of this thesis in the conception of the novel field of explainable interactive multiobjective optimization.

7.1 Author's contributions

In this section, we will give a brief overview of the author's role and contribution in the included articles that have been co-authored. In all of the included articles, the author was the first and corresponding author. Particularly, in Articles II and III, the author was the sole author, and his contributions to these works do not require further elaborations.

In the work related to R-XIMO, in Article I, the author was in charge of the development and implementation of the proposed method, setting up the numerical tests and analyzing their results, conducting the experiments with a real decision maker, and writing the major part of the article. The author was also in a key role in the revision of the article based on the referees' feedback.

In the work related to DESDEO, in Article IV, the author has been in a major role in developing the DESDEO framework, focusing particularly on the development of the `desdeo-problem`, `desdeo-tools`, and `desdeo-mcdm` packages. The author was also involved in drafting the overall manuscript, with an emphasis on providing the examples involving scalarization-based interactive methods and their hybridization with evolutionary methods, and the section showcasing the potential user interface. The author also led the revision process in addressing the referees' comments.

7.2 Main conclusions

We have seen different perspectives in applying the concept of explainability to interactive multiobjective optimization in the works discussed in Chapters 3, 4, and 5. We are now in a position to return to the three main research questions of this thesis, which were presented in Chapter 1. We will answer each research question, RQ1–RQ3, in Sections 7.2.1, 7.2.2, and 7.2.3, respectively. Naturally, our answers to the research questions are not meant to be exhaustive, but rather act as examples and proof-of-concepts, providing us with valuable insights on the potential of explainability in interactive multiobjective optimization. Lastly, we synthesize the answers and highlight the main findings in Section 7.2.4.

7.2.1 The first research question

Our first research question (RQ1) was "In which ways can explainability aid a decision maker in finding their most preferred solution to a multiobjective optimization problem when applying an interactive multiobjective optimization method?" With R-XIMO (c.f., Chapter 3), we are able to provide a decision maker with an explanation on how the preferences they have provided have affected the computed solution in a reference point based interactive multiobjective optimization method. Furthermore, we also support the decision maker in providing further preferences by generating suggestions based on the explanations. The explanations can also provide the decision maker with insight on the trade-offs between the objective functions near solutions of interest. Moreover, these suggestions can potentially help a decision maker in reaching their preferred solution in fewer iterations.

From the perspective of the INFRINGER method (c.f., Chapter 4), explainability could play a role in modeling the preferences of a decision maker. While this was not directly explored in the presented work, belief rule-based systems have the option to incorporate knowledge provided by the decision maker when building the rule base. By incorporating such knowledge, the rules modeling the preferences, and the corresponding belief degrees, may provide the decision maker with valuable information on their preferences, possibly supporting their learning. Moreover, by modeling the rules according to the knowledge of the decision maker, we could also elicit the preference model in a way, which is more understandable to the decision maker. This, in turn, can help the decision maker in providing preferences.

Lastly, in the XLEMOO method (c.f., Chapter 5), the populations generated by an evolutionary method, and the preferences expressed by a decision maker, can be utilized in an interactive setting to generate explanations on the connection of the found non-dominated solutions and their corresponding objective function values. The explanations then provide the decision maker with information on the characteristics of the decision variable values of the solutions that best match their preferences. This can provide the decision maker with insights that would otherwise, in the absence of explanations, not be available. The explanations can also support the decision maker in providing preferences when applying an interactive method.

7.2.2 The second research question

The second research question (RQ2) was: "Does incorporating explainability in interactive multiobjective optimization methods improve their decision-support capabilities?" This question is tricky to answer because evaluating the decision support provided by an interactive method can be a very subjective matter. However, both the R-XIMO method and the INFRINGER method were tested by a decision maker with expertise in the domain of the problem being solved. In the case of R-XIMO, the decision maker clearly expressed that the suggestions derived from the explanations were useful. Likewise, the decision maker expressed the solutions computed by the INFRINGER method to match their expectations, but this was not directly attributed to explanations. Lastly, in the XLEMOO method, it was demonstrated in a hypothetical showcase how the explanations describing decision variable values related to solutions near the preferences could prove informative to a more engineering oriented decision maker. For such decision makers, the decision variable values can play an important role in the overall decision-making process.

7.2.3 The third research question

The third research question (RQ3) was: "In which ways can explainability improve the justifiability of the decision made when applying interactive multiobjective optimization methods?" Beginning with the R-XIMO method, the addition of explainability to understand the connection between the provided preferences and the computed objective vectors can help the decision maker to justify the trade-offs made when they change their preferences. To exemplify this, we can consider a case where the decision maker wishes to improve the objective function value of a certain objective function, but does not know how they should modify the reference point. Instead of guessing, the suggestions derived from the explanations in R-XIMO directly tell the decision maker how the preferences have affected the output of the interactive method. This gives the decision maker grounds to make the decision on how to modify their preferences, and thanks to the explanations, they can justify their choices in modifying the preferences. I.e., they can justify to themselves, and possibly other stakeholders, the worsening of a certain component in a new reference point, when compared to the previous one, with the knowledge that the change will very likely lead to the best possible gain in the value of the objective function they wish to improve. Therefore, the preferences given by a decision maker, that have led to a decision in an interactive method, can be justified to the decision maker, and other stakeholders, based on the explanations.

When a belief rule-based model is utilized to model the preferences of a decision maker during an interactive multiobjective optimization method, the rule base utilized could be directly initialized based on information provided by the decision maker. The belief rule-base model can then be improved by querying the decision maker, as was done in the INFRINGER method, to fine-tune the preference model during the course of an interactive method. This can lead to an understandable model, where the trained rules and associated belief degrees can be communicated to a decision maker in an explainable way. This means that the preference model becomes understandable by the decision maker, which can support them to understand why an interactive method generated certain solutions. Consequently, a decision maker can then take an active role in not just providing the preferences to train the preference model, but possibly also in auditing it, which is made possible by the model's explainable nature. Instead of guessing if a preference model is correct or not, a decision maker may be able verify it. If a decision maker can be confident that a preference model utilized is indeed in accord with their own preferences, then the decision maker can justify to themselves, and to other stakeholders, the solutions found based on the observation that the preference model accurately represents their values and goals. This, in turn, can increase the confidence and trust the decision maker has in the optimization process and its outcomes.

In the XLEMOO method, the justifiability of the solutions found is again improved thanks to the available explanations. The decision maker can base the changes in their preferences on the characteristics of the decision variable values of the solutions found near their preferences. Moreover, if the decision maker is a more technically inclined person, and is familiar with the formulation of the problem being solved, the characteristics of the decision variable values communicated by the explanations can also support the decision maker in auditing the problem formulation itself. This can either promote confidence or raise doubts in the decision maker, when it comes to the problem model. These aspects can help a decision maker in justifying to themselves, and other stakeholders, the changes made in the provided preferences during an interactive process, and the reliability of the solutions found.

7.2.4 The enhancements provided by explainability

Table 1 presents a summary of the responses to the three research questions, as interpreted in the context of the three interactive multiobjective optimization methods enhanced with explainability presented in this thesis: R-XIMO, INFRINGER, and XLEMOO. These methods have approached explainability from varying perspectives. Therefore, we are left with varied answers to the research questions as well.

TABLE 1 Summary of the answers to the three main research questions of this thesis. The questions where RQ1: "In which ways can explainability aid a decision maker in finding their most preferred solution to a multiobjective optimization problem when applying an interactive multiobjective optimization method?"; RQ2: "Does incorporating explainability in interactive multiobjective optimization methods improve their decision-support capabilities?"; and RQ3: "In which ways can explainability improve the justifiability of the decision made when applying interactive multiobjective optimization methods?" The answers have been provided from the perspective of the three explainable interactive methods discussed in this thesis: R-XIMO (c.f., Chapter 3), INFRINGER (c.f., Chapter 4), and XLEMOO (c.f., Chapter 5).

	R-XIMO	INFRINGER	XLEMOO
RQ1	providing preferences, understanding the connection between preferences and objective function values, learning about trade-offs	understanding and learning about preferences, providing preferences	understanding the characteristics of preferred solutions in terms of decision variable values, providing preferences
RQ2	yes, explanations contribute directly to decision-support	maybe, explainable model used, but explanations do not directly contribute to decision-support	maybe, explanations contributed directly to decision-support, but not assessed by a real decision maker
RQ3	help the decision maker explicate the changes in their preferences	allow the decision maker to audit a preference model used to find solutions	help the decision maker explicate the changes in their preferences, audit the correctness of the problem's model

From Table 1, we notice that explainability offers a very promising aid to decision makers (RQ1) when providing preferences in interactive methods. In addition, explanations can also provide the decision maker with knowledge on aspects of the problem, the solutions, and their own preferences, that can be of further aid in finding solutions during the course of applying an interactive method. In other words, the explanations enable the decision maker to learn about these aspects. That being said, it is important to highlight the fact that in the INFRINGER method, explainability was posed as an advantage of the utilized belief rule-based model, but was not directly utilized because of the complexity of the model. This limitation withstanding, a simpler model, with a less complicated rule-base, could deliver the previously discussed advantages related to the INFRINGER method.

However, whether explanations really improve the decision-support capabilities of interactive methods (RQ2) is not completely clear from the research presented in this thesis. As seen from Table 1, only in the R-XIMO method we are confident that explanations have improved an interactive method's decision-support capabilities. But even there, we only have feedback from a single decision maker to rely on. Moreover, in the INFRINGER method, explainability did not directly play a role in supporting the decision maker, and in the XLEMOO method, the explanations were not assessed by a real decision maker. These are notable limitations of the presented works as well.

The justifiability that explanations provide (RQ3) in interactive methods has clearly the most potential when it comes to a decision maker being able to justify the preferences used to themselves and other stakeholders, as seen from Table 1. Of course, justifiability can also be a subjective matter, and may depend on the situation in which the justifications are needed. However, explainability has a promising potential to offer grounds to decision makers to justify the provided preference that have led to a decision in an interactive method. Explanations related to the connection of decision variable values and their corresponding objective functions values may also provide grounds to decision makers to justify the validity of the solutions found by interactive methods. That being said, any justifications derived from explanations in the context of interactive methods, should ultimately be judged by a decision maker.

In summary, explanations have clearly the most potential when it comes to supporting a decision maker in providing preferences, learning about their preferences, and understanding the connection between their preferences and the computed solutions in interactive multiobjective optimization. Other aspects, where explanations can play a key role in enhancing the decision-support capabilities of interactive methods, lie in explicating how a decision maker's preferences have been modeled, the characteristics of preferred solutions in other terms than just objective function values—in this case decision variable values—and verifying the utilized problem model. Whether these truly improve the decision-support capabilities of interactive methods remains an open question, and deserve further studies beyond the scope of this thesis. Furthermore, the ways that explanations improve the justifiability of the decision made with the aid of interactive methods, are also subject to further debate. Despite these loose ends, it is nevertheless clear that the further study and research on the application of explainability to enhance interactive multiobjective optimization methods, is not just a promising new research avenue, but also necessary if we wish to provide increasingly better decision-support to decision makers in tackling multiobjective optimization problems in the future as well.

7.3 Future directions

We already outlined potential future research directions for the three interactive methods in their respective chapters (Chapters 3, 4, and 5). And we presented some of the future steps in the development of DESDEO in Chapter 6 as well. However, in this section, we outline some of the general future research directions that are needed, and have the most potential, based on the overall findings and limitations of this thesis.

First of all, the communication of explanations to a decision maker requires further research. In this thesis, all explanations have been presented in a textual format. While text explanations are easy to generate, they are also limited in their capability to convey more complex information in an understandable way. For instance, in the R-XIMO method, the decision maker found the explanations to be too verbose. Likewise, the rules generated by the XLEMOO method can become harder to interpret as the number of decision variables in the problem being solved increases. Not to mention the complexity of the preference model in the INFRINGER method. Specialized visualizations for communicating explanations in the context of interactive methods could address these issues, as they could be suitable for communicating more involved explanations in a comprehensive way. For instance, SHAP value are often communicated via so-called waterfall and bee swarm plots in machine learning applications[1]. However, the usefulness of these, and other existing visualizations, and the possibility of developing specialized new visualizations to address the challenges unique to interactive methods, remains an open research direction.

Another direction is to apply explainability to other existing interactive methods. When it comes to explaining the connection between preferences and generated solutions, methods with different types of preference information, such as those belonging to the NAUTILUS family and the synchronous NIMBUS method (c.f., Chapter 2), would be promising next steps. For instance, the trade-off free nature of the NAUTILUS methods would perhaps benefit from counterfactual explanations, providing the decision maker with detailed insights on the solutions they are approaching as they near the Pareto optimal front. Different interactive methods will likely benefit from different types of explanations.

The large number of solutions generated in evolutionary multiobjective optimization methods makes them also succulent to be studied further by enhancing them with explainable machine learning, as was done in the XLEMOO method in an indicator-based evolutionary method. In addition, applying an explainable

[1] C.f., https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html. Accessed: May 9, 2024.

classifier to study the properties of the rankings and decomposition produced by domination- and decomposition-based interactive methods, respectively (c.f., Chapter 2), is as well an interesting research direction. It was also noted that the incorporation of learnable evolutionary models can also result in improved performance of evolutionary methods, which is certainly also a matter worth of further research.

Additionally, it is obvious from the results of this thesis that we need further studies with real decision makers to accurately measure the actual benefits of explainability introduced to interactive methods. This is also related to the matter of auditing potentially explainable preference models, as present in the INFRINGER method. These kinds of studies could provide insights on what kind of explanations can serve decision makers best when it comes to enhancing decision-support. The previously discussed visualizations for explainable interactive methods should also be explored in these kind of studies.

The future research directions discussed thus far are very experimental in their nature. To support these directions, the availability of interactive methods for researchers to experiment with freely will be very important. The DESDEO framework will be in a key role in these future studies as well. As our understanding on how to best enhance interactive methods with explainability increases, so will the software needs evolve. To best facilitate these endeavors, DESDEO must be constantly developed to meet the various emerging requirements of explainability. Additionally, to supplement these experimental research directions, theoretical studies should be pursued as well. These studies could leverage the mathematical properties unique to multiobjective optimization problems for enhancing the explainability of interactive methods.

In essence, the main general research directions put forth by this thesis can be condensed into three:

1. researching novel ways to communicate explanations to decision makers in interactive methods,
2. applying explanations to a wider range of different interactive methods, and
3. studying the benefits of explanations in interactive methods in studies with real decision makers.

These encompass, however, only the tip of the iceberg, and by applying a hint of imagination, one can easily come up with a lot more additional research directions to enhance interactive multiobjective optimization with explainability.

To conclude, based on the results of this thesis, explanations are still a very fresh concepts in the context of interactive multiobjective optimization. It is evident that the further study of explanations to enhance interactive methods is more than warranted. This thesis has laid one of the foundational bricks on the road paving our way towards a new breed of interactive multiobjective optimization methods enhanced with explainability. We posit with confidence that the field of *explainable interactive multiobjective optimization* will gain more and more traction in the coming decades.

Ultimately, the efforts in pursuing research in the direction outlined by this thesis will amount to better decision-support tools for solving multiobjective optimization problems across different problem domains. These problems will be accompanied by a diverse assortment of decision makers, each of them equipped with a similarly diverse array of decision-support needs. And part of these needs will be met by enhancing the decision-support tools utilized with explainability.

YHTEENVETO (SUMMARY IN FINNISH)

Tässä väitöskirjassa esiteltiin kolme uutta interaktiivista monitavoiteoptimointimenetelmää: R-XIMO, INFRINGER ja XLEMOO. Nämä menetelmät hyödyntävät vaihtelevasti selitettävyyttä tukien päätöksentekijää monitavoiteoptimointiongelmien ratkaisemisessa. Myös avoimen lähdekoodin DESDEO-ohjelmistokehys esiteltiin, koska se on tukenut edellä mainittujen menetelmien kehittämistä.

Väitöskirjan johtopäätökset osoittavat, että selitettävyydellä on selkeästi potentiaalia tukea päätöksentekijöitä interaktiivisessa monitavoiteoptimoinnissa monilla eri tavoilla. Selitettävyys voi auttaa päätöksentekijöitä mieltymysten ilmaisemisessa, sekä ymmärtämään miten mieltymykset ovat vaikuttaneet ratkaisuprosessiin ja ratkaisujen laskemiseen. Selitettävyys voi lisäksi myös tukea päätöksentekijää perustelemaan tehtyjä valintoja ja päätöksia muille asianomaisille, itselleen, sekä se voi paljastaa tärkeää tietoa liittyen monitavoiteoptimointiongelman mallinnukseen. Selitettävyys voi myös auttaa päätöksentekijän mieltymysten mallintamisessa ja näiden mallien ymmärtämisessä.

Esitetyn tutkimuksen selkeimpiä rajoitteita oli selitettävyyden tutkiminen rajoitetulla määrällä interaktiivisia menetelmiä. Lisäksi, itse selitettävyyden hyötyjä päätöksentekijälle on vaikea arvioida ilman laajempaa otantaa, jossa useamman päätöksentekijän kokemuksia selitettävyyden tuomista hyödyistä voisi mitata. Erityisesti päätöksentekijän mieltymysten mallintamisessa, selitettävyyden hyötyä ei saatu varmuudella selvitettyä, vaikkakin sen hyödyntäminen vaikuttaa lupaavalta.

Tulevaisuuden tutkimussuuntauksia selitettävyyden hyödyntämisessä interaktiivisessa monitavoiteoptimoinnissa tunnistettiin myös paljon. Nämä voidaan tiivistää kolmeen pääsuuntaukseen: 1. uusien menetelmien tutkiminen, joiden avulla selityksiä voidaan tehokkaammin kommunikoida päätöksentekijöille, kuten erilaiset visualisoinnit; 2. selitettävyyden soveltaminen muiden, kuin tutkimuksessa esiteltyjen, interaktiivisten menetelmien parantamiseen; sekä 3. selitettävyyden tuomien etujen arviointi tutkimuksissa, joihin osallistetaan oikeita päätöksentekijöitä. DESDEO-ohjelmistokehys voi auttaa paljon näiden suuntauksien tavoittelemisessa, mutta itse kehystä täytyy myös ensin parantaa, jotta selitettävyyttä voidaan soveltaa paremmin interaktiivisissa menetelmissä.

Kaiken kaikkiaan, väitöskirja osoittaa kuitenkin, että selitettävyyden soveltaminen interaktiivisessa monitavoiteoptimoinnissa on hyvin lupaavaa. Tämä voi johtaa uusien interaktiivisten menetelmien kehittämiseen, ja voi myös auttaa olemassa olevien menetelmien parantamisessa. Väitöskirja on luonut pohjan uudelle tutkimussuuntaukselle, eli selitettävälle interaktiiviselle monitavoiteoptimoinnille. Selitettävyyden avulla voimme jatkossa kehittää työkaluja päätöksenteon tukemiseksi, jotka ovat ymmärrettäviä, läpinäkyviä, sekä auttavat päätöksentekijöitä löytämään parhaan mahdollisen ratkaisun ongelmiin, joissa on useampi ristiriitainen tavoite.

REFERENCES

Afsar, B., Fieldsend, J. E., Guerreiro, A. P., Miettinen, K., Rojas Gonzalez, S. & Sato, H. 2023. Many-objective quality measures. In D. Brockhoff, M. Emmerich, B. Naujoks & R. R. Purshouse (Eds.) Many-Criteria Optimization and Decision Analysis. Springer, 113–148.

Afsar, B., Miettinen, K. & Ruiz, F. 2021. Assessing the performance of interactive multiobjective optimization methods: a survey. ACM Computing Surveys (CSUR) 54 (4), 1–27.

Afsar, B., Silvennoinen, J., Misitano, G., Ruiz, F., Ruiz, A. B. & Miettinen, K. 2023. Designing empirical experiments to compare interactive multiobjective optimization methods. Journal of the Operational Research Society 74 (11), 2327–2338.

Afsar, B., Silvennoinen, J., Ruiz, F., Ruiz, A. B., Misitano, G. & Miettinen, K. 2024. An experimental design for comparing interactive methods based on their desirable properties. Annals of Operations Research. doi:https://doi.org/10.1007/s10479-024-05941-6.

Allmendinger, R., Ehrgott, M., Gandibleux, X., Geiger, M. J., Klamroth, K. & Luque, M. 2017. Navigation in multiobjective optimization methods. Journal of Multi-Criteria Decision Analysis 24 (1-2), 57–70.

Alves, M. J. & Costa, J. P. 2009. An exact method for computing the nadir values in multiple objective linear programming. European Journal of Operational Research 198 (2), 637–646.

Antonio, L. M. & Coello, C. A. C. 2017. Coevolutionary multiobjective evolutionary algorithms: survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 22 (6), 851–865.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R. et al. 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion 58, 82–115.

Avriel, M. 2020. Nonlinear programming. In Mathematical Programming for Operations Researchers and Computer Scientists. CRC Press, 271–367.

Battiti, R. & Passerini, A. 2010. Brain–computer evolutionary multiobjective optimization: a genetic algorithm adapting to the decision maker. IEEE Transactions on Evolutionary Computation 14 (5), 671–687.

Belton, V., Branke, J., Eskelinen, P., Greco, S., Molina, J., Ruiz, F. & Słowiński, R. 2008. Interactive multiobjective optimization from a learning perspective. In J. Branke, K. Deb, K. Miettinen & R. Słowiński (Eds.) Multiobjective Optimization: Interactive and Evolutionary Approaches. Springer, 405–433.

Benayoun, R., de Montgolfier, J., Tergny, J. & Laritchev, O. 1971. Linear programming with multiple objective functions: Step method (stem). Mathematical Programming 1 (1), 366–375.

Bermúdez, J. L. 2009. Decision theory and rationality. Oxford University Press.

Biran, O. & Cotton, C. 2017. Explanation and justification in machine learning: A survey. In IJCAI-17 Workshop on Explainable AI (XAI), 8–13.

Biresselioglu, M. E., Kaplan, M. D. & Yilmaz, B. K. 2018. Electric mobility in Europe: A comprehensive review of motivators and barriers in decision making processes. Transportation Research Part A: Policy and Practice 109, 1–13.

Bishop, C. M. & Nasrabadi, N. M. 2006. Pattern recognition and machine learning. Springer.

Branke, J., Branke, J., Deb, K., Miettinen, K. & Slowiński, R. (Eds.) 2008. Multiobjective optimization: Interactive and evolutionary approaches. Springer.

Branke, J., Greco, S., Słowiński, R. & Zielniewicz, P. 2014. Learning value functions in interactive evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation 19 (1), 88–102.

Brockhoff, D., Emmerich, M., Naujoks, B. & Purshouse, R. 2023. Many-criteria Optimization and Decision Analysis: State-of-the-art, Present Challenges, and Future Perspectives. Springer.

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

Cao, Y., Zhou, Z., Hu, C., He, W. & Tang, S. 2020. On the interpretability of belief rule-based expert systems. IEEE Transactions on Fuzzy Systems 29 (11), 3489–3503.

Censor, Y. 1977. Pareto optimality in multiobjective problems. Applied Mathematics and Optimization 4 (1), 41–59.

Chen, Y.-W., Yang, J.-B., Xu, D.-L., Zhou, Z.-J. & Tang, D.-W. 2011. Inference analysis and adaptive training for belief rule based systems. Expert Systems with Applications 38 (10), 12845–12860.

Cheng, R., Jin, Y., Olhofer, M. & Sendhoff, B. 2016. A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation 20 (5), 773–791.

Chugh, T., Sindhya, K., Hakanen, J. & Miettinen, K. 2019. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. Soft Computing 23, 3137–3166.

Confalonieri, R., Coba, L., Wagner, B. & Besold, T. R. 2020. A historical perspective of explainable artificial intelligence. WIREs Data Mining and Knowledge Discovery 11 (1).

Corrente, S., Greco, S., Matarazzo, B. & Słowiński, R. 2024. Explainable interactive evolutionary multiobjective optimization. Omega 122, 102925.

Deb, K., Abouhawwash, M. & Dutta, J. 2015. An optimality theory based proximity measure for evolutionary multi-objective and many-objective optimization. In A. Gaspar-Cunha, C. Henggeler Antunes & C. C. Coello (Eds.) Evolutionary Multi-Criterion Optimization. Springer, 18–33.

Deb, K., Agrawal, R. B. et al. 1995. Simulated binary crossover for continuous search space. Complex Systems 9 (2), 115–148.

Deb, K. & Jain, H. 2013. 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.

Deb, K., Miettinen, K. & Chaudhuri, S. 2010. Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. IEEE Transactions on Evolutionary Computation 14 (6), 821–841.

Deb, K. & Miettinen, K. 2009. A review of nadir point estimation procedures using evolutionary approaches: A tale of dimensionality reduction. In Proceedings of the Multiple Criterion Decision Making (MCDM-2008) Conference, 1–14.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Rransactions on Evolutionary Computation 6 (2), 182–197.

Deb, K. & Srinivasan, A. 2006. Innovization: Innovating design principles through optimization. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 1629–1636.

Deb, K., Tewari, R., Dixit, M. & Dutta, J. 2007. Finding trade-off solutions close to KKT points using evolutionary multi-objective optimization. In 2007 IEEE Congress on Evolutionary Computation. IEEE.

Dessouky, M., Ghiassi, M. & Davis, W. 1986. Estimates of the minimum nondominated criterion values in multiple-criteria decision-making. Engineering Costs and Production Economics 10 (1), 95–104.

Dudas, C., Ng, Amos H.C.and Pehrsson, L. & Boström, H. 2013. Integration of data mining and multi-objective optimisation for decision support in production systems development. International Journal of Computer Integrated Manufacturing 27 (9), 824–839.

El Morr, C. & Ali-Hassan, H. 2019. Descriptive, predictive, and prescriptive analytics. In C. El Morr & H. Ali-Hassan (Eds.) Analytics in Healthcare: A Practical Introduction. Springer, 31–55.

Flach, P. 2012. Machine learning: the art and science of algorithms that make sense of data. Cambridge University Press.

Flora, M., Potvin, C., McGovern, A. & Handler, S. 2022. Comparing explanation methods for traditional machine learning models part 1: an overview of current methods and quantifying their disagreement. https://doi.org/10.48550/arXiv.2211.08943.

Gardiner, L. R. & Vanderpooten, D. 1997. Interactive multiple criteria procedures: some reflections. In J. Clímaco (Ed.) Multicriteria Analysis. Springer, 290–301.

Gass, S. I. & Assad, A. A. 2005. An annotated timeline of operations research: An informal history. Springer.

Goienetxea Uriarte, A., Ruiz Zúñiga, E., Urenda Moris, M. & Ng, A. H. 2017. How can decision makers be supported in the improvement of an emergency department? A simulation, optimization and data mining approach. Operations Research for Health Care 15, 102—122.

Goix, N., Birodkar, V., Gardin, F., Schertzer, J.-M., Jeong, H., Kumar, M., Gramfort, A., Staley, T., Tour, T. D. L., Deng, B., C, Pedregosa, F., Wu, L., Rokem, A., Jackson, K. & Mrahim 2020. scikit-learn-contrib/skope-rules v1.0.1. https://zenodo.org/doi/10.5281/zenodo.4316670.

Goodman, B. & Flaxman, S. 2017. European Union regulations on algorithmic decision-making and a "right to explanation". AI Magazine 38 (3), 50–57.

Greco, S., Matarazzo, B. & Slowinski, R. 2001. Rough sets theory for multicriteria decision analysis. European Journal of Operational Research 129 (1), 1–47.

Gunning, D. & Aha, D. 2019. DARPA's explainable artificial intelligence (XAI) program. AI Magazine 40 (2), 44–58.

Hakanen, J., Chugh, T., Sindhya, K., Jin, Y. & Miettinen, K. 2016. Connections of reference vectors and different types of preference information in interactive multiobjective evolutionary algorithms. In 2016 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 1–8.

Hakanen, J., Radoš, S., Misitano, G., Saini, B. S., Miettinen, K. & Matković, K. 2022. Interactivized: visual interaction for better decisions with interactive multiobjective optimization. IEEE Access 10, 33661–33678.

Hwang, C.-L. & Masud, A. 1979. Multiple Objective Decision Making–Methods and Applications: A State-of-the-Art Survey. Springer.

Isermann, H. & Steuer, R. E. 1988. Computational experience concerning payoff tables and minimum criterion values over the efficient set. European Journal of Operational Research 33 (1), 91–97.

Ishibuchi, H., Tsukamoto, N. & Nojima, Y. 2008. Evolutionary many-objective optimization: A short review. In 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). IEEE, 2419–2426.

James, G., Witten, D., Hastie, T. & Tibshirani, R. 2013. An introduction to statistical learning. Springer.

Jarecki, J. B. & Rieskamp, J. 2022. Comparing attribute-based and memory-based preferential choice. Decision 49 (1), 65–90.

Jordan, M. I. & Mitchell, T. M. 2015. Machine learning: trends, perspectives, and prospects. Science 349 (6245), 255–260.

Jourdan, L., Corne, D., Savic, D. & Walters, G. 2005. Preliminary investigation of the 'learnable evolution model'for faster/better multiobjective water systems design. In Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005. Proceedings 3. Springer, 841–855.

Kamath, U. & Liu, J. 2021. Explainable artificial intelligence: An introduction to interpretable machine learning. Springer.

Kania, A., Sipilä, J., Misitano, G., Miettinen, K. & Lehtimäki, J. 2022. Integration of lot sizing and safety strategy placement using interactive multiobjective optimization. Computers & Industrial Engineering 173, 108731.

Kaplan, J. 2016. Artificial intelligence: What everyone needs to know. Oxford University Press.

Kaplan, R. M. & Frosch, D. L. 2005. Decision making in medicine and health care. Annual Review of Clinical Psychology 1, 525–556.

Keeney, R. L. & Raiffa, H. 1993. Decisions with multiple objectives: preferences and value trade-offs. Cambridge University Press.

Korhonen, P., Salo, S. & Steuer, R. E. 1997. A heuristic for estimating nadir criterion values in multiple objective linear programming. Operations Research 45 (5), 751–757.

Korhonen, P. & Wallenius, J. 1988. A pareto race. Naval Research Logistics 35 (6), 615–623.

Köksalan, M. M., Wallenius, J. & Zionts, S. 2011. Multiple criteria decision making: from early history to the 21st century. World Scientific.

Legg, S., Hutter, M. et al. 2007. A collection of definitions of intelligence. Frontiers in Artificial Intelligence and Applications 157, 17.

Li, J.-P., Wang, Y., Yang, S. & Cai, Z. 2016. A comparative study of constraint-handling techniques in evolutionary constrained multiobjective optimization. In 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE, 4175–4182.

Lin, P., Zhang, L. & Tiong, R. L. 2023. Multi-objective robust optimization for enhanced safety in large-diameter tunnel construction with interactive and explainable AI. Reliability Engineering & System Safety 234, 109172.

Linardatos, P., Papastefanopoulos, V. & Kotsiantis, S. 2020. Explainable AI: a review of machine learning interpretability methods. Entropy 23 (1), 18.

Lipton, Z. C. 2018. The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery. Queue 16 (3), 31–57.

Lundberg, S. M. & Lee, S.-I. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (Eds.) Advances in Neural Information Processing Systems 30. Curran Associates, Inc., 4765–4774.

Lárraga, G., Saini, B. S. & Miettinen, K. 2023. Incorporating preference information interactively in NSGA-III by the adaptation of reference vectors. In International Conference on Evolutionary Multi-Criterion Optimization. Springer, 578–592.

MacKay, D. J. 2003. Information theory, inference and learning algorithms. Cambridge university press.

Mckay, M. D., Beckman, R. J. & Conover, W. J. 2000. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 42 (1), 55–61.

Michalski, R. S. 1983. A theory and methodology of inductive learning. In Machine learning. Elsevier, 83–134.

Michalski, R. S. 2000. Learnable evolution model: Evolutionary processes guided by machine learning. Machine learning 38, 9–40.

Miettinen, K., Hakanen, J. & Podkopaev, D. 2016. Interactive nonlinear multiobjective optimization methods. In S. Greco, M. Ehrgott & J. R. Figueira (Eds.) Multiple Criteria Decision Analysis: State of the Art Surveys. Springer, 927–976.

Miettinen, K. & Mäkelä, M. M. 2006. Synchronous approach in interactive multiobjective optimization. European Journal of Operational Research 170 (3), 909—922.

Miettinen, K. & Mäkelä, M. M. 2002. On scalarizing functions in multiobjective optimization. OR Spectrum 24 (2), 193–213.

Miettinen, K., Ruiz, F. & Wierzbicki, A. P. 2008. Introduction to multiobjective optimization: interactive approaches. In J. Branke, K. Deb, K. Miettinen & R. Słowiński (Eds.) Multiobjective Optimization: Interactive and Evolutionary Approaches. Springer, 27–57.

Miettinen, K. & Ruiz, F. 2016. NAUTILUS framework: towards trade-off-free interaction in multiobjective optimization. Journal of Business Economics 86 (1–2), 5–21.

Miettinen, K. 1999. Nonlinear multiobjective optimization. Boston: Kluwer Academic Publishers.

Miller, B. L., Goldberg, D. E. et al. 1995. Genetic algorithms, tournament selection, and the effects of noise. Complex Systems 9 (3), 193–212.

Misitano, G. 2020. INFRINGER: a novel interactive multi-objective optimization method able to learn a decision maker's preferences utilizing machine learning. University of Jyväskylä. Master's Thesis.

Misitano, G. 2023. gialmisi/XLEMOO: article code. https://zenodo.org/doi/10.5281/zenodo.8090344.

Moradi, B. & Mirzaei, A. 2016. A new automated design method based on machine learning for cmos analog circuits. International Journal of Electronics 103 (11), 1868–1881.

Moradi, B. 2019. Multi-objective mobile robot path planning problem through learnable evolution model. Journal of Experimental & Theoretical Artificial Intelligence 31 (2), 325–348.

Moradi, B. 2020. The new optimization algorithm for the vehicle routing problem with time windows using multi-objective discrete learnable evolution model. Soft Computing 24 (9), 6741–6769.

Morgenstern, O. 1953. Theory of games and economic behavior. Princeton University Press.

Nakayama, H. 1995. Aspiration level approach to interactive multi-objective programming and its applications. In P. M. Pardalos, Y. Siskos & C. Zopounidis (Eds.) Advances in Multicriteria Analysis. Springer, 147–174.

Niu, Y., Kong, D., Wen, R., Cao, Z. & Xiao, J. 2021. An improved learnable evolution model for solving multi-objective vehicle routing problem with stochastic demand. Knowledge-Based Systems 230, 107378.

Nocedal, J. & Wright, S. J. 1999. Numerical optimization. Springer.

Ordóñez, C., Threlfall, C. G., Kendal, D., Hochuli, D. F., Davern, M., Fuller, R. A., van der Ree, R. & Livesley, S. J. 2019. Urban forest governance and decision-making: A systematic review and synthesis of the perspectives of municipal managers. Landscape and Urban Planning 189, 166–180.

Osika, Z., Salazar, J. Z., Roijers, D. M., Oliehoek, F. A. & Murukannaiah, P. K. 2023. What lies beyond the pareto front? a survey on decision-support methods for multi-objective optimization. In Proceedings of the 32nd International Joint

Conference on Artificial Intelligence, IJCAI 2023 (IJCAI-23). International Joint Conferences on Artificial Intelligence, 6741–6749.

Pajasmaa, J. 2023. Group decision making in multiobjective optimization: a systematic literature review. University of Jyväskylä. Master's Thesis.

Pedro, L. R. & Takahashi, R. H. 2013. Decision-maker preference modeling in interactive multiobjective optimization. In Evolutionary Multi-Criterion Optimization: 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings 7. Springer, 811–824.

Pour, P. A., Bandaru, S., Afsar, B. & Miettinen, K. 2022. Desirable properties of performance indicators for assessing interactive evolutionary multiobjective optimization methods. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, 1803–1811.

Ribeiro, M. T., Singh, S. & Guestrin, C. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144.

Roy, D., Srivastava, R., Jat, M. & Karaca, M. S. 2022. A complete overview of analytics techniques: descriptive, predictive, and prescriptive. In P. M. Jeyanthi, T. Choudhury, D. Hack-Polay, T. P. Singh & S. Abujar (Eds.) Decision Intelligence Analytics and the Implementation of Strategic Business Management. Springer, 15–30.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence 1 (5), 206–215.

Ruiz, A. B., Ruiz, F., Miettinen, K., Delgado-Antequera, L. & Ojalehto, V. 2019. NAUTILUS Navigator: free search interactive multiobjective optimization without trading-off. Journal of Global Optimization 74 (2), 213–231.

Sachan, S., Yang, J.-B., Xu, D.-L., Benavides, D. E. & Li, Y. 2020. An explainable AI decision-support-system to automate loan underwriting. Expert Systems with Applications 144, 113100.

Saini, B. S., Lárraga, G. & Miettinen, K. 2023. Using a database to support interactive multiobjective optimization, visualization, and analysis. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation, 1703–1711.

Sawaragi, Y., Nakayama, H. & Tanino, T. 1985. Theory of multiobjective optimization. Acedemic Press, Inc.

Schaffer, J. D. 1985. Multiple objective optimization with vector evaluated genetic algorithms. In Proceedings of the 1st International Conference on Genetic Algorithms. Psychology Press, 93–100.

Searle, J. R. 1980. Minds, brains, and programs. Behavioral and Brain Sciences 3 (3), 417–424.

Shapley, L. S. 1953. A value for N-person games. Contributions to the Theory of Games 2 (28), 307–317.

Shavarani, S. M., López-Ibáñez, M., Allmendinger, R. & Knowles, J. 2023. An interactive decision tree-based evolutionary multi-objective algorithm. In International Conference on Evolutionary Multi-Criterion Optimization. Springer, 620–634.

Slack, D., Hilgard, S., Jia, E., Singh, S. & Lakkaraju, H. 2020. Fooling LIME and SHAP: adversarial attacks on post hoc explanation methods. In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. ACM. AIES ’20.

Smedberg, H. 2023. Knowledge discovery for interactive decision support and knowledge-driven optimization. University of Skövde. Ph. D. Thesis.

Stankevich, A. 2017. Explaining the consumer decision-making process: critical literature review. Journal of International Business Research and Marketing 2 (6).

Steuer, R. E. 1986. Multiple Criteria Optimization: Theory, Computation, and Application. Wiley, New York.

Sukkerd, R., Simmons, R. & Garlan, D. 2018. Towards explainable multi-objective probabilistic planning. In Proceedings of the 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems, 19–25.

Sutton, R. S. & Barto, A. G. 2018. Reinforcement learning: An introduction. MIT press.

Tanabe, R. & Ishibuchi, H. 2020. An easy-to-use real-world multi-objective optimization problem suite. Applied Soft Computing 89, 106078.

Tanner, L. 1991. Selecting a text-processing system as a qualitative multiple criteria problem. European Journal of Operational Research 50 (2), 179–187.

Thiele, L., Miettinen, K., Korhonen, P. J. & Molina, J. 2009. A preference-based evolutionary algorithm for multi-objective optimization. Evolutionary Computation 17 (3), 411–436.

Vapnik, V. 1999. The nature of statistical learning theory. Springer.

Wang, H., Olhofer, M. & Jin, Y. 2017. A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. Complex & Intelligent Systems 3, 233–245.

Wang, J., Liu, Y., Sun, J., Jiang, Y. & Sun, C. 2016. Diversified recommendation incorporating item content information based on MOEA/D. In 2016 49th Hawaii International Conference on System Sciences (HICSS). IEEE, 688–696.

Weistroffer, H. 1985. Careful usage of pessimistic values is needed in multiple objectives optimization. Operations Research Letters 4 (1), 23–25.

Wierzbicki, A. P. 1980. The use of reference objectives in multiobjective optimization. In G. Fandel & T. Gal (Eds.) Multiple Criteria Decision Making Theory and Application. Springer, 468–486.

Wierzbicki, A. P. 1982. A mathematical basis for satisficing decision making. Mathematical Modelling 3 (5), 391–405.

Xin, B., Chen, L., Chen, J., Ishibuchi, H., Hirota, K. & Liu, B. 2018. Interactive multiobjective optimization: a review of the state-of-the-art. IEEE Access 6, 41256–41279.

Yadav, D., Ramu, P. & Deb, K. 2023. Interpretable self-organizing map assisted interactive multi-criteria decision-making following Pareto-Race. Applied Soft Computing 149, 111032.

Yang, J.-B., Liu, J., Wang, J., Sii, H.-S. & Wang, H.-W. 2006. Belief rule-base inference methodology using the evidential reasoning approach-rimer. IEEE Transactions on systems, Man, and Cybernetics-part A: Systems and Humans 36 (2), 266–285.

Yang, L.-H., Liu, J., Ye, F.-F., Wang, Y.-M., Nugent, C., Wang, H. & Martínez, L. 2022. Highly explainable cumulative belief rule-based system with effective rule-base modeling and inference scheme. Knowledge-Based Systems 240, 107805.

You, Y., Sun, J., Guo, Y., Tan, Y. & Jiang, J. 2022. Interpretability and accuracy tradeoff in the modeling of belief rule-based systems. Knowledge-Based Systems 236, 107491.

Zhan, H. & Cao, Y. 2019. Relationship explainable multi-objective optimization via vector value function based reinforcement learning. https://doi.org/10.48550/arXiv.1910.01919.

Zhang, Q. & Li, H. 2007. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11 (6), 712–731.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N. & Zhang, Q. 2011. Multiobjective evolutionary algorithms: a survey of the state of the art. Swarm and Evolutionary Computation 1 (1), 32–49.

Zhou, Z.-J., Hu, G.-Y., Hu, C.-H., Wen, C.-L. & Chang, L.-L. 2019. A survey of belief rule-base expert system. IEEE Transactions on Systems, Man, and Cybernetics: Systems 51 (8), 4944–4958.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. & da Fonseca, V. 2003. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation 7 (2), 117–132.

Zitzler, E. & Künzli, S. 2004. Indicator-based selection in multiobjective search. In X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán & H.-P. Schwefel (Eds.) Parallel Problem Solving from Nature - PPSN VIII. Springer, 832–842.

ORIGINAL PAPERS

I

TOWARDS EXPLAINABLE INTERACTIVE MULTIOBJECTIVE OPTIMIZATION: R-XIMO

by

Misitano G., Afsar B., Lárraga G., Miettinen K. 2022

Autonomous Agents and Multi-Agent Systems, 36 (2), 43

Autonomous Agents and Multi-Agent Systems (2022) 36:43
https://doi.org/10.1007/s10458-022-09577-3

Towards explainable interactive multiobjective optimization: R-XIMO

Giovanni Misitano[1] · Bekir Afsar[1] · Giomara Lárraga[1] · Kaisa Miettinen[1]

Accepted: 6 July 2022 / Published online: 13 August 2022

Abstract
In interactive multiobjective optimization methods, the preferences of a decision maker are incorporated in a solution process to find solutions of interest for problems with multiple conflicting objectives. Since multiple solutions exist for these problems with various trade-offs, preferences are crucial to identify the best solution(s). However, it is not necessarily clear to the decision maker how the preferences lead to particular solutions and, by introducing explanations to interactive multiobjective optimization methods, we promote a novel paradigm of *explainable interactive multiobjective optimization*. As a proof of concept, we introduce a new method, *R-XIMO*, which provides explanations to a decision maker for reference point based interactive methods. We utilize concepts of explainable artificial intelligence and SHAP (Shapley Additive exPlanations) values. R-XIMO allows the decision maker to learn about the trade-offs in the underlying problem and promotes confidence in the solutions found. In particular, R-XIMO supports the decision maker in expressing new preferences that help them improve a desired objective by suggesting another objective to be impaired. This kind of support has been lacking. We validate R-XIMO numerically, with an illustrative example, and with a case study demonstrating how R-XIMO can support a real decision maker. Our results show that R-XIMO successfully generates sound explanations. Thus, incorporating explainability in interactive methods appears to be a very promising and exciting new research area.

Keywords Interactive methods · Multiple criteria optimization · Explainable artificial intelligence · Decision making · Reference point

Bekir Afsar, Giomara Lárraga and Kaisa Miettinen have contributed equally to this work.

Giovanni Misitano
giovanni.a.misitano@jyu.fi

Bekir Afsar
bekir.b.afsar@jyu.fi

Giomara Lárraga
giomara.g.larraga-maldonado@jyu.fi

Kaisa Miettinen
kaisa.miettinen@jyu.fi

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

1 Introduction

Real-life optimization problems seldom consist of only a single objective to be optimized. Instead, multiple conflicting objectives are to be considered simultaneously. These problems are known as *multiobjective optimization* problems and many solutions, known as *Pareto optimal solutions*, exist with various trade-offs between the objectives. The characteristics that define the best solution to be implemented in practice depend on the problem and subjective information. This information can be obtained from a human domain expert, known as a *decision maker* (DM). If the DM provides their preferences, we can find the DM's best (i.e., most preferred) solution.

The type of preferences a DM can provide varies a lot (see, e.g., [1–3]). When the preferences are incorporated into the solution process also matters. The DM can provide preferences before the optimization, but they can be too optimistic or pessimistic. Alternatively, a representative set of Pareto optimal solutions can be generated for the DM to choose from, but this can be both computationally and cognitively demanding. In contrast to these, in *interactive multiobjective optimization* methods [4, 5], preferences are incorporated iteratively during the solution process [1]. Interactive methods are many and vary in various aspects, such as the type of preference information required from the DM and how preferences are incorporated in the optimization process [4, 6, 7].

Moreover, the course of an interactive solution process can be divided into a *learning* and a *decision* phase [5]. Roughly speaking, as the name suggests, in the learning phase, the DM learns about the trade-offs and the feasibility of one's preferences to identify a region of interest and, in the decision phase, one converges to the most preferred solution in that region. Unfortunately, interactive methods typically offer little support to the DM during the learning phase making it hard for the DM to learn. This lack of support is an open issue in interactive multiobjective optimization [8, 9], which we will address in our work.

An example of preference information is a reference point consisting of desirable objective function values. We propose an approach to support the DM in applying interactive reference point based methods [10, 11], where *explanations* are provided to the DM about *why* an interactive method has mapped their preferences to certain solutions. Reference point based methods are classified, e.g., in [12], as ad hoc methods arguing that they do not support the DM in directing the solution process to provide preferences for the next iteration. Thus, these methods may seem like black-boxes to DMs. Therefore, explanations can help the DM learn about the trade-offs between the objectives in the problem, for instance. The general concept of an iteration of a reference point based interactive methods is illustrated in Fig. 1. The DM provides a reference point per iteration to get desirable values for objective functions. There are many ways a solution can be computed based on a reference point, e.g., by minimizing an appropriate scalarizing function that maps the reference point to the closest Pareto optimal solution. Thus, by modifying the reference point, different solutions can be found.

In addition, by utilizing the explanations, we can also support the DM by deriving *suggestions* from the explanations that provide information about *how* preferences can be modified to achieve some desired results, such as improving a certain objective function value in a solution of the next iteration. An example of the second iteration where we want to support the DM is illustrated in Fig. 1. There, the DM wishes to improve *Objective 2* in the initial solution and wonders how the initial reference point should be modified to achieve this goal. Indeed, according to the advice given in [13], we consider

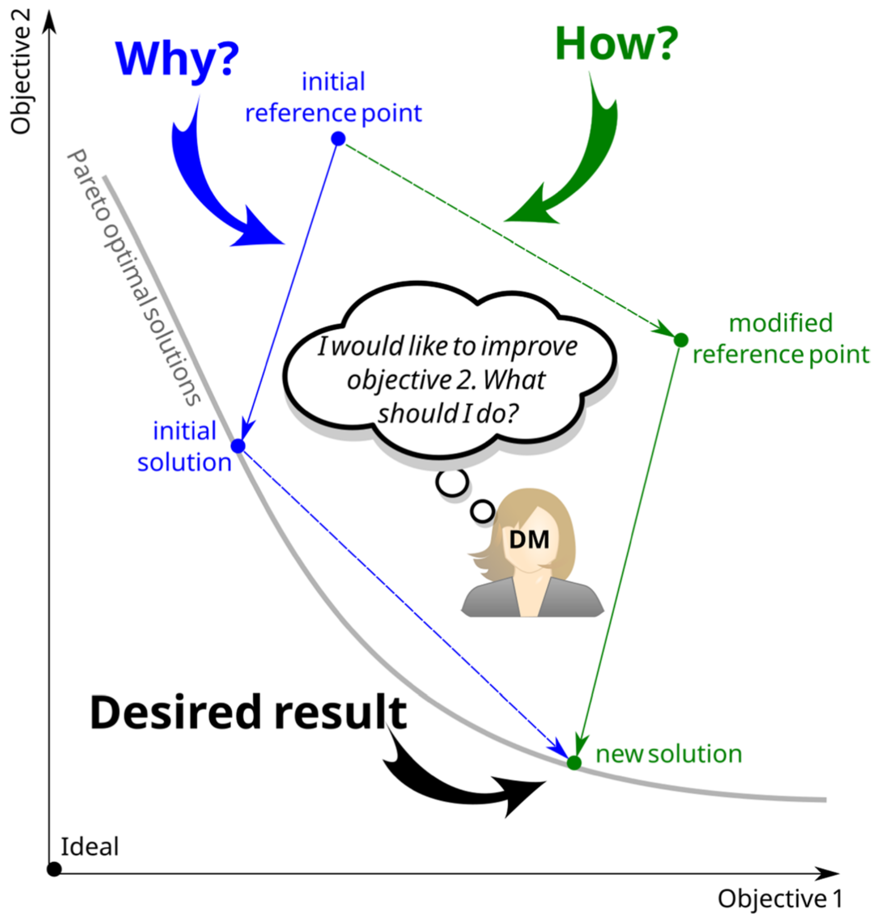


Fig. 1 The general concept of reference point based interactive multiobjective optimization methods illustrated with a problem with two objectives to be minimized. The questions of *why* a reference point has been mapped to a specific solution and *how* the reference point could be changed to achieve a desired result are highlighted

two central questions in interactive multiobjective optimization which can arise in the mind of the DM:

1. **Why** *preferences have been mapped to the computed solution(s)?*
2. **How** *can preferences be changed to affect the computed solution(s)?*

We borrow ideas from the field of *explainable artificial intelligence* (XAI) [14]. We do not attempt to create a new interactive multiobjective optimization method. Instead, we present a method that is able to explain the behavior of reference point based methods and support the DM in learning about the multiobjective optimization problem and providing preference information. There are methods in the field of XAI that can be used to formulate explanations for the predictions made by black-box machine learning models. Most of these methods have the advantage of being model agnostic, which means that they can be applied to any kind of (machine learning) model [15]. We show in our work that these methods can be applied in interactive multiobjective optimization methods as well and used to successfully formulate explanations.

Our main contribution is developing the concept of *explainable interactive multiobjective optimization* (XIMO) by exploring reference point based interactive multiobjective optimization methods. The ideas introduced in this paper are applicable to other interactive methods as well. XIMO is a very broad topic and our paper will, hopefully, lead to more follow-up research exploring the application of the concept of explainability in multiobjective optimization. Our proposed method, R-XIMO, derives explanations and supports a DM in providing a reference point to reflect desired changes in the objective functions. This method is also ideal to be incorporated as an agent in a multi-agent system supporting the DM in an interactive multiobjective solution process as discussed in [16].

Our paper is structured as follows. In Sect. 2, we introduce the background concepts required to understand the ideas discussed in the paper. Then, we introduce our proposed method R-XIMO in Sect. 3. In Sect. 4, we give an illustrative example on how R-XIMO can support a DM in practice, and we also present a case study with a real DM solving a multiobjective optimization problem in Finnish forest management. We validate R-XIMO further numerically and present the results in Sect. 5. We discuss the results of Sects. 4 and 5, as well as future research perspectives of R-XIMO, and XIMO in general, in Sect. 6. Lastly, we conclude our work in Sect. 7.

2 Background

2.1 Concepts of multiobjective optimization

Multiobjective optimization [1] consist of multiple conflicting objective functions to be optimized simultaneously. Such problems can be mathematically formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \mathbf{F}(\mathbf{x}) = \big(f_1(\mathbf{x}), \dots, f_k(\mathbf{x})\big) \\ \text{subject to} \quad & \mathbf{x} \in S, \end{aligned} \tag{1}$$

where $f_i(\mathbf{x})$, $i = 1, \dots, k$ are objective functions (with $k \geq 2$), and $\mathbf{x} = (x_1, ..., x_n)^T$ is a vector of n decision variables belonging to the feasible set $S \subset \mathbb{R}^n$. For every decision vector $\mathbf{x}$, there is a corresponding objective vector $\mathbf{F}(\mathbf{x})$. In the rest of this article, we refer only to minimization problems, but the conversion of a function to maximization is trivial (i.e., multiplying by -1).

Because of the conflict between the objective functions, not all of them can achieve their optimal values simultaneously. Given two feasible solutions $\mathbf{x}^1, \mathbf{x}^2 \in S$, $\mathbf{x}^1$ dominates $\mathbf{x}^2$ if and only if $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2)$ for all $i = 1, \dots k$, and $f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$ for at least one index $j = 1, \dots, k$. A solution $\mathbf{x}^* \in S$ is Pareto optimal if and only if there is no solution $\mathbf{x} \in S$ that dominates it. The set of all Pareto optimal solutions is called a Pareto optimal set, and the corresponding objective vectors constitute a Pareto optimal front. A feasible solution $\mathbf{x}^* \in S$ and the corresponding objective vector $\mathbf{F}(\mathbf{x}^*)$ in the objective space are weakly Pareto optimal if there does not exist another feasible solution $\mathbf{x} \in S$ such that $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for all $i = 1, \dots, k$.

The ideal point $\mathbf{z}^*$ and nadir point $\mathbf{z}^{nad}$ represent the lower and upper bounds of the objective function values among Pareto optimal solutions, respectively. The ideal point is calculated by minimizing each objective function separately. The nadir point represents the worst objective function values in the Pareto optimal set. Obtaining its value is not straightforward, as it requires computing the Pareto optimal set. However, it can be approximated [1]. The components of a utopian point $\mathbf{z}^{**}$ are derived by improving the components of the ideal point with a small positive ε.

As mentioned, typically, solving a multiobjective optimization problem involves a DM who has deeper knowledge of the problem. The DM is responsible for finding the most preferred solution among the conflicting objectives.

There are different types of methods for solving multiobjective optimization problems, for example, scalarization based and population based (like evolutionary) methods [17]. Scalarizing functions convert a multiobjective optimization problem into a single objective one [1, 11]. They usually also incorporate the preference information of the DM. Problem (1) can be converted into a scalarized one as

$$\begin{aligned} &\text{minimize} && s(\mathbf{F}(\mathbf{x}); \mathbf{p}) \\ &\text{subject to} && \mathbf{x} \in S, \end{aligned} \tag{2}$$

where $\mathbf{p}$ is a set of parameters required by the scalarizing function s. Several scalarizing functions have been proposed in the literature [1, 11]. We are interested in scalarizing functions [18] that consider a reference point $\bar{z}$ provided by the DM. As mentioned, a reference point consists of desirable objective function values, also known as aspiration levels. As examples, we utilize scalarizing function from different methods (for more information about reference point based scalarizing functions, see [10, 11]).

The scalarizing function of the GUESS method [19] is the following

$$\mathrm{GUESS}(\bar{z}; \mathbf{F}, \mathbf{z}^{\mathrm{nad}}) = \min_{\mathbf{x}\in S} \max_{i=1,\dots,k} \left[\frac{f_i(\mathbf{x}) - z_i^{\mathrm{nad}}}{z_i^{\mathrm{nad}} - \bar{z}_i} \right]. \tag{3}$$

From the STOM method [20], we get

$$\mathrm{STOM}(\bar{z}; \mathbf{F}, \mathbf{z}^{**}) = \min_{\mathbf{x}\in S} \max_{i=1,\dots,k} \left[\frac{f_i(\mathbf{x}) - z_i^{**}}{\bar{z}_i - z_i^{**}} \right] + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{\bar{z}_i - z_i^{**}}, \tag{4}$$

and from the reference point method (RPM) [18, 21] we get

$$\mathrm{RPM}(\bar{z}; \mathbf{F}, \mathbf{z}^{**}, \mathbf{z}^{\mathrm{nad}}) = \min_{\mathbf{x}\in S} \max_{i=1,\dots,k} \left[\frac{f_i(\mathbf{x}) - \bar{z}_i}{z_i^{\mathrm{nad}} - z_i^{**}} \right] + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{\mathrm{nad}} - z_i^{**}}. \tag{5}$$

Scalarizing functions (4) and (5) contain an augmentation term with a small, positive multiplier ρ. This term guarantees that the solution will not be weakly Pareto optimal, as can be the case for (3). Actually, the solutions of (4) and (5) are properly Pareto optimal (for further information, see [1]). For the three scalarizing functions, the denominator must not equal zero. In fact, it is positive when $z_i^* < \bar{z}_i < z_i^{\mathrm{nad}}$ for all $i = 1, \dots, k$.

As mentioned in the introduction, we consider reference point based methods, where in each iteration, the DM provides a reference point and the method generates one or some Pareto optimal solutions reflecting the preferences. Depending on the method, the scalarizing function used to generate the solution(s) varies (it can, e.g., be one of the three functions above). The DM can iteratively compare the obtained solutions and provide new reference points until the most preferred solution is found.

2.2 Explainable artificial intelligence and SHAP values

The central goal of machine learning methods [22] is to approximate, or *predict*, new information based on past observations. State-of-the-art machine learning methods, such as deep neural networks, have shown vast potential for various applications across many fields, see, e.g., [23–26]. It is typical for the most accurate machine learning models, which are often the most complex ones, to be also the most opaque [27], but not necessarily always [28]. These models are often employed in high-stakes domains, such as healthcare [29] and self-driving cars [30], where their opaque black-box nature can become problematic, see, e.g., [31, 32].

Because the true value of a prediction of a machine learning model is often unknown, the validity of the predictions cannot be checked by comparing it to the true value. Therefore, the viability of the prediction needs to be validated in some other way. An example is to provide some explanation justifying the prediction. Based on this explanation, a human, or humans, can then decide whether the prediction is sound or not.

XAI [14] sheds light on black-box models to understand how they make predictions. Many different XAI methods exist [33]. Usually, they try to explain the predictions made by black-box models, which have already been trained, as is done by LIME [34], for instance. The explanations are therefore not a result of the model itself, but an external tool. This kind of explanation is known as *post-hoc*. Another typical approach is to come up with new, inherently explainable, machine learning models, such as Bayesian rule lists [35]; or to simply tap into the explainability inherently found in interpretable models, such as decision trees [36]. Explanation models that do not depend on the type of machine learning model are known as *model agnostic* ones. Typically, these models can explain any machine learning model. For example, they are able to explain the prediction of an individual input for some previously trained model. And as we will later see in our work, some model agnostic explanation models can also be utilized to explain black-boxes that are not machine learning models at all. For reviews on the recent advancements in XAI, see, e.g., [15, 37].

Typically, a machine learning model g is trained on input–output training set pairs consisting of vectors with M features (also known as attributes) $\mathbf{a}$ and output values y. Training consists of finding internal parameter values for g so that when g is evaluated with some new observation $\mathbf{a}^*$, which was not present in the training set, the output of g, $g(\mathbf{a}^*) = y^*$, would be as close as possible to the true output value, i.e., $y^* \approx y^{\text{true}}$, which is often unknown. The output of the model g is also known as a *prediction*.

In our work, we focus on ad-hoc explanation methods unified by the SHAP framework [38]. The reason for this is that the SHAP framework guarantees certain theoretically sound properties (local accuracy, missingness, consistency, and uniqueness; see [38] for an in-depth discussion on their implications). By utilizing the SHAP framework, so-called SHAP values can be computed. SHAP values are based on Shapley values [39], which in turn are based on game theory [40].

Shapley values can be used to assign a value to the contribution of a single player to the payout in an n-player game. In other words, Shapley values can be used to characterize the contribution of a single entity (i.e., an attribute in an input to a machine learning model) when multiple entities collaborate to achieve a common goal (i.e., make a prediction). Thus, Shapley values can be used akin to sensitivity analysis to explore how a prediction made by a machine learning model changes when certain combinations of attributes are present or missing in the input, but with the added value of also having the four properties listed above. For instance, for some input $\mathbf{a}$ and prediction $g(\mathbf{a})$, a positive value for a Shapley value ϕ_i would indicate that the value of the attribute $a_i \in \mathbf{a}$ has overall contributed positively (i.e., increasingly) to the output value $g(\mathbf{a})$, and vice versa for a negative value for ϕ_i, and when ϕ_i is zero, attribute i has not contributed to the output value. With this kind of information, it is possible to come up with plausible explanations on how the machine learning method has made some particular prediction for a given input.

However, a typical machine learning model is not able to work with missing attributes; at least not without retraining the model, which in most cases can be very time-consuming. This makes Shapley values not directly applicable when generating explanations for some arbitrary machine learning model. That is why SHAP values are used, instead. In particular, kernel SHAP [38], which combines the idea behind Shapley values and LIME [34], is of particular interest because it is a model agnostic approach for computing SHAP values. Kernel SHAP requires so-called *missing data*, which is used to replace attributes in the input to a machine learning model to simulate missing attributes when explaining its predictions. In this way, the input to the machine learning model has always the same number of attributes, and the model does not have to be retrained when computing SHAP values. We use kernel SHAP to compute SHAP values in R-XIMO, proposed in Sect. 3, when it is validated in Sect. 5.

2.3 Explainability in multiobjective optimization

In what follows, we provide a brief literature review on explainable multiobjective optimization. The emphasis here is not on studies that use multiobjective optimization methods to generate explanations, but rather on studies that apply the existing explainable methods (or propose new ones) for multiobjective optimization.

A diversified recommendation framework based on a decomposition-based evolutionary algorithm was proposed in [9]. The authors modeled the recommender system as a multiobjective optimization problem and applied MOEA/D [41] to generate explainable recommendation lists for each user while maintaining a high recommendation accuracy.

A method explaining the reasoning behind the solution found for a multiobjective probabilistic planning problem was proposed in [42]. Their method generates verbal explanations about why it chose a specific solution among the other alternatives and also about the trade-off made between conflicting objectives in the final solution. Explaining trade-offs among various objectives was also studied in [43] via reinforcement learning by utilizing a correlation matrix that represents the relative importance between objectives.

There are some recent initiatives in the literature that incorporate explanations into interactive methods. For the sake of explainability, the interactive method called INFRINGER [44] utilized belief-rule-based systems to learn and model the DM's preferences. Similarly, in [45], the authors modeled the DM's preferences by using "if..., then..." decision rules, which were then used to explain the impact of the DM's preferences on the obtained solutions. They proposed a method called XIMEA-DRSA, which uses the decision rules as a preference model to guide the search in the solution process.

3 R-XIMO

In this section, we introduce the method proposed in this paper, R-XIMO, to explain how reference point based interactive multiobjective optimization methods map preference information into solutions. We start by describing the setting and general assumptions made in Sect. 3.1. In Sect. 3.2, we describe in detail how SHAP values are used to interpret a black-box that maps reference points to the Pareto optimal front. Finally, in Sect. 3.3, we discuss how the SHAP values are used to generate explanations and suggestions for a DM to allow them to make meaningful trade-offs regarding the preferences they have expressed.

3.1 Setting and assumptions

In general, a DM has domain expertise about the multiobjective optimization problem, allowing them to understand the existence of conflicts among the objectives (i.e., gaining in one objective in a Pareto optimal solution will result in a loss in at least one other objective). Assuming that a DM acts rationally (see, e.g., [46] for a discussion on rationality), they are only interested in Pareto optimal solutions. But a DM does not necessarily understand how the interactive method transforms the preference information into solution candidates during the solution process. According to these characteristics of DMs, we will assume that they perceive interactive multiobjective optimization methods as black-boxes.

Let us consider black-boxes mapping reference points $\bar{z}$ to objective vectors $\mathbf{z}$ on the Pareto optimal front for a problem (1) with k objectives. We define such a black-box as

$$\mathfrak{B}(\bar{z}) : \mathbb{R}^k \rightarrow \mathbb{R}^k_{\text{Pareto}}, \tag{6}$$

where the subterm *Pareto* means that the objective vectors and the reference points are mapped to solutions that lie on the Pareto optimal front.

In particular, we use black-boxes, which minimize reference point based scalarizing functions [1, 11]. As mentioned, as examples, we consider the scalarizing functions (3), (4) and (5), and the DM provides preferences as a reference point. We assume that the DM is informed of the values for the ideal and nadir points when providing reference points, as it was originally assumed in [21]. This will allow the DM to provide more realistic reference points. Depending on the type of black-box (6) considered, it may also be necessary to assume that each aspiration level in the reference point is between the objective's respective components in the ideal and nadir points. Lastly, we assume the DM to be interacting with an interactive method that acts like the black-box defined in (6) over the course of a few iterations until they find a most preferred solution.

3.2 Using SHAP values to explain reference point based black-box models

The idea behind SHAP values discussed in Sect. 2.2 can also be applied to other types of black-boxes, not necessarily related to machine learning models. We apply the idea to an interactive multiobjective optimization method explaining its behavior to a DM. We limit the discussion to a simple case of an interactive method (6), where a DM is only required to provide a reference point over the course of a few iterations. Then, the input to the model is the reference point provided by the DM and the prediction is the result of solving problem (2) with some scalarizing function s.

We can use SHAP values to formulate explanations for models adherent with (6). Since the reference point provided by a DM and the output of (6) have $k \geq 2$ dimensions, the SHAP values computed are represented by a $k \times k$ square matrix Φ with elements ϕ_{ij}, $i, j = 1, \ldots, k$:

$$\Phi = \begin{pmatrix} \phi_{11}, & \phi_{12}, & \dots, & \phi_{1k} \\ \phi_{21}, & \phi_{22}, & \dots, & \phi_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{k1}, & \phi_{k2}, & \dots, & \phi_{kk} \end{pmatrix}. \tag{7}$$

The *average effect* of a reference point on a solution is represented by (7). How the ith component in the resulting objective vector has been affected by the jth component in the reference point, is represented by the value of the element ϕ_{ij} in (7). Thus, the SHAP values in (7) can be used to induce how, on average, the input $\bar{z}$ has affected the output $\mathbf{z}$. Expanding on the discussion given for the interpretation of Shapley values in Sect. 2.2, a positive value for ϕ_{ij} means that on average, the jth component in the reference point had an increasing effect on the value of objective i in the solution, and vice versa for negative values. A value of zero for ϕ_{ij} means that there was no effect between the two. When objectives are minimized, an increasing effect means *impairing*, and a decreasing effect means *improving*.

We know that in multiobjective optimization, the objectives are conflicting. Therefore, we can say that when two objectives have an increasing effect on each other (i.e., both ϕ_{ij} and ϕ_{ji} are positive for some i, j) the aspiration levels set by the DM in the reference point are not simultaneously achievable on the Pareto optimal front. This is because in the specific region of the front the reference point was mapped to, objectives i and j are conflicting. Note that in case $i = j$, the value of ϕ_{ij} is understood as the objective's effect upon itself. It is important to mention that not all objectives need be always conflicting, and that the conflict can be between more than two objectives, but in our work, we only consider conflicts between pairs of objectives for the sake of simplicity. Therefore, the exact conflicting nature of two considered objectives depends on which region of the Pareto optimal front is observed.

Because of the properties mentioned in Sect. 2 for SHAP values, we know that the values are local and unique. Especially, the local nature of the SHAP values is important because it guarantees that the SHAP values computed describe the conflicts among objectives in a local region of the Pareto optimal front. Moreover, a direct consequence of the uniqueness of the SHAP values warrants that the explanations derived from the SHAP values can be assumed to be unique (albeit the way the values are interpreted can lead to different explanations, but any reasoning based solely of the actual numerical values should lead to the same conclusions). This is why we have decided in our work to utilize SHAP values.

We use SHAP values (7) to deduce how a component in a given reference point affects the solution computed by a black-box (6). Particularly, we can gather information about the conflict between two objectives. We can then communicate this information to the DM giving them support in formulating new reference points. Thus, an explainable support system can be created to support the DM in achieving their goals in an interactive solution process utilizing SHAP values. How this kind of system can be realized, is discussed in the next subsection.

3.3 Utilizing explanations and suggestions to aid a decision maker

We choose to demonstrate the plausibility of utilizing SHAP values for explaining interactive multiobjective optimization methods with a simple application as follows. Consider a DM has provided a reference point $\bar{z}$ to a black-box (6) and has been presented with a

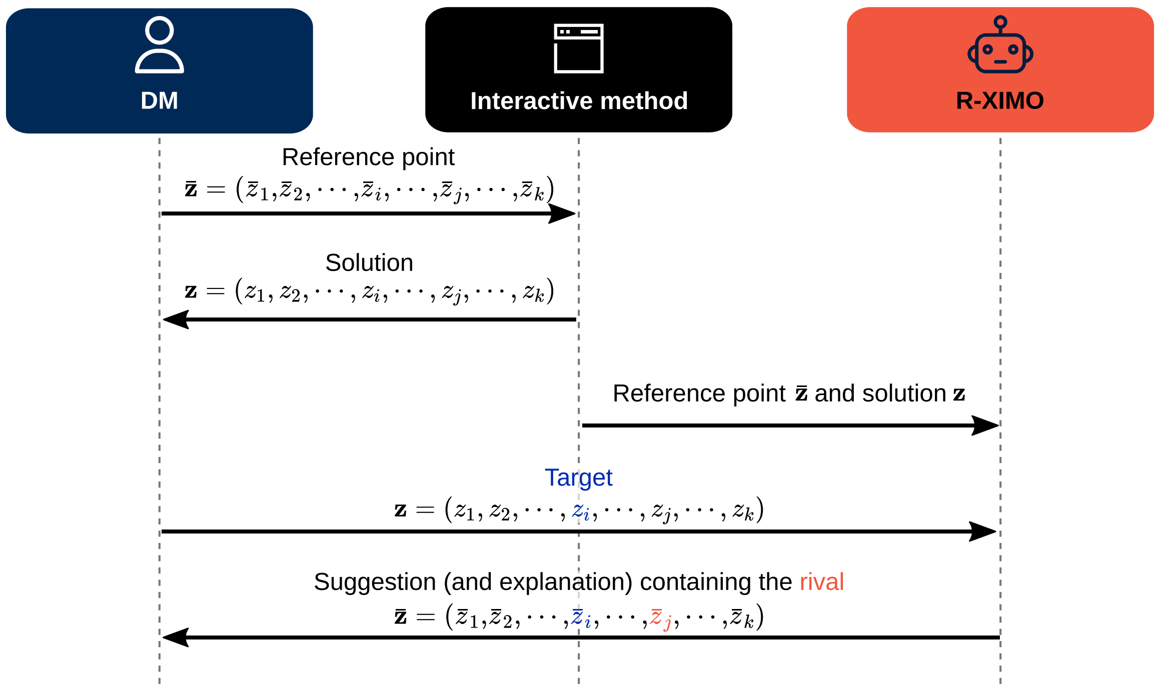

DM
Interactive method
R-XIMO
Reference point
$\bar{\mathbf{z}} = (\bar{z}_1, \bar{z}_2, \cdots, \bar{z}_i, \cdots, \bar{z}_j, \cdots, \bar{z}_k)$
Solution
$\mathbf{z} = (z_1, z_2, \cdots, z_i, \cdots, z_j, \cdots, z_k)$
Reference point $\bar{\mathbf{z}}$ and solution $\mathbf{z}$
Target
$\mathbf{z} = (z_1, z_2, \cdots, z_i, \cdots, z_j, \cdots, z_k)$
Suggestion (and explanation) containing the rival
$\bar{\mathbf{z}} = (\bar{z}_1, \bar{z}_2, \cdots, \bar{z}_i, \cdots, \bar{z}_j, \cdots, \bar{z}_k)$

Fig. 2 Illustration on how R-XIMO interacts with the interactive method and the DM. R-XIMO is aware of both the reference point provided by the DM and the solution computed by the interactive method. After the DM selects a target, R-XIMO can provide a suggestion and explanation with information on the rival. In the figure, a single iteration of an interactive method combined with R-XIMO is depicted

solution $\mathbf{z}$. Now the DM wishes to see an improvement in the value of the ith objective in $\mathbf{z}$. We designate this objective as the *target* and define its index as i_{target}. We can then use the computed SHAP values to find the component in $\bar{\mathbf{z}}$, which had the most impairing effect on the target in $\mathbf{z}$ (i.e., $\phi_{i_{\text{target}}j} = \max_{\phi_{ij}, i=i_{\text{target}}} \Phi$). We name this objective with the most impairing effect as the *rival* with index j_{rival}. Therefore, we can formulate explanations for the DM on how the solution $\mathbf{z}$ relates to the given reference point $\bar{z}$ from the perspective of the target, and how the DM can change the reference point for the next iteration to achieve a better value in the target.

The general idea of our proposed method is depicted in Fig. 2. Since our method enhances **r**eference point based interactive methods with explanations, we call it R-XIMO.

The details of the procedure to compute the rival and generate an explanation in R-XIMO are given in Algorithm 1. The input to Algorithm 1 are the black-box $\mathfrak{B}$ (6), a reference point $\bar{z}$, the solution $\mathbf{z}$ computed utilizing the black-box and the reference point, missing data Z_{missing} needed in computing SHAP values, and the index of the target objective i_{target} provided by the DM. The missing data is used by the routine `shap_values` in Algorithm 1 to calculate the SHAP values. The routine `shap_values` can be any routine able to compute SHAP values like in (7) (e.g., kernel SHAP). The output of Algorithm 1 is the index of the rival objective j_{rival} and an explanation `explanation` on how the reference point $\bar{z}$ given has affected the solution $\mathbf{z}$ computed.

When choosing the missing data Z_{missing} to be used in R-XIMO, it is important that such data is available in the vicinity of the reference point being explained to assure the locality of the explanations. Therefore, missing data should be generated as evenly as possible in the domain space of (6), but since we assume a DM to provide reference points with component values bounded by the respective components of the ideal and nadir points, it is enough for the generated missing data to be bound in a similar way. However, when experimenting, we found that we could use a representation of the Pareto optimal front of the original multiobjective optimization problem as the missing data without any loss in performance of R-XIMO. This, we believe, is because the Pareto optimal front characterizes the trade-offs among the objectives in the problem, which is what we are primarily interested in.

For computing the index of the rival j_{rival} in Algorithm 1, the general idea is to find the element $\phi_{i_{\text{target}}j} \in \Phi$ with the largest positive value. If this value exists and it is not the target itself, then the index j of the element found is defined as `worst_effect`. Otherwise `worst_effect` is set to be -1 indicating that it does not exist. Likewise, we can also find the element $\phi_{i_{\text{target}}j}$ with the smallest negative value and define its j index as `best_effect`. The routine `why_objective_i` in Algorithm 1 computes both of these values. In cases where `worst_effect` does not exist, we can find the element $\phi_{i_{\text{target}}j}$ with the largest negative value and set its j index as `least_negative` as is done in Algorithm 1. Lastly, if `worst_effect` $= i_{\text{target}}$, we can find the element $\phi_{i_{\text{target}}j}$ with the second largest value and define `second_worst` to be equal to j. The value of j_{rival} returned by Algorithm 1 is therefore always either `worst_effect`, `second_worst`, or `least_negative`. An implementation of Algorithm 1 is discussed in Sect. 5.1.

Algorithm 1 SHAP-XIMO: deducing j_{rival} based on SHAP values computed for a black-box (6). The explanations referred to in the algorithm (`explanation_1 - explanation_9`) are given in Table 1.

```
Input: $\mathfrak{B}, \bar{\mathbf{z}}, \mathbf{z}, Z_{\text{missing}}, i_{\text{target}}$
Output: $j_{\text{rival}}$, explanation
 1: $\Phi \leftarrow$ shap_values($\mathfrak{B}, \bar{\mathbf{z}}, Z_{\text{missing}}$)
 2: worst_effect, best_effect $\leftarrow$ why_objective_i($\Phi, i_{\text{target}}$)
 3: if nothing has improved in $\mathbf{z}$ compared to $\bar{\mathbf{z}}$ then
 4:     if worst_effect$\neq i_{\text{target}}$ then
 5:         $j_{\text{rival}}$, explanation $\leftarrow$ worst_effect, explanation_1
 6:     else
 7:         second_worst $\leftarrow \arg\max_{j\in[1,k]\setminus i_{\text{target}}}(\Phi_{i_{\text{target}},j})$
 8:         $j_{\text{rival}}$, explanation $\leftarrow$ second_worst, explanation_2
 9:     end if
10: else if everything has improved in $\mathbf{z}$ compared to $\bar{\mathbf{z}}$ then
11:     worst_effect $\leftarrow \arg\max_{j\in[1,k]}(\Phi_{i_{\text{target}},j})$
12:     if $i_{\text{target}} =$ worst_effect then
13:         second_worst $\leftarrow \arg\max_{j\in[1,k]\setminus i_{\text{target}}}(\Phi_{i_{\text{target}},j})$
14:         $j_{\text{rival}}$, explanation $\leftarrow$ second_worst, explanation_3
15:     else
16:         $j_{\text{rival}}$, explanation $\leftarrow$ worst_effect, explanation_4
17:     end if
18: else if $i_{\text{target}} \neq$ best_effect and $i_{\text{target}} \neq$ worst_effect then
19:     if best_effect $= -1$ then        ▷ -1 means that best_effect was not found
20:         $j_{\text{rival}}$, explanation $\leftarrow$ worst_effect, explanation_5
21:     else if worst_effect $= -1$ then
22:         least_negative $\leftarrow \arg\max_{j\in[1,k]\setminus i_{\text{target}}}(\Phi_{i_{\text{target}},j})$
23:         $j_{\text{rival}}$, explanation $\leftarrow$ least_negative, explanation_6
24:     else
25:         $j_{\text{rival}}$, explanation $\leftarrow$ worst_effect, explanation_7
26:     end if
27: else if $i_{\text{target}} =$ worst_effect then
28:     second_worst $\leftarrow \arg\max_{j\in[1,k]\setminus i_{\text{target}}}(\Phi_{i_{\text{target}},j})$
29:     $j_{\text{rival}}$, explanation $\leftarrow$ second_worst, explanation_8
30: else if worst_effect = -1 then
31:     least_negative $\leftarrow \arg\max_{j\in[1,k]\setminus i_{\text{target}}}(\Phi_{i_{\text{target}},j})$
32:     $j_{\text{rival}}$, explanation $\leftarrow$ least_negative, explanation_6
33: else ($i_{\text{target}} =$ best_effect)
34:     $j_{\text{rival}}$, explanation $\leftarrow$ worst_effect, explanation_9
35: end if
36: return $j_{\text{rival}}$, explanation
```

The nine possible explanations (indexed by $\mathtt{n} = 1, \ldots, 9$) returned by Algorithm 1 are listed in Table 1. From each of the explanations, a *suggestion* is derived to support the DM in achieving their goal of improving the value of the target in the solution. The explanations tell the DM how the given reference point $\bar{\mathbf{z}}$ is related to the solution $\mathbf{z}$, and how the components of $\bar{\mathbf{z}}$ have affected the value of the target in $\mathbf{z}$. In supporting the DM, the suggestion derived from the explanation is most relevant. However, the explanation can help

Table 1 Explanations `explanation_n` returned by Algorithm 1 for indices $n = 1, \ldots, 9$

Index `n`	Explanation part	Suggestion part
1	Each objective value in the solution is worse when compared to the reference point. The reference point was too demanding. The component `worst_effect` in the reference point had the most impairing effect on objective i_{target} in the solution.	Try improving the component i_{target} and impairing the component `worst_effect`
2	Each objective value in the solution was worse when compared to the reference point. The reference point was too demanding. The component i_{target} in the reference point had the most impairing effect on objective i_{target} in the solution. The component `second_worst` had the second most impairing effect on objective i_{target}.	Try improving the component i_{target} and impairing the component `second_worst`
3	Each objective value in the solution had a better value when compared to the reference point. The reference point was pessimistic. The component i_{target} in the reference point had the least improving effect on objective i_{target} in the solution. The component `second_worst` had the second least improving effect on the objective i_{target}.	Try improving the component i_{target} and impairing the component `second_worst`
4	Each objective value in the solution had a better value when compared to the reference point. The reference point was pessimistic. The component `worst_effect` in the reference point had the least improving effect on the objective i_{target} in the solution.	Try improving the component i_{target} and impairing the component `worst_effect`
5	None of the components in the reference point had an improving effect on the objective $i_{\texttt{target}}$ in the solution. The component `worst_effect` in the reference point had the most impairing effect on objective i_{target} in the solution.	Try improving the component i_{target} and impairing the component `worst_effect`
6	None of the objectives in the reference point had an impairing effect on objective i_{target} in the solution. Objective `least_negative` in the reference point had the least improving effect on objective i_{target} in the solution.	Try improving the component i_{target} and impairing the component `least_negative`
7	The objective i_{target} was most improved in the solution by the component `best_effect` and most impaired by the component `worst_effect` in the reference point.	Try improving the component i_{target} and impairing the component `worst_effect`
8	The objective i_{target} was most impaired in the solution by its component in the reference point. The component `second_worst` had the second most impairing effect on the objective i_{target}.	Try improving the component i_{target} and impairing the component `second_worst`

Table 1 (continued)

Index `n`	Explanation part	Suggestion part
`9`	The objective i_{target} was most improved in the solution by its component in the reference point. The component `worst_effect` had the most impairing effect on objective i_{target}.	Try improving the component i_{target} and impairing the component `worst_effect`

From each explanation, a suggestion is derived. When the suggestion is followed, it leads to the improvement of the value of objective i_{target} in the solution. How these explanations are communicated to the DM exactly can vary (e.g., based on the problem and general setting of the solution process) the DM gain additional insight related to the multiobjective optimization problem, and it can help the DM build confidence in the suggestion given as well. Therefore, in practice, the explanation should be shown to the DM only when they request to see it. The suggestion should be always provided to the DM. Examples of utilizing R-XIMO are given in Sect. 4.

The first four explanations in Table 1 are relevant in cases where the components of the given reference point are either worse in regard to every objective value when compared to the solution ($\mathtt{n} = 1, 2$), or the components in the reference point are all better than in the solution ($\mathtt{n} = 3, 4$). Such reference points can be expected to arise when the DM is still in an early stage of the interactive solution process and is therefore still learning about the problem. In case of $\mathtt{n} = 2$, the suggestion still prompts the DM to improve the target component in the reference point despite the target having the most impairing effect on the target objective in the solution. This can feel counter intuitive, but it is done because worsening the rival component in the reference point can lead to a situation where some other objective than the target improves, if the target component is left unchanged. By still improving the target component in the reference point, we try to guarantee that the DM will see an improvement in the target objective in the solution.

The following two explanations in Table 1 arise when none of the components in the reference point had an improving effect on the target ($\mathtt{n} = 5$), or when none of the components had an impairing effect on the target ($\mathtt{n} = 6$). In the first case, the DM may want to be careful when improving the value of the target in the next reference point since in the area of the Pareto optimal front the solution resides, the other objectives seem to be all in conflict with the target objective. In the second case, the DM may want to experiment with improving the value of the target objective in the next reference point since none of the other objectives had any impairing effects on the target.

The seventh explanation in Table 1 ($\mathtt{n} = 7$) is the explanation that one can expect to arise in most cases after the DM has gained some insight about the problem and its trade-offs. In this case, some component in the reference point had an improving effect on the target objective in the solution and some other component had an impairing effect. In this case, neither `best_effect` nor `worst_effect` is the target objective.

The last two explanations in Table 1 ($\mathtt{n} = 8, 9$) arise when the condition of the first four explanations are not met and the most impairing or improving effect on the target objective's value in the solution was due to the target objective's component in the reference point. The eighth explanation ($\mathtt{n} = 8$) is something the DM does not probably desire to see when they care about the target objective, and could therefore be a reason for the DM to mistrust the interactive method. On the other hand, the last explanation ($\mathtt{n} = 9$) is probably the one a DM would expect to see as they deem the target objective to be the most important, when providing a reference point.

We can think of impairing a component of the reference point as a way to gain more room in terms of improving some other objective value in the solution. This is why the suggestion in Table 1 always prompts the DM to improve the target component in the reference point. In this way, we can assume that the room gained in impairing the rival is reflected in the improvement on the target. We can justify this on basis of the objectives being in conflict in multiobjective optimization problems.

Validating the explanations in Table 1 (i.e., how useful the explanation and suggestion is to a human DM) is impossible without either human participants or advanced artificial DMs. To our knowledge, no artificial DMs exist that could help validate the explanations. In Sect. 4.1 we provide an illustrative example how the explanations (and suggestions) generated by R-XIMO, can support a hypothetical DM. In Sect. 4.2, we demonstrate R-XIMO in a case study with a real DM. We also validate the suggestions derived from the explanations in Sect. 5 numerically—i.e., does improving the target and impairing the rival computed by R-XIMO lead to an improvement in the value of the target in the solution? As we will see, the suggestions generated by R-XIMO can reveal to the DM the best component to be impaired in a reference point when a given target objective is to be improved. This alone can be very valuable information to a DM.

4 Example and case study

In this section, we show how R-XIMO can be applied in solving multiobjective optimization problems interactively. We demonstrate this both with an illustrative example in Sect. 4.1, and a case study involving a real DM in Sect. 4.2.

4.1 Illustrative example

In this subsection, we demonstrate with an example how R-XIMO supports a DM by providing explanations and suggestions (Table 1) in an interactive solution process. An analyst (one of the authors) acted as the DM to illustrate the support R-XIMO provides in solving a real-world multiobjective optimization problem. The problem considered was originally proposed in [47] and modified in [48]. A Python notebook with the described solution process is available online.[1]

4.1.1 Problem description

The problem describes a (hypothetical) pollution of a river. There is a fishery company and a city in a valley along the river. The company is located near the head of the valley, and it causes industrial pollution on the river. The city is located downstream from the fishery and is the source of municipal waste pollution on the river. Water quality is measured in terms of dissolved oxygen level (DO), while industrial and municipal pollution is quantified in pounds of biochemical oxygen demanding material (BOD). There are some existing treatment facilities that reduce the BOD in the water, and their costs are paid by the company and the city. To deal with the water pollution, additional water treatment facilities should be built, which would incur higher costs, raising the city's tax rate and decreasing the company's return on investment.

The two decision variables, x_1 and x_2, control the amount of BOD removed from water in two treatment plans located in the company and in the city, respectively. The original problem had four objectives; f_1 maximizing DO in the city, f_2 maximizing DO at the state line downstream from the city, f_3 maximizing percent return on investment at the company, and f_4 minimizing the additional tax rate in the city. We use the modified version of the problem [48], in which the fifth objective (f_5) is added to describe the functionality of the treatment facilities. Thus, the multiobjective optimization problem has five objectives and two decision variables (we consider it as a minimization problem by multiplying the first three objectives by -1), as follows:

[1] https://github.com/gialmisi/shap-experiments/blob/d7ac397c8b2e76bea3a083b68dae6636abd03ff4/notebooks/river_pollution.ipynb

$$\begin{aligned}
\text{minimize} \quad & f_1(\mathbf{x}) = -4.07 - 2.27x_1 \\
\text{minimize} \quad & f_2(\mathbf{x}) = -2.60 - 0.03x_1 - 0.02x_2 \\
& \qquad - \tfrac{0.01}{1.39-x_1^2} - \tfrac{0.30}{1.39-x_2^2} \\
\text{minimize} \quad & f_3(\mathbf{x}) = -8.21 + \tfrac{0.71}{1.09-x_1^2} \\
\text{minimize} \quad & f_4(\mathbf{x}) = -0.96 + \tfrac{0.96}{1.09-x_2^2} \\
\text{minimize} \quad & f_5(\mathbf{x}) = \max\{|x_1 - 0.65|, |x_2 - 0.65|\} \\
\text{subject to} \quad & 0.3 \le x_1, x_2 \le 1.0.
\end{aligned} \tag{8}$$

4.1.2 Solution process

We can now describe the interactive solution process using R-XIMO with a DM. To scalarize (8), we used STOM (4) and an approximation of the Pareto optimal front of (8) computed utilizing evolutionary methods (NSGA-III [49], MOEA/D [41], and RVEA [50]). The scalarized version of (8) was solved by finding the objective vector that minimizes (2) in the Pareto optimal front. At the beginning of the solution process, the ideal $(-6.34, -3.44, -7.5, 0, 0)$ and nadir $(-4.75, -2.85, -0.32, 9.70, 0.35)$ points were calculated based on the approximation of the Pareto optimal front and shown to the DM.

Iteration 1. First, the DM set the ideal point as the reference point to see how difficult it is to achieve these promising values. The obtained result was $(-5.75, -2.91, -6.91, 0.20, 0.13)$. The DM desired to improve the water quality in the city (f_1) and R-XIMO returned the following suggestion: " *Try improving the 1st component and impairing the 3rd component.*"

Iteration 2. Since the reference point had been too optimistic, and the DM realized that to improve f_1, he needed to impair f_3 (the return on investments). Therefore, he adjusted all aspiration levels accordingly but most impairments were made in the 3rd one, and he set the next reference point as $(-6.00, -3.20, -6.00, 0.10, 0.10)$. As a consequence, the following solution was obtained: $(-6.00, -2.92, -6.26, 0.21, 0.20)$. He was happy with the return on investments (f_3), the addition to the tax (f_4), and the efficiency of the treatment facilities (f_5). However, the water quality after the city (f_2) was inadequate, so he wanted to improve that objective with the support of R-XIMO, which made the following suggestion: " *Try improving the 2nd component and impairing the 4th component.* "

Iteration 3. Based on the given suggestion, the DM realized the trade-off between f_2 and f_4. He followed the suggestion and impaired the 4th aspiration level, set the reference point $(-6.00, -3.20, -6.00, 1.00, 0.10)$ and obtained the corresponding solution $(-5.90, -3.06, -6.60, 1.21, 0.16)$. There was a good improvement on f_2, but the DM wished to improve it even further, if possible. R-XIMO provided the following suggestion in response to the DM's request of improving the value of f_2: " *Try improving the 2nd component and impairing the 5th component.*"

Iteration 4. To improve f_2, the DM needed to impair the aspiration level for f_5 and kept the same aspiration levels for the other objectives as in the previous reference point: $(-6.00, -3.20, -6.00, 1.00, 0.20)$. As a consequence, the following solution was obtained: $(-6.09, -3.09, -5.79, 1.44, 0.24)$. As can be observed, the water quality in and after the city (f_1 and f_2) improved, while the economic objectives (f_3 and f_4) and facility efficiency (f_5) deteriorated. The DM was not satisfied with the last three objectives, particularly the last one. He wished to improve it, and the following suggestion was made to achieve his purpose: " *Try improving the 5th component and impairing the 3rd component.*"

Iteration 5. Therefore, he reduced his economic expectations (f_3) and improved the efficiency (f_5) in his reference point: $(-6.00, -3.20, -5.50, 1.00, 0.12)$. The DM was almost happy with the returned solution $(-5.94, -3.08, -6.49, 1.38, 0.17)$ since he nearly obtained what he desired without sacrificing the third objective. However, he wanted to ensure that the addition to the tax rate (f_4) could be decreased without jeopardizing other objectives. To understand whether this is possible, the DM requested an explanation in addition to the suggestion for improving the tax rate. R-XIMO returned the following: " *None of the components in the reference point had an impairing effect on objective f_4 in the solution. The 1st component of the reference point had the least improving effect on objective f_4 in the solution. Try improving the 4th component and impairing the 1st component.*"

Iteration 6. The DM improved his aspiration level for the fourth objective based on the suggestion and kept the others the same as before: $(-6.00, -3.20, -5.50, 0.80, 0.12)$, because none of the components had an impairing effect on objective f_4 based on the given explanation. The solution obtained was $(-5.95, -3.06, -6.45, 1.25, 0.18)$. As can be seen, the return on investments (f_3) was relatively higher than his aspiration level for that objective, he obtained sufficient water quality for the city (f_1), and after the city (f_3), the addition to tax (f_4) was slightly improved from the previous solution, and the efficiency of the facilities was nearly identical. The DM was satisfied with this solution and decided to stop the solution process.

4.1.3 Observations

Clearly, the suggestions made by R-XIMO assisted the DM in recognizing the trade-offs among the objectives and efficiently providing his preference information to get more preferred solutions. Having the option to request an explanation was also beneficial to the DM; for example, in iteration 5, the DM benefited from the explanation provided by R-XIMO. At that point, the DM gained sufficient insight into the problem and was mostly aware of the existing conflicts among the objectives. He was almost satisfied but wanted to improve one specific objective further, if possible. That is why he requested an explanation from R-XIMO whether he missed some other existing conflicts or not. Based on the given explanation, he understood that there were no other objectives impairing his target objective. Therefore, he followed the first part of the suggestion (improving the target objective) but not the second part, which suggested impairing some other objective having the least improving effect on the target objective (which he learned from the explanation). As experienced, the DM was not forced to follow the suggestions but benefited from the explanations.

4.2 Real case study

As a proof of concept, we consider a multiobjective optimization problem with a domain expert as the DM in a case study in Finnish forest management. We first briefly outline the problem and then describe the setting and solution process with the DM. We also report the DM's opinions and feedback regarding R-XIMO and the support it provides.

4.2.1 Problem description

Finnish forests are divided into managerial areas known as stands. In a forest management problem, for each stand, a particular management strategy is to be chosen to be employed over a certain time period. Some examples of available strategies are, for instance, that trees in a stand are cut down or thinned out, or the stand is left untouched. Depending on which strategy is employed for a stand, corresponding consequences will ensue. These consequences can be regarded as objectives, and by considering multiple consequences at the same time, the forest management problem can be modeled as a multiobjective optimization problem.

In our case, we have three objectives to be maximized simultaneously over the considered time period: income from sold timber (*Income*), carbon dioxide stored in the trees ($\mathit{Stored\ CO_2}$), and the combined habitat suitability index indicating how habitable the forest is for fauna (*CSHI*). Solutions to the problem will be represented by objective vectors of the form (Income, $\mathit{Stored\ CO_2}$, CSHI). These objectives are in conflict; for instance, cutting down trees and selling the timber for increased profit will release stored carbon dioxide and make the stand inhabitable for the fauna; or thinning out a stand can increase its combined suitable habitat index, but it will also release stored carbon, and it can be financially unprofitable; or leaving the stand as it is will maximize the stored carbon dioxide and provide zero income.

The objective values for each stand in the considered forest (consisting of multiple stands) are aggregated, which means that the objectives represent the whole forest instead of single stands. Therefore, a solution to the multiobjective optimization problem consists of choosing a managerial strategy for each individual stand, and then summing each objective over all available stands, i.e., for the whole forest. We have computed a representative set of Pareto optimal solutions based on simulated data. For details on the problem and how the solutions have been generated, see Chapter 5 in [51] and [44]. The representation of the Pareto front used in the case study is available online.[2]

4.2.2 Setting

The forest management problem was solved utilizing a simple interactive method, where the DM provides a reference point in each iteration. R-XIMO was used to generate suggestions and explanations. Based on the reference point, a new solution was then computed utilizing a scalarizing function (5). Prior to the experiment, the DM was already familiar with this kind of interactive multiobjective optimization process. Before the solution process, the DM was informed about the support R-XIMO offers, namely, that after a solution is computed based on a provided reference point, he may express whether he would like to improve any of the objective function values computed based on the reference point. Before starting to solve the problem, the DM was asked whether he would like to see the explanations generated by R-XIMO in addition to the suggestions, to which he agreed.

Overall, the forest management problem was solved twice by the DM (with different strategies behind the preferences). All the information related to the solution processes shown to the DM was in textual or tabulated formats. No visualizations were used. After

[2] https://github.com/gialmisi/shap-experiments/blob/c3c66df02f1c5d7ef994a3d5dce00b17ede4724c/data/forest.csv

Table 2 The solutions and reference points of the first solution process

	Ideal point			Nadir point		
	Income	Stored CO_2	CSHI	Income	Stored CO_2	CSHI
	6.285	8.269	3.244	1.877	6.733	2.139
	Current solution			Reference point		
Iteration	Income	Stored CO_2	CSHI	Income	Stored CO_2	CSHI
1	–	–	–	4.500	7.750	2.800
2	4.599	7.833	2.823	4.500	7.650	2.800
3	4.657	7.705	2.872	4.400	7.705	2.800
4	4.613	7.772	2.854	–	–	–

The Incomes shown are scaled down by a factor of 10e-7, the Stored CO_2 by a factor of 10e-9, and the CSHI values by a factor of 10e-4. The ideal and nadir points of the representative set of Pareto optimal solutions considered are also shown

the two solution processes, the DM was asked some additional questions. In what follows, we describe the two solution processes, followed by the answers to the presented questions and some general observations. Two Python notebooks are available online with the contents of the two optimization processes described next.[3,4]

4.2.3 First solution process

Iteration 1. First, the DM was shown the ideal and nadir points shown in Table 2. With the first reference point, the DM wished to achieve a solution with a moderate amount of income and a moderate CSHI, with “quite a bit” of stored carbon dioxide. This reference point is shown in Table 2.

Iteration 2. The solution with the objective function values shown in Table 2 was computed based on the reference point given in the first iteration. The first thing the DM noted was how close the objective function values were to the reference point given. He then wished to improve either the stored carbon dioxide or the CSHI value by lowering the income. He decided that he would like to improve the CHSI value. Therefore, CHSI was chosen as the target in R-XIMO, which produced the following suggestion: *Try improving the CSHI and impairing the Stored* CO_2. In formulating a new reference point, the DM did not, however, wish to improve CHSI any further. The reference point given by the DM in the second iteration is shown in Table 2.

Iteration 3. After seeing the newly computed solution shown in Table 2, the DM wondered what should be changed to improve the income. R-XIMO provided the following suggestion: *Try improving the Income and impairing the Stored* CO_2. But the DM did not wish to impair the stored carbon dioxide anymore. Instead, he wanted to improve the stored carbon dioxide next, which was set as the target in R-XIMO. The provided suggestion by

[3] https://github.com/gialmisi/shap-experiments/blob/b446c696ae74c51fa9e1ee4a10aada98b1bc7f81/notebooks/CaseStudySolutionProcess1.ipynb

[4] https://github.com/gialmisi/shap-experiments/blob/b446c696ae74c51fa9e1ee4a10aada98b1bc7f81/notebooks/CaseStudySolutionProcess2.ipynb

Table 3 The solutions and reference points of the second solution process. See the caption of Table 2 for additional details

	Current solution			Reference point		
Iteration	Income	Stored CO_2	CSHI	Income	Stored CO_2	CSHI
1	–	–	–	3.500	7.850	3.000
2	3.720	7.983	3.057	3.500	7.750	3.100
3	3.579	7.810	3.136	–	–	–

R-XIMO was: *Try improving the Stored CO_2 and impairing the Income*. The DM thought that the suggestion was what he expected and proceeded as suggested. The reference point given by the DM is shown in Table 2.

Iteration 4. The solution in Table 2 was shown to the DM. After seeing the solution, the DM thought it was a "good and reasonable solution", was happy with it and stopped the solution process.

4.2.4 Second solution process

Iteration 1. After the DM completed the first solution process, he wished to solve the problem once more from a more ecological point of view. Thus, he preferred high values for the stored carbon dioxide and CSHI. The nadir and ideal points were naturally the same as earlier. He provided the first reference point shown in Table 3.

Iteration 2. The computed solution in Table 3 was shown to the DM. He was quite happy with it, but wished to still improve CSHI, which was set as the target. R-XIMO provided the following suggestion: *Try improving the CSHI and impairing the Stored CO_2*. The DM did as suggested and provided the reference point shown in Table 3.

Iteration 3. The first thing the DM noticed once he saw the computed solution shown in Table 3 was that the income also improved in addition to CSHI. The DM was happy with this solution and decided to stop the solution process.

4.2.5 Questions and answers

After the two solution processes, the DM was asked a few questions regarding R-XIMO and the support it provides. Below, we present the questions and the DM's answers. The answers have been slightly paraphrased to improve comprehensibility.

How useful did you find the suggestions? "*I really liked them. I liked how easy they were to understand. The fact that something was to be improved and something was to be impaired was nice. I liked that a lot. But I did not always understand why one [of the objectives] was highlighted over another.*"

How easy were the suggestions to understand? "*Generally, quite easy. Could be still simpler.*"

Did you pay any attention to the explanations? "*No, they were too long. I did not want to read them.*" (At this point, the DM went back to the explanations to read them out of curiosity.)

Did you find the explanations and suggestions supporting during the interactive solution process? "*Yes, I think so. The suggestions sort of highlighted where I should put my attention. Normally, I would just randomly change things until I get to where I want to go. I think I got where I wanted to be with fewer iterations.*"

Did you find the suggestions too repetitive or otherwise frustrating? "*No, because I did not have to iterate very often. Normally, I would find it frustrating to go back and forth [between iterations], but this time it was not frustrating because the suggestions were highlighting where I should focus, which made finding a solution a little bit easier.*"

Would you have preferred the suggestions or explanations, or both, to be visualized? "*I do not think so. If I had provided the reference points in a visual way, then yes.*"

4.2.6 Observations

The suggestions generated by R-XIMO were well received by the DM. It was also observed that each suggestion, when followed, led to an improvement in the target objective expressed by the DM. Even though in the second iteration of the first solution process (Table 2) the DM did not improve the target component in the reference point, but instead only impaired the rival, the computed solution had a better value for the target objective when compared to the previous solution. It was also interesting to note that in the third iteration in the first solution process, the first suggestion given by R-XIMO was not preferable in the opinion of the DM, which prompted him to change his preferences regarding how he would like to improve the solution. While the suggestions were well received by the DM, the explanations were practically ignored. The main reason for this was their length according to the DM. Nevertheless, the support R-XIMO provided to the DM decreased the number of iterations needed to reach a preferred solution, according to the DM. Saving the DM's time is naturally desirable.

5 Validation and results

In this section, we discuss how we have numerically validated R-XIMO. We begin with a general description of the validation setting, assumptions made, and give an example of a possible implementation of R-XIMO in Sect. 5.1. Then, we describe the numerical validation process to study how well and how often the suggestions generated by R-XIMO lead to desirable outcomes in Sect. 5.2. After that, we discuss the results of the validations and the observations made in Sects. 5.3 and 5.4, respectively.

5.1 Setting and implementation

In the numerical validation, R-XIMO is utilized according to the following pattern:

1. An initial reference point $\bar{z}_0$ is randomly generated.
2. An initial solution $\mathbf{z}_0$ is computed by utilizing $\bar{z}_0$ and the black-box $\mathfrak{B}$ (6).
3. An objective i_{target} is selected. Details about selecting the target are given later.
4. According to Algorithm 1, objective j_{rival} is computed and an explanation provided.
5. In the next iteration, a new reference point $\bar{z}_1$ is provided, where the component i_{target} is changed by a value δ; and the component j_{rival} is impaired by the same value δ. The value δ is a constant scalar value relative to the range of the respective objective function (i.e., the difference of components of the ideal and nadir points).
6. A new solution $\mathbf{z}_1$ is then computed with $\bar{z}_1$ and $\mathfrak{B}$.

Our goal is to compare $\mathbf{z}_0$ with $\mathbf{z}_1$. The expected result is that objective i_{target} should have a better value in $\mathbf{z}_1$ when compared to $\mathbf{z}_0$ in cases where the component i_{target} is improved and the component j_{rival} is impaired in $\bar{z}_1$ relative to $\bar{z}_0$. The value δ represents the change the DM makes in the components corresponding to the target and the rival in the reference point. We have limited the value δ to affect just i_{target} and j_{rival} since R-XIMO generates suggestions only concerning these two.

In the validation, we deal with two multiobjective optimization problems, the river pollution problem [47] (river problem, also considered in Sect. 4) and the vehicle crash-worthiness design problem [52] (car problem, described in more detail in the Appendix). Both problems have all objectives to be minimized. The river problem has five objectives and two decision variables, while the car problem has three objectives and five decision variables. In both problems, variables are subject to box constraints.

We consider three black-boxes (6) defined with the scalarizing functions (3), (4), and (5). We are only interested in whether the solutions computed by the considered black-boxes can be improved by utilizing the suggestions generated by R-XIMO or not. Thus, we are not comparing the performances of the scalarizing functions. We have chosen these scalarizing functions because they can generate different solutions [11]. Therefore, using them to validate R-XIMO shows how well it works for different black-boxes.

The version of R-XIMO utilized in the validations was implemented in Python utilizing the DESDEO software framework [53] for defining and solving multiobjective optimization problems. The SHAP library [38] was used to compute SHAP values. The kernel SHAP method was selected because it can be applied to any kind of black-box models to generate SHAP values.

The source code of the R-XIMO implementation is available online on GitHub.[5] Likewise, the numerical data generated during the validation is also available online.[6]

5.2 Validation

We generated approximations of the Pareto optimal fronts for both problems considered utilizing evolutionary multiobjective optimization methods (NSGA-III [49], MOEA/D [41], and RVEA [50]). Following the discussion in Sect. 3.3, we utilized the fronts as the missing data Z_{missing} (referred to in Algorithm 1) in the kernel SHAP method to compute the SHAP values for the considered black-boxes. The ideal and nadir points were calculated for both problems based on the approximations of their Pareto optimal fronts.

When calculating the SHAP values, the missing data was also used as an approximation to the original multiobjective optimization problem. This is because kernel SHAP requires evaluating the original black-box many times over the course of computing the SHAP values. However, the solutions $\mathbf{z}_0$ and $\mathbf{z}_1$ were computed using the original (analytical) formulations of the underlying multiobjective optimization problems. This was done to get more accurate solutions. In other words, the approximation of the Pareto optimal fronts were used only when calculating SHAP values.

During the course of the validations, many experiments were conducted. One experiment consisted of running R-XIMO 200 times (a single *batch*) for each objective. This amount of runs was empirically found to give statistically enough data, while taking a

[5] https://github.com/gialmisi/shap-experiments.

[6] https://nextcloud.jyu.fi/index.php/s/2R4FBDy7m533C2E.

Table 4 The five strategies employed in the validation of R-XIMO

Strategy	Description
A	Do as suggested, improve the target and impair the rival in the reference point
B	Business-as-usual, only improve the target in the reference point
C	Improve the target and impair a random component, which is not the target or the rival, in the reference point
D	Do not improve the target and impair the rival in the reference point
E	Do not improve the target and impair a random component, which is not the rival or the target, in the reference point

moderate amount of time to compute. In each batch, one of the objectives was always set as the target. For the river problem, this meant a total of 1000 iterations, and for the car crash problem, this meant a total of 600 iterations. The initial reference point $\bar{z}_0$ was generated randomly and resided in the objective space bounded by the ideal and nadir points for each problem. Between the experiments, the problem, the value δ, and the scalarizing function, were varied. Four different δ values were considered: 5%, 10%, 15%, and 20%. These values were relative to the distance between the ideal and nadir points of the considered problem and respective component being changed. Therefore, the δ values were constant and depended only on the range of the objective being changed. We decided to choose four different δ values to test how much the amount the components in the initial reference point are changed affects the change seen in the solution $\mathbf{z}_1$ when compared to $\mathbf{z}_0$. The reason for choosing these four values for δ is based on empirical testing; we found that increasing δ to be greater than 20% of the range of the respective objective, started to yield wildly varying results. In the numerical validations conducted, we found the chosen four values to give the best insight on the effect the value of δ has on the performance of R-XIMO, at least for two problems considered.

In the validation, we considered five possible ways a reference point may be changed in respect to the target and rival. These ways are characterized by the following *strategies*: A) the target is improved and the rival is impaired in the reference point $\bar{z}_0$; B) only the target is improved; C) the target is improved while some other component than the rival or the target is impaired; D) the target is not improved and the rival is impaired; and E) the target is not improved while some other component than the rival or the target is impaired. These strategies have been listed in Table 4.

Strategy A is equivalent to following the suggestion of R-XIMO fully. Strategy B represent the naive, or business-as-usual, course of action of only improving the target. Strategies B-D represent scenarios where the suggestions of R-XIMO are followed only in part. These strategies are included to check how the suggestions provided by R-XIMO work if followed only partly, and especially to check if the provided suggestion really is the best course of action if the target is to be improved. Partly following the suggestions is also a realistic behavior that can be expected from a real DM. The last strategy, strategy E, represents the case of not following the suggestion at all. This strategy has been included purely for validation purposes. Comparing how the target objective's value varies between solutions $\mathbf{z}_0$ and $\mathbf{z}_1$ when employing different strategies, gives a good indication on the performance of R-XIMO (strategy A) when compared to alternative courses of action (strategies B-E) in respect to the target and rival. Especially, the comparison of strategy A to strategy B gives a fair idea of the added value of R-XIMO to the DM, since strategy B represents the naive course of action a DM would take without the support provided by R-XIMO.

Table 5 Numerical validation results for the river problem. All listed values are percentages

Delta	SF	Strategy	Success	Neutral	Failure	median	MAD	min	max	μ_{95}	σ_{MAD}
5	RPM	A	**81.80**	0.00	18.20	**– 3.36**	2.30	– 7.83e+01	1.71e+04	– 3.54(8.00)	3.42
5	RPM	B	74.30	0.40	25.30	– 1.16	0.58	– 4.77e+01	3.46e+01	– 1.19(4.60)	0.87
5	RPM	C	71.80	0.10	28.10	– 1.21	2.26	– 4.69e+01	8.98e+01	– 1.45(6.50)	3.34
5	RPM	D	71.80	0.30	27.90	– 1.85	1.90	– 9.17e+01	1.82e+02	– 1.52(6.60)	2.81
5	RPM	E	49.40	0.30	50.30	0.00	0.43	– 5.67e+01	1.18e+02	0.03(3.20)	0.64
10	RPM	A	**83.80**	0.10	16.10	**– 6.58**	4.16	– 5.99e+01	2.45e+01	– 5.90(10.50)	6.17
10	RPM	B	74.50	0.20	25.30	– 2.19	1.10	– 6.31e+01	2.83e+02	– 2.34(3.40)	1.63
10	RPM	C	76.00	0.10	23.90	– 2.48	4.11	– 6.94e+01	2.66e+03	– 2.48(8.00)	6.09
10	RPM	D	72.90	0.20	26.90	– 1.61	3.56	– 6.52e+01	4.64e+02	– 2.25(8.50)	5.27
10	RPM	E	52.30	0.50	47.20	– 0.00	2.26	– 5.02e+01	1.22e+02	0.07(3.20)	3.34
15	RPM	A	**85.50**	0.00	14.50	**– 9.64**	5.99	– 6.18e+01	4.11e+01	– 9.08(10.80)	8.88
15	RPM	B	77.30	0.00	22.70	– 3.23	1.42	– 8.66e+01	7.76e+03	– 3.14(3.80)	2.11
15	RPM	C	76.40	0.00	23.60	– 3.84	6.75	– 7.30e+01	2.14e+02	– 4.49(10.10)	10.01
15	RPM	D	71.50	0.20	28.30	– 0.39	3.07	– 5.22e+01	1.24e+02	– 1.80(5.10)	4.55
15	RPM	E	51.50	0.30	48.20	– 0.00	0.71	– 6.82e+01	4.03e+05	0.05(2.60)	1.06
20	RPM	A	**86.50**	0.10	13.40	**– 13.75**	8.43	– 8.41e+01	5.80e+01	– 12.74(12.40)	12.50
20	RPM	B	81.50	0.00	18.50	– 4.78	2.13	– 7.12e+01	9.11e+05	– 4.79(3.50)	3.16
20	RPM	C	75.10	0.20	24.70	– 4.79	7.96	– 1.15e+02	2.90e+02	– 4.83(9.40)	11.81
20	RPM	D	71.90	0.10	28.00	– 2.01	6.40	– 7.93e+01	9.30e+01	– 3.87(8.00)	9.49
20	RPM	E	50.60	0.30	49.10	0.00	1.89	– 7.27e+01	3.37e+02	– 0.04(2.10)	2.80
5	GUESS	A	**80.50**	0.00	19.50	**– 2.21**	1.37	– 8.75e+01	1.94e+05	– 2.37(6.10)	2.03
5	GUESS	B	73.20	0.10	26.70	– 0.90	0.57	– 4.03e+01	1.64e+02	– 0.91(3.70)	0.85
5	GUESS	C	70.00	0.00	30.00	– 0.88	1.87	– 5.45e+01	1.96e+05	– 1.03(5.00)	2.77
5	GUESS	D	71.10	0.30	28.60	– 1.15	1.23	– 6.65e+01	2.74e+04	– 1.08(6.00)	1.82
5	GUESS	E	49.70	0.10	50.20	0.00	1.26	– 5.21e+01	9.55e+03	0.21(3.50)	1.86
10	GUESS	A	**80.50**	0.00	19.50	**– 4.40**	2.53	– 7.08e+01	4.71e+04	– 4.31(5.70)	3.76

Table 5 (continued)

Delta	SF	Strategy	Success	Neutral	Failure	median	MAD	min	max	μ_{95}	σ_{MAD}
10	GUESS	B	73.90	0.10	26.00	− 1.86	0.97	− 5.62e+01	4.79e+03	− 1.83(5.00)	1.44
10	GUESS	C	68.10	0.00	31.90	− 1.56	3.51	− 6.80e+01	1.85e+09	− 1.17(3.90)	5.20
10	GUESS	D	74.80	0.10	25.10	− 2.33	2.15	− 8.82e+01	2.65e+09	− 2.22(5.60)	3.18
10	GUESS	E	50.40	0.50	49.10	0.00	1.99	− 8.91e+01	2.65e+09	0.15(3.20)	2.95
15	GUESS	A	**82.70**	0.10	17.20	**− 6.43**	3.98	− 1.54e+02	2.71e+08	− 7.35(6.10)	5.90
15	GUESS	B	76.60	0.00	23.40	− 2.72	1.36	− 4.80e+01	1.81e+07	− 2.95(4.10)	2.01
15	GUESS	C	69.50	0.10	30.40	− 2.47	5.78	− 7.46e+01	1.69e+09	− 3.42(4.90)	8.56
15	GUESS	D	73.10	0.20	26.70	− 3.64	3.71	− 1.04e+02	9.79e+03	− 3.63(6.10)	5.49
15	GUESS	E	48.00	0.60	51.40	− 0.00	4.10	− 9.44e+01	8.90e+08	− 1.19(2.50)	6.08
20	GUESS	A	**83.40**	0.00	16.60	**− 9.54**	4.78	− 9.18e+01	8.60e+09	− 10.34(5.50)	7.09
20	GUESS	B	78.60	0.40	21.00	− 3.18	1.30	− 6.46e+01	2.99e+03	− 3.06(4.20)	1.92
20	GUESS	C	70.60	0.20	29.20	− 4.53	8.02	− 1.00e+02	4.17e+05	− 5.10(3.60)	11.89
20	GUESS	D	74.20	0.30	25.50	− 5.16	4.77	− 1.06e+02	1.82e+09	− 5.07(6.40)	7.07
20	GUESS	E	51.70	0.20	48.10	− 0.10	4.84	− 8.41e+01	8.93e+08	1.20(2.80)	7.17
5	STOM	A	71.90	0.10	28.00	**− 1.81**	3.42	− 1.47e+02	2.81e+03	− 2.14(4.80)	5.08
5	STOM	B	**72.30**	0.10	27.60	− 1.13	1.54	− 1.24e+02	6.71e+03	− 1.21(2.80)	2.28
5	STOM	C	71.40	0.20	28.40	− 1.19	2.78	− 9.54e+01	4.68e+03	− 1.86(5.20)	4.13
5	STOM	D	66.30	0.20	33.50	− 0.30	1.00	− 1.95e+02	1.62e+02	− 0.33(3.10)	1.49
5	STOM	E	55.30	0.20	44.50	− 0.01	1.12	− 8.89e+01	5.58e+02	− 0.16(3.60)	1.66
10	STOM	A	**73.20**	0.00	26.80	**− 3.23**	7.49	− 3.54e+02	1.20e+03	− 5.65(5.80)	11.11
10	STOM	B	66.60	0.20	33.20	− 1.56	1.83	− 1.05e+02	2.95e+03	− 1.75(3.30)	2.71
10	STOM	C	68.00	0.00	32.00	− 2.26	5.01	− 1.79e+02	1.17e+04	− 3.22(6.00)	7.42
10	STOM	D	68.00	0.20	31.80	− 0.52	2.67	− 3.18e+02	1.81e+02	− 0.24(4.30)	3.95
10	STOM	E	55.50	0.20	44.30	− 0.01	1.56	− 7.63e+01	4.85e+02	− 0.07(3.50)	2.32
15	STOM	A	66.60	0.00	33.40	**− 3.98**	9.20	− 3.13e+02	4.04e+03	− 9.05(4.80)	13.64
15	STOM	B	63.00	0.00	37.00	− 2.31	2.82	− 1.40e+02	1.12e+03	− 3.35(3.10)	4.18

Table 5 (continued)

Delta	SF	Strategy	Success	Neutral	Failure	median	MAD	min	max	μ_{95}	σ_{MAD}
15	STOM	C	62.40	0.10	37.50	– 2.51	6.68	– 2.26e+02	5.01e+03	– 4.43(5.10)	9.90
15	STOM	D	**67.10**	0.10	32.80	– 0.63	3.84	– 3.60e+02	1.27e+03	– 1.64(4.60)	5.69
15	STOM	E	53.60	0.40	46.00	– 0.01	2.06	– 1.33e+02	7.99e+01	– 0.73(3.30)	3.05
20	STOM	A	**68.60**	0.10	31.30	**– 7.12**	12.91	– 4.22e+02	1.79e+04	– 11.33(7.40)	19.14
20	STOM	B	62.00	0.10	37.90	– 2.95	4.33	– 1.20e+02	2.11e+03	– 3.65(2.30)	6.42
20	STOM	C	61.80	0.00	38.20	– 2.83	10.52	– 2.66e+02	1.31e+03	– 4.91(5.40)	15.59
20	STOM	D	66.60	0.10	33.30	– 0.52	3.20	– 2.89e+02	1.70e+03	0.05(2.80)	4.74
20	STOM	E	53.50	0.20	46.30	– 0.01	3.39	– 1.76e+02	6.77e+02	– 0.39(3.70)	5.03

The values in bold face are the highest success rates in the column *Success* and the smallest values of the median changes of the target in the column *median* for each *Delta* for all scalarizing functions (SF) considered

Table 6 Numerical validation results for the car problem. All listed values are percentages

Delta	SF	Strategy	Success	Neutral	Failure	median	MAD	min	max	μ_{95}	σ_{MAD}
5	RPM	A	**72.00**	18.00	10.00	– 1.80	1.80	– 1.96e+01	8.33e+00	– 1.86(26.33)	2.67
5	RPM	B	69.50	21.67	8.83	– 1.50	1.23	– 1.76e+01	6.40e+00	– 1.27(28.67)	1.82
5	RPM	C	71.83	17.67	10.50	– **2.03**	1.76	– 1.92e+01	2.41e+01	– 1.97(26.00)	2.61
5	RPM	D	60.33	21.50	18.17	– 0.13	0.34	– 1.77e+01	6.24e+00	– 0.16(8.50)	0.51
5	RPM	E	56.17	22.83	21.00	– 0.02	0.16	– 1.88e+01	2.78e+01	– 0.03(8.00)	0.23
10	RPM	A	**77.67**	12.33	10.00	– **3.45**	2.53	– 2.19e+01	1.01e+01	– 3.46(26.67)	3.76
10	RPM	B	74.17	20.00	5.83	– 2.96	2.26	– 1.91e+01	5.18e– 01	– 2.43(30.00)	3.35
10	RPM	C	78.83	12.00	9.17	– **3.45**	2.70	– 2.06e+01	3.62e+01	– 3.88(24.83)	4.00
10	RPM	D	63.50	17.17	19.33	– 0.42	1.26	– 2.05e+01	3.16e+01	– 0.62(14.17)	1.87
10	RPM	E	50.50	22.00	27.50	– 0.02	0.70	– 2.06e+01	9.05e+00	– 0.10(8.83)	1.04
15	RPM	A	**87.33**	6.33	6.33	– **6.81**	3.34	– 2.45e+01	1.31e+01	– 6.36(24.33)	4.95
15	RPM	B	78.00	15.50	6.50	– 4.24	3.23	– 2.08e+01	1.27e+00	– 3.55(28.83)	4.79
15	RPM	C	82.00	9.67	8.33	– 5.59	3.22	– 2.26e+01	4.14e+01	– 5.37(24.83)	4.77
15	RPM	D	67.83	15.00	17.17	– 0.81	1.90	– 2.25e+01	3.47e+01	– 1.18(14.00)	2.82
15	RPM	E	55.67	16.17	28.17	– 0.23	0.91	– 2.24e+01	4.50e+01	– 0.36(9.83)	1.35
20	RPM	A	**85.00**	5.00	10.00	– **8.12**	4.06	– 2.44e+01	3.07e+01	– 8.45(22.83)	6.02
20	RPM	B	81.50	13.00	5.50	– 5.86	2.69	– 2.25e+01	2.13e+00	– 4.80(30.00)	3.99
20	RPM	C	83.17	9.67	7.17	– 7.61	3.58	– 2.41e+01	4.32e+01	– 6.57(27.17)	5.31
20	RPM	D	66.33	11.67	22.00	– 1.23	2.86	– 2.17e+01	4.14e+01	– 1.82(17.67)	4.25
20	RPM	E	55.83	18.67	25.50	– 0.01	0.55	– 2.17e+01	3.52e+01	– 0.10(7.33)	0.82
5	GUESS	A	63.50	21.17	15.33	– 1.23	1.31	– 1.81e+01	4.67e+01	– 1.34(15.33)	1.94
5	GUESS	B	**64.17**	24.17	11.67	– 0.87	0.87	– 1.62e+01	2.94e+01	– 0.98(16.67)	1.29
5	GUESS	C	63.00	22.83	14.17	– **1.30**	1.32	– 3.19e+01	3.86e+01	– 1.45(17.33)	1.96
5	GUESS	D	54.50	21.33	24.17	– 0.15	0.86	– 1.64e+01	4.67e+01	– 0.44(17.50)	1.28
5	GUESS	E	48.67	23.33	28.00	– 0.03	0.62	– 2.39e+01	4.21e+01	– 0.22(7.67)	0.91
10	GUESS	A	**71.50**	13.33	15.17	– **3.31**	3.31	– 2.22e+01	5.26e+01	– 3.66(22.00)	4.90

Table 6 (continued)

Delta	SF	Strategy	Success	Neutral	Failure	median	MAD	min	max	μ_{95}	σ_{MAD}
10	GUESS	B	68.17	21.33	10.50	– 2.14	2.14	– 2.24e+01	3.69e+00	– 2.28(21.33)	3.17
10	GUESS	C	64.33	18.00	17.67	– 2.26	2.34	– 2.69e+01	3.82e+01	– 2.10(14.83)	3.47
10	GUESS	D	55.00	20.33	24.67	– 0.31	2.20	– 1.98e+01	5.56e+01	– 1.03(16.83)	3.27
10	GUESS	E	49.33	21.33	29.33	– 0.05	1.10	– 2.71e+01	3.96e+01	– 0.51(10.00)	1.63
15	GUESS	A	72.17	12.00	15.83	– **6.06**	4.91	– 2.43e+01	5.26e+01	– 6.33(20.83)	7.28
15	GUESS	B	**72.50**	19.50	8.00	– 2.95	2.95	– 2.31e+01	9.16e–01	– 3.29(24.50)	4.38
15	GUESS	C	70.33	10.17	19.50	– 5.54	5.29	– 3.18e+01	3.81e+01	– 4.89(20.00)	7.84
15	GUESS	D	55.50	18.67	25.83	– 0.36	2.74	– 2.18e+01	5.76e+01	– 2.10(17.67)	4.06
15	GUESS	E	50.17	16.33	33.50	– 0.06	2.90	– 2.99e+01	5.02e+01	– 1.00(10.83)	4.30
20	GUESS	A	73.33	8.67	18.00	– **7.10**	5.30	– 2.58e+01	4.87e+01	– 7.96(19.83)	7.85
20	GUESS	B	**76.17**	17.83	6.00	– 3.59	3.54	– 2.77e+01	7.60e+00	– 3.93(22.50)	5.25
20	GUESS	C	68.17	9.67	22.17	– 6.37	6.25	– 3.35e+01	4.99e+01	– 5.79(20.50)	9.27
20	GUESS	D	57.83	13.67	28.50	– 0.72	4.16	– 2.15e+01	6.45e+01	– 2.41(18.50)	6.17
20	GUESS	E	47.67	14.67	37.67	– 0.00	5.10	– 2.96e+01	5.14e+01	– 1.42(13.17)	7.56
5	STOM	A	**77.17**	9.83	13.00	– **1.53**	1.53	– 2.40e+01	1.05e+02	– 1.91(16.33)	2.27
5	STOM	B	72.83	12.17	15.00	– 0.76	0.76	– 1.82e+01	1.48e+02	– 0.85(16.50)	1.12
5	STOM	C	71.67	9.17	19.17	– 0.99	1.27	– 2.03e+01	1.15e+02	– 1.40(17.67)	1.88
5	STOM	D	66.83	13.00	20.17	– 0.30	0.35	– 2.50e+01	7.80e+00	– 0.25(8.67)	0.51
5	STOM	E	57.83	14.50	27.67	– 0.12	0.19	– 1.64e+01	9.17e+00	– 0.15(10.00)	0.28
10	STOM	A	**77.83**	6.67	15.50	– **3.23**	3.21	– 3.18e+01	1.22e+02	– 4.02(16.83)	4.76
10	STOM	B	71.50	9.17	19.33	– 1.44	1.44	– 1.99e+01	1.21e+02	– 1.39(15.67)	2.13
10	STOM	C	71.67	6.00	22.33	– 1.79	2.34	– 2.16e+01	1.14e+02	– 2.33(14.17)	3.47
10	STOM	D	69.33	14.00	16.67	– 0.25	0.56	– 2.39e+01	4.08e+01	– 0.36(8.83)	0.82
10	STOM	E	60.50	11.00	28.50	– 0.17	0.44	– 2.07e+01	3.95e+01	– 0.31(11.17)	0.65
15	STOM	A	71.83	5.83	22.33	– **3.64**	4.56	– 3.17e+01	1.27e+02	– 4.57(15.50)	6.76

Table 6 (continued)

Delta	SF	Strategy	Success	Neutral	Failure	median	MAD	min	max	μ_{95}	σ_{MAD}
15	STOM	B	71.83	8.33	19.83	− 1.89	1.90	− 2.38e+01	1.29e+02	− 2.01(14.33)	2.82
15	STOM	C	72.50	4.00	23.50	− 2.86	3.35	− 2.22e+01	1.15e+02	− 3.97(15.33)	4.97
15	STOM	D	**74.33**	8.50	17.17	− 0.78	0.81	− 2.66e+01	2.60e+01	− 0.57(9.33)	1.21
15	STOM	E	61.67	7.83	30.50	− 0.47	0.61	− 1.96e+01	3.78e+01	− 0.44(11.50)	0.91
20	STOM	A	72.17	5.00	22.83	− **5.87**	5.60	− 3.26e+01	9.90e+01	− 7.03(16.17)	8.31
20	STOM	B	69.83	4.33	25.83	− 2.60	2.92	− 2.42e+01	1.07e+02	− 3.26(12.17)	4.33
20	STOM	C	70.33	2.00	27.67	− 3.25	4.82	− 2.31e+01	1.10e+02	− 5.33(13.67)	7.15
20	STOM	D	**73.67**	9.17	17.17	− 1.17	1.20	− 2.86e+01	1.22e+01	− 1.06(11.67)	1.78
20	STOM	E	63.50	7.17	29.33	− 0.56	0.90	− 2.18e+01	5.82e+01	− 0.60(11.33)	1.34

The values in bold face are the highest success rates in the column *Success* and the smallest values of the median changes of the target in the column *median* for each *Delta* for all scalarizing functions (SF) considered

Each experiment with its variations was repeated for each strategy (Table 4). This resulted in 60 experiments performed for each problem. In each experiment, the reference points $\bar{z}_0$ and $\bar{z}_1$, the solutions $\mathbf{z}_0$ and $\mathbf{z}_1$, the index of the rival j_{rival} and the index of the target i_{target}, and the type of the explanation and suggestion (Table 1) generated, were recorded.

5.3 Results

The main results of the numerical validation runs are shown in Tables 5 and 6. All the numerical values shown in the tables are percentages. In what follows, *change* refers to the relative change of the target objective in $\mathbf{z}_0$ when compared to $\mathbf{z}_1$. Since all the objectives in the experiments are to be minimized, a negative change means an improvement in the target and a positive change indicates an impairement of the target.

The first three columns (*Delta*, *SF*, *Strategy*) in Tables 5 and 6 show the value δ, the scalarizing function (SF) used, and the strategy employed (Table 4), respectively. For each experiment, the overall rates of success (target was improved in $\mathbf{z}_1$ compared to $\mathbf{z}_0$), neutral (target was the same in $\mathbf{z}_1$ and $\mathbf{z}_0$), and failures (target was impaired in $\mathbf{z}_1$ compared to $\mathbf{z}_0$) are recorded in the columns *Success*, *Neutral*, and *Failure*, respectively.

To indicate how much the target objective's value was changed after the reference point was modified, in each experiment the *median* of the change was computed. To show the variation in the change, the *mean absolute deviation (MAD)* was used. These values are listed in the columns *median* and *MAD*, respectively. The median was used because the target's change had some very small and large values in the experiments making the mean an inaccurate measure. These values can be seen in the tables in the columns *min* and *max*, respectively. The MAD was used instead of the standard deviation for the same reason the median was used. In other words, the median and the MAD were used because they are more resilient to outliers when compared to the mean and the standard deviation.

Utilizing the MAD and assuming the changes of the target in the experiments would follow a normal distribution, a standard deviation σ_{MAD} was computed for the changes observed in each experiment and recorded in the last column of Tables 5 and 6. Again, assuming a standard distribution, the median, and the computed standard deviation σ_{MAD} (centered on the median) were used to introduce a cut−off, where the values of change residing inside the $2\sigma_{\text{MAD}}$ confidence interval were used to compute a mean μ_{95} recorded in the penultimate column in each table. The parentheses following the values listed on this column show, in percentages, how many samples were cut−off in each experiment when calculating μ_{95}. The purposes of the last two columns are to give the reader quantities that are perhaps more familiar and easier to interpret than the median and MAD. The quantities μ_{95} and σ_{MAD} are less accurate than the median and MAD, respectively. The μ_{95} and σ_{MAD} should therefore be considered with some care.

5.4 Observations

Some observations of the results in Tables 5 and 6 are worth mentioning. The average rates for a success, neutral, and failure for each strategy across all the experiments are shown in the stacked bar graphs in Fig. 3 for the river and car problem. We can see that the average rates are very similar for strategies A, B, C and D for the river problem; and for strategies A, B and C for the car problem, while for strategy D the success rate seems a little lower, yet notably higher than strategy E. Strategy E seems to have the lowest success rate and

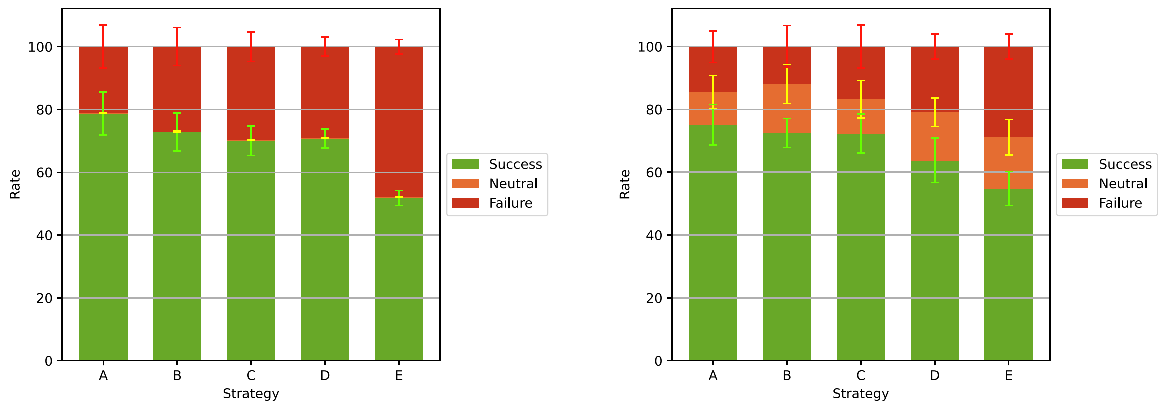


Fig. 3 Average of the success, neutral, and failure rates observed for each strategy for the river (left) and car (right) problems. The error bars show the standard error for each rate

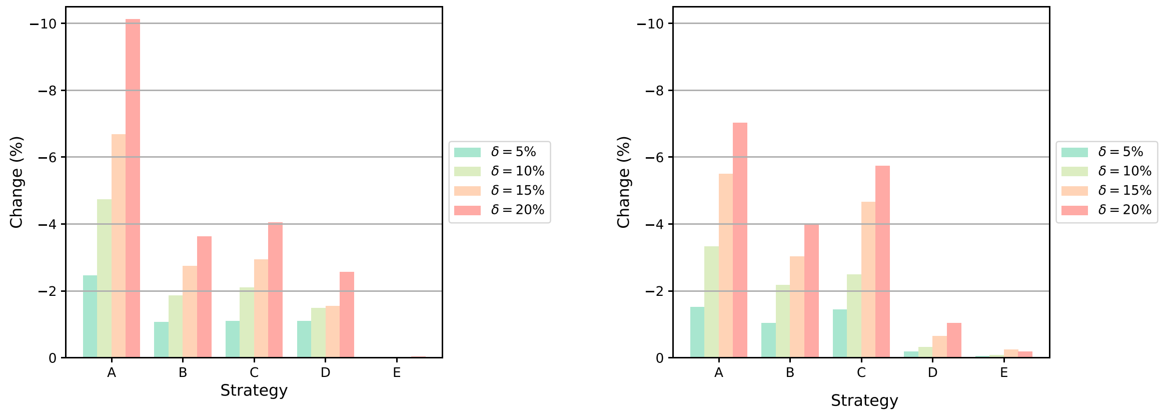


Fig. 4 The average of the median changes observed in the target for each strategy and δ value for the river (left) and car (right) problems

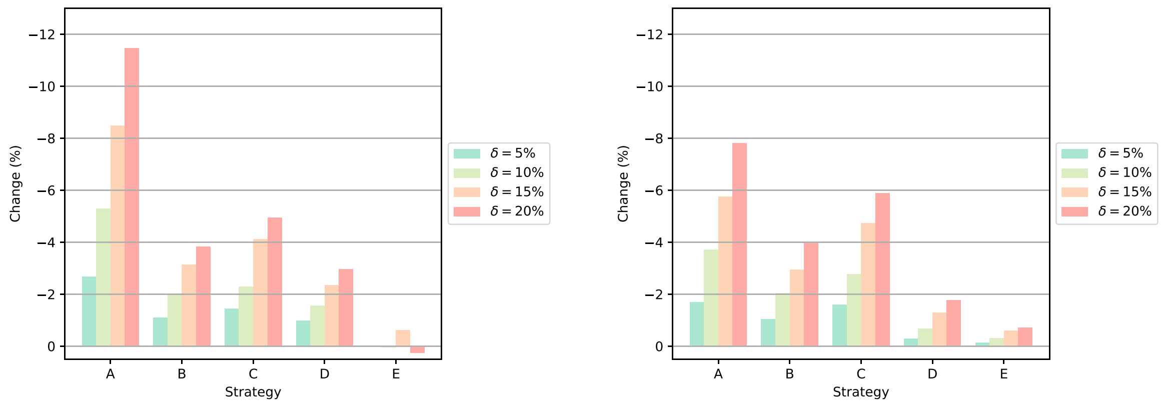


Fig. 5 The average of the μ_{95} means observed in the change of the target for each strategy and δ value for the river and car problems

highest failure rate for both problems. Looking just at the success rates, it seems that the desired result of improving the target can be achieved by just improving the target component in the reference point $\bar{z}_0$; it does not seem to matter which component is impaired, or if another component is impaired at all. If we do not improve the target and impair a component, which is not the rival (strategy E), then the success rates seem to be the worst for

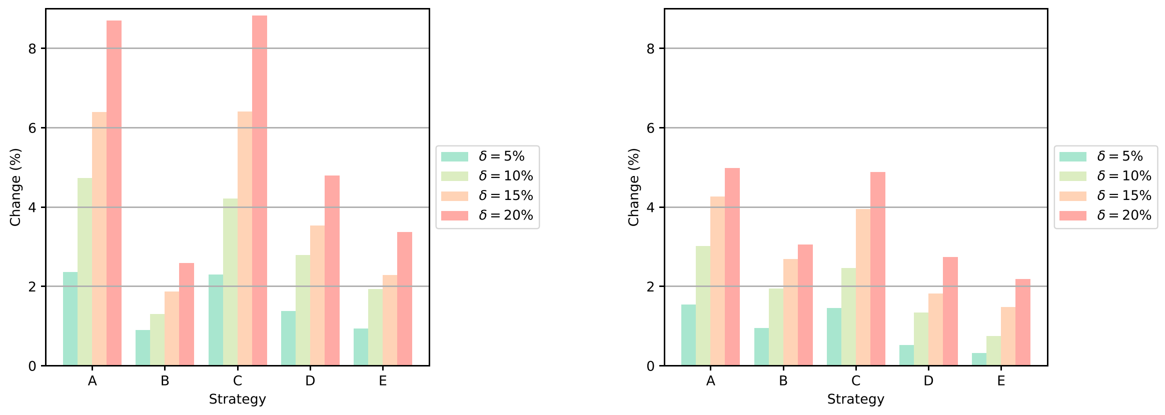


Fig. 6 The average of the median absolute deviation of changes in the target for each strategy and δ value for the river and car problems

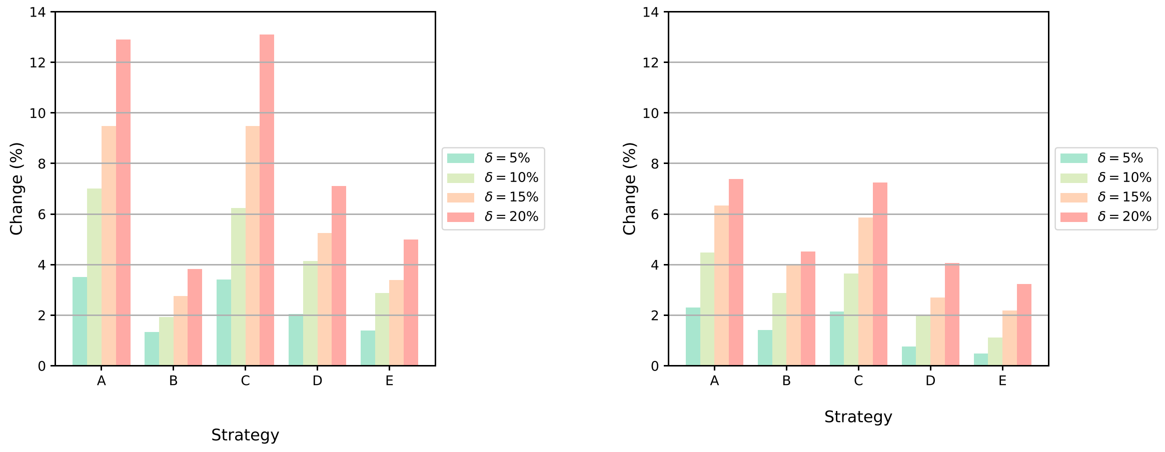


Fig. 7 The average of the σ_{MAD} deviations of changes in the target for each strategy and δ value for the river and car problems

both problems. Lastly, the rate for a neutral outcome is very low for the river problem and significantly higher for the car problem across all strategies.

The average median of the changes observed for each value δ is grouped by strategy for both problems and shown as grouped bar charts in Fig. 4. It is evident that, on average, the greatest negative changes in the target objective can be achieved by employing strategy A in both problems. The changes observed for strategy E seem to average out at zero for both problems. While strategies B, C, and D seem to yield somewhat similar results for the river problem, for the car problem, strategy D is clearly inferior to strategies B and C, while strategy C seems to be better than strategy B. Looking at Fig. 5, we can see similar results to what we see in Fig. 4—the average values of the medians are very close to the respective values of the average means. With the cut-off introduced, the mean values are close to the medians.

Looking at the average values of the MADs of the changes observed in the target shown in Fig. 6, we can observe the greatest variation for strategies A and C, for both problems. For the river problem, the variations for strategies B and E seem similar, while the variations for strategy D are a little bit higher than for B and E. For the car problem, the variations for strategies B, D, and E are more similar, with the variations in strategy B being still the highest. We can see a similar pattern for the average of the σ_{MAD} deviations shown in

Fig. 7. However, the values of the σ_{MAD} variations are noticeably larger than the medians across both problems.

In all Figs. 4, 5, 6, and 7, we can clearly see that the average changes and deviations of the changes increase systematically as the value of δ increases. The average changes seem to be best for strategy A, but also the deviations seem to be greatest for strategy A. This means that while the best average improvement of the target can be observed by employing strategy A, it can also yield very varying results. The overall worst strategy seems to be strategy E, which results, on average, in no observed change in the target. The deviations for strategy E are also small, indicating that the average changes observed in the target are not just zero-centered but also very small. It is also evident that looking just at the success rates in Fig. 3 is not enough. For instance, just looking at the success rates for strategy A would indicate that it is no different from strategy B, while the average changes clearly indicate that a better result can be achieved by employing strategy A.

In summary, the results indicate that employing strategy A, that is, improving the target and impairing the rival, as suggested by R-XIMO, has the best chance of achieving the desirable result of improving the target and in the greatest amount. The choice of the rival objective seems to also matter, otherwise strategies A and C should be similar. Moreover, it also seems that strategy D (only impairing the rival and not improving the target) can yield improvements in the target objective. Overall, the improvement of the target component in the reference point seems to be a sound course of action to be always taken when the target objective is to be improved in the solution.

6 Discussion

In this section, we discuss the validity of R-XIMO. In Sect. 6.1, we consider the results of the illustrative example given in Sect. 4.1, and in Sect. 6.2 we consider the results of the case study with a real DM conducted in Sect. 4.2. Likewise, in Sect. 6.3, we discuss the results of the numerical validations of Sect. 5. Lastly, in Sect. 6.4, we outline the overall potential of R-XIMO, its future prospects and XIMO in general.

6.1 On the example

In Sect. 4.1, the usefulness of R-XIMO was demonstrated with the river problem and an analyst as the DM. R-XIMO generated explanations and suggestions at each iteration. The most important benefit for the DM was understanding the trade-offs among the conflicting objectives with the assistance of R-XIMO. The DM was able to learn about the conflicts between the objectives and the feasibility of their preferences, thanks to the explanations and suggestions. The DM noted that the assistance increased his confidence in the final solution because he gained enough insight into the problem throughout the solution process. Moreover, this assistance made it easier for the DM to provide preference information.

When we pondered on the interactive solution process, we noticed that the DM was not aware of how strong the conflict degrees among the objectives were. This emphasizes the necessity of providing not just the trade-offs among the objectives, but also the degrees of conflict between them. Furthermore, we need to underline the importance of providing these explanations visually. We did not work on visualization perspectives of explanations because our aim was to demonstrate the benefits of explanations in interactive methods. However, this needs attention in the future.

6.2 On the case study

We utilized R-XIMO in the case study in Sect. 4.2 with a real DM. While the suggestions clearly supported the DM in the solution processes, it was also evident that the explanations were too convoluted for a real DM, which led the DM to completely ignore the explanations. Based on the DM's answers to the questions presented, the suggestions generated by R-XIMO were valuable and aided the DM in both gaining a sense of direction on what to change in the reference point to achieve a desirable result and by reducing the number of required iterations. The DM did, however, state that the suggestions could be still simpler.

It is obvious that if explanations are to be presented to the DM, further studies are needed to make this information palatable for real DMs. However, this does not mean the explanations generated by R-XIMO are completely useless since the suggestions, which were found to be very useful, are derived from the explanations. Moreover, as seen in the example given in Sect. 4.1, the explanations can be useful to an analyst. We think the most valuable lesson from the case study is the observation that future studies on how the suggestions and explanations are presented to the DM are definitely needed. We believe the right direction to pursue is the exploration of new visualizations and graphical user interfaces that better support conveying the explanations and suggestions to human DMs in a graphical format.

6.3 On the numerical validations

From the success rates shown in Fig. 3, we notice that when the target component chosen by the DM is improved or the rival computed by R-XIMO is impaired in the reference point (strategies A-D in Table 4), the target is improved most of the time (around 70–80% of the time). When neither the target is improved nor the rival is impaired (strategy E in Table 4), the success rates are clearly the worst with a failure at around 50% of the time. Therefore, improving the target in the solution requires improving it in the reference point or impairing the rival. A combination of these, improving the target and impairing the rival (strategy A in Table 4), seems to yield the best results when compared to the other strategies. Lastly, the higher rates of neutral outcomes observed for the car problem can stem from the more challenging shape of the Pareto optimal front of the problem, which may have been more challenging for the underlying optimization method used to minimize the scalarizing functions considered. Thus, there may have been local minima that multiple different reference points were mapped to. However, the results of the numerical validations do not seem to have suffered from this in any major fashion.

The above has two important implications. First, improving the target in the reference point has a clear effect on the value of the target in the solution (strategies A-D). This is expected since the black-box considered in (6) finds a solution close to the given reference point. Secondly, because the success rates for strategy E are clearly worse than for strategy D (not improving the target but impairing the rival) implies that the rival computed is indeed, on average, the best component to be impaired in the reference point. If this was not the case, then there should be no significant differences in the success rates between strategies D and E. Based on the success rates alone, we claim that the proposed method suggests a rival, which when impaired in the reference point, will yield better results for the target in the solution. This means that R-XIMO is able to capture (local) conflicts between the target and some other objective (the rival in our case).

Success rates alone do not give strong evidence that improving the target and impairing the rival (strategy A) in the reference point would be significantly better than strategies B-D, but the results show that improving the target in the reference point is always a sound action if the value of the target is to be improved. As said, this is an expected result and gives us confidence in the numerical validations.

The results for the relative improvements of the target objective's value in the solution (Figs. 4 and 5) indicate clearly that improving the target and impairing the rival in the reference point is the best strategy (strategy A). The results for strategies B, C, and D indicate that smaller improvements can be observed in the target's value in the solution when either the target is improved or the rival is impaired in the reference point. Lastly, not improving the target and not impairing the rival in the reference point leads to almost no improvement in the target's value in the solution (strategy E), and is therefore the worst course of action to be taken, which is expected.

The above observations are also confirmed when looking at Tables 5 and 6. We clearly notice from the *median* columns that the best improvements of the target's value in the solution are achieved in almost all cases across both problems when strategy A is employed. The best success rates (in the column *Success*) are also found to belong to strategy A in most cases, but not as often as the best improvement of the target.

Interestingly, strategies B (only improving the target) and C (improving the target and impairing something else than the rival) yield similar results for the river problem, while for the car problem, strategy C is somewhat better, when it comes to the relative improvement of the target objective's value in the solution (Figs. 4 and 5). One conclusion from this is that with more objectives, impairing some objective in addition to improving the target, is more important when compared to a problem with less objectives.

If we compare strategies A (improving the target and impairing the rival) and C (improving the target and impairing something else than the rival) in Figs. 4 and 5, we can see that the actual choice of the rival does also matter when the target component is improved. The results of strategy A in the river problem are clearly better than for strategy C, while for the car problem, the difference is less, but still notable. Again, this can be an indication that with less objectives, the actual choice of the rival is not as important when compared to a problem with more objectives.

Therefore, in the river problem, it is important to choose a rival, and choose it correctly for the best result, while in the car problem, the choice of the rival is not that critical and only improving the target (strategy B) can yield good results as well. This observation on the importance of choosing the rival correctly, with more objective functions present, is further confirmed by the results for strategy D (not improving the target and impairing the rival). With more objectives, in the river problem impairing only the rival will yield better results when compared to the same results for the car problem. It is also clear that impairing at least *some* component in the reference point is always better than just improving the target, which is confirmed by the improvements for strategy C (improving the target and impairing something else than the rival) being always better than for strategy B (only improving the target), for both problems. Again, this kind of behavior is expected with trade-offs existing among the objectives.

From all of this, we conclude that the rival computed by R-XIMO is, on average, the best component to be impaired in the reference point when a DM wishes to improve the target in the solution. The correct choice of the rival seems to be more important in problems with more objective functions. For the best result, the target should be improved in the reference point as well. To make more general assumptions on how important the choice of the rival is when the number of objective functions varies, further numerical studies are required.

The results for the deviations of the relative improvement of the target in the solution (Figs. 6 and 7) indicate that the best performing strategy (strategy A) is also the most volatile because of the high values of deviation when compared to the other strategies. However, for strategy A, the average absolute values of the improvement of the target in the solution (in Figs. 4 and 5) are close to the deviations. This indicates that when the target fails to improve, its new value is probably closer to its original value in the previous solution. This is assuring because it means that if the explanations provided to the DM fail helping the DM reach the desired result, the magnitude of failure is small (i.e., the target has only a slightly worse value). Interestingly, improving the target and impairing a random component in the reference point (strategy C) is as volatile as strategy A. Strategy C is also the only other strategy, apart from A, where two components are changed in the reference point. This means that changing more components in the reference point yields more varying solutions, which is an expected behavior from the black-boxes considered (6). The variations of strategy D in the river problem are clearly greater than the variations of strategies B and E, while in the car problem, the variations of strategy B are greater than strategies D and E. It seems that the rival has a greater effect of the volatility of the results depending on the number of objectives at hand. Overall, when the average improvements and the average deviations are compared to each other, there are no conflicts with the success rates (Fig. 3).

Comparing the case study to the numerical validations conducted in Sect. 5, the changes in the components of the reference point expressed by the DM varied between δ values of 2.3% and 9.0%. If the actions taken by the DM are compared to the strategies A-E, the DM pursued actions described by strategy A most of the time, with pursuing an action described by strategy D once. Based on these observations alone, any specific conclusions are hard to make. But if it is generally true that δ values around $15-20\%$ lead to greater improvements in the target objective, then some way of communicating the amount the DM should change the target and rival components in the reference point is worth exploring in the future. In conclusion, the numerical validations have shown that the suggestions generated by R-XIMO can help the DM reach the desired outcome of improving the target in the solution.

6.4 Potential of R-XIMO and future prospects

Explainability clearly provides support to the DM, as discussed in Sects. 6.1 and 6.2, and R-XIMO is able to generate sound explanations (and therefore suggestions), as discussed in Sect. 6.3. R-XIMO supports the DM in learning about the multiobjective optimization problem and helps them in formulating new preferences in reference point based interactive multiobjective optimization methods. These issues have been seldom addressed in past research, as mentioned in Sect. 1.

The importance of providing explanations to the DM in interactive multiobjective optimization methods is emphasized when they are used to make real-life decisions affecting humans. Incidentally, in the member nations of the European Union, when decisions affect humans, individuals should have a right to an explanation on why, and on which basis, such decisions have been made [54]. Therefore, R-XIMO, and the concept of XIMO in general, has important societal implications.

Making trade-offs in decision making is challenging, as discussed in [55]. R-XIMO supports the DM in this regard as well by suggesting trade-offs directly to the DM. Therefore, the cognitive load on the DM is lower in reference point based interactive multiobjective optimization methods with the support of R-XIMO. Moreover, not much research has been done on studying the local conflicts in multiobjective optimization as discussed in [56]. Because SHAP values are local, as discussed in Sect. 2.2, the interactions of the objectives represented by SHAP values reflect the local conflicts among them. Thus, R-XIMO can give insight to the DM about the trade-offs local to the current solution.

In our work, we did not utilize the SHAP values computed in R-XIMO to convey to the DM any information about the actual magnitude of the conflicts among the objectives in a multiobjective optimization problem. Moreover, no support was provided to the DM on how much the target and rival should be perturbed. We made the deliberate choice to only communicate simple and easy to understand ideas to the DM with the explanations generated by R-XIMO to not overburden the DM with additional information.

If we chose to numerically show the SHAP values to the DM, we run a risk of the DM starting to compare the SHAP values to each other, or to the objectives, instead of comparing the objectives. This would greatly reduce the actual support provided to the DM by R-XIMO. Therefore, it is important that information is communicated to the DM is an easily understood way, and in a way that minimizes the possibility for the DM to confuse different types of information. By keeping the suggestions and explanations textual, we at least have made a clear distinction between objectives (numerical information) and the suggestions and explanations (textual information).

To further minimize the additional cognitive load on the DM imposed by R-XIMO, we decided to only show the suggestions derived from the explanations (Table 1) by default, and give the DM the option to see the more detailed explanation, if they so desire. The detailed explanations were formulated in a causal tone since it has been demonstrated previously that such explanations have high cognitive value to DMs [57]. Moreover, by separately providing a suggestion to the DM, we provide them with *actionable insights* (mentioned in [57]) on how they may reach their goal of improving a specific objective. However, the SHAP values can portray valuable information to the DM and can further help them in deciding how much the components of the reference points should change. This is something R-XIMO does not provide support for. We believe this information should be communicated to the DM graphically, but to find the actual best way of communication, further studies with human participants are required. The graphical communication of explanations in multiobjective optimization is also yet to be explored.

We argue that R-XIMO could help the DM in avoiding some cognitive biases as well, such as anchoring. Indeed, as argued in [58], a common reason for the anchoring effect in interactive methods is that DMs do not have time to spend in a long interactive process. R-XIMO can speed up an interactive process by eliminating some of the time required for a DM to think about trade-offs and possibly help the DM away from a previous solution aided by the suggestions. However, based solely on our work, we cannot make any definitive claims regarding the cognitive support offered to DMs by R-XIMO.

R-XIMO can be readily scaled up to convey suggestions and explanations to the DM about the interaction between other components of the reference point and solution than the target and rival. For instance, the second order effects, i.e., the components in the reference point with the second greatest improving or impairing effect on the target in the solution. This information is already available in R-XIMO and remains only to be exploited. However, we believe that further research should be conducted on how to best convey explanations to DMs in the context of multiobjective optimization before R-XIMO is scaled up.

This is also the reason why we decided to keep the suggestions and explanations simple and focused only on the most significant effects (i.e., the target and the rival).

As mentioned in the introduction, R-XIMO can be implemented as an agent to support a DM in various ways. A generic multiagent architecture for any type of interactive methods was proposed in [16], which allows more efficient and reliable interactive solution processes through the use of specialized agents. Moreover, the architecture enables a DM to select the most suited interactive methods based on their needs in different phases during the interactive solution process. In the aforementioned architecture, a preference agent that interacts with a DM constructs their preference model by actively observing and learning the preferences. The preference agent notifies a DM whenever they provide uncertain or contradictory preferences with the help of constructed preference model. R-XIMO does not currently keep a history of the DM's preferences, but it clearly has the potential to be implemented as a preference agent as part of the said architecture and can be extended to provide not only trade-offs among objectives, but also to explain uncertainties and/or contradictions in the provided preference information. This may enable a DM to provide more accurate and reliable preference information.

7 Conclusions

We proposed the R-XIMO method to be applied with reference point based interactive multiobjective optimization methods to explain to a DM why their preferences have lead to solutions shown and how the reference point may be modified to achieve a more preferred solution. Our method can be used with any reference point based interactive method. We have incorporated multiobjective optimization with ideas from explainable artificial intelligence, and utilized SHAP values to generate the explanations. We have demonstrated the usefulness of R-XIMO in practice with an illustrative example and a case study, and we validated it numerically. We can safely say that R-XIMO has a high potential as a decision support tool clearly augmenting the insight offered by the existing reference point based interactive methods. Nevertheless, in its current state, our work should still be regarded as a proof-of-concept. Future studies are needed to explore what kind of explanations best serve the needs of DMs in finding preferred solutions in multiobjective optimization problems. In the case study, we considered the opinion of only one DM, which is clearly a limitation. To properly assess the usefulness of explanations to humans in multiobjective optimization, studies with more DMs are still required.

In the future, other tools and methods apart from SHAP values should be explored in XIMO, and implementing new interactive methods with inherent explainability should be considered as well considering also different types of preference information. Some promising methods to explore in this regard are, for instance, individual conditional expectations [59], belief rule-based systems [60] (as already preliminarily explored in [44]), and scoped rules [61], just to name a few.

By supporting the DM in providing reference points and learning about the underlying multiobjective optimization problem, we provide answers to open questions in interactive multiobjective optimization. In addition, we also advance the field of explainable interactive multiobjective optimization, or XIMO, to new horizons. Our work can be regarded as pioneering and the first of its kind for explaining scalarization-based interactive multiobjective optimization methods. Our work investigates and proposes ideas that have a strong potential to inspire a plethora of various future works exploring the concept of explainability in multiobjective optimization.

Appendix

The crash-worthiness design of vehicles problem [52] is a real-world engineering problem in which the frontal structure of vehicles is designed for crash safety. The vehicle's frontal structure absorbs the energy created by the crash, increasing passenger safety. Improving a vehicle's energy absorption capacity often increase the overall vehicle mass. On the other hand, lightweight designs are required to minimize a vehicle's mass and, as a result, its fuel consumption. Therefore, higher energy absorption and lightweight design conflict with each other, and we must find a compromise between the two to achieve a proper design.

As design variables in this problem, the thickness of five reinforced components surrounding the frontal structure affecting crash safety is chosen. The mass of a vehicle (f_1), deceleration during the full-frontal crash (f_2, which influences passenger injuries), and toe board intrusion in the offset-frontal crash (f_3, which affects the vehicle's structural integrity) are all defined as objectives to be minimized. Mathematical formulation of the multiobjective optimization problem is as follows:

$$\begin{aligned}&\text{minimize}\ \ F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))\\ &\text{subject to}\ \ 1 \leq x_j \leq 3, \quad \text{where}\, j = 1, \dots, 5\end{aligned}$$

where f_i $(i = 1, 2, 3)$ represents the relevant objectives and the decision variable x_j $(j = 1, \dots, 5)$ represents the thickness of five components of the frontal structure. Objectives have the following formulations:

$$\begin{aligned}f_1(\mathbf{x}) = {} & 1640.2823 + 2.3573285x_1 + 2.3220035x_2 + 4.5688768x_3\\ & + 7.7213633x_4 + 4.4559504x_5\\ f_2(\mathbf{x}) = {} & 6.5856 + 1.15x_1 - 1.0427x_2 + 0.9738x_3 + 0.8364x_4\\ & - 0.3695x_1x_4 + 0.0861x_1x_5 + 0.3628x_2x_4\\ & - 0.1106x_1^2 - 0.3437x_3^2 + 0.1764x_4^2\\ f_3(\mathbf{x}) = {} & -0.0551 + 0.0181x_1 + 0.1024x_2 + 0.0421x_3 - 0.0073x_1x_2\\ & + 0.024x_2x_3 - 0.0118x_2x_4 - 0.0204x_3x_4 - 0.008x_3x_5\\ & - 0.0241x_2^2 + 0.0109x_4^2\end{aligned}$$

More details about the crash-worthiness design of vehicles problem can be found in the original study [52].

Acknowledgements This work has been supported by the Academy of Finland (Grant Numbers 311877 and 322221) and the Vilho, Yrjö and Kalle Väisälä Foundation of the Finnish Academy of Science and Letters. 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. We also thank Kyle Eyvindson for acting as the decision maker in the case study conducted in Sect. 4.2.

Funding Open Access funding provided by University of Jyväskylä (JYU).

References

1. Miettinen, K. (1999). *Nonlinear multiobjective optimization*. Boston: Kluwer Academic Publishers.
2. Luque, M., Ruiz, F., & Miettinen, K. (2008). Global formulation for interactive multiobjective optimization. *OR Spectrum, 33*(1), 27–48. https://doi.org/10.1007/s00291-008-0154-3.
3. Ruiz, F., Luque, M., & Miettinen, K. (2011). Improving the computational efficiency in a global formulation (GLIDE) for interactive multiobjective optimization. *Annals of Operations Research, 197*(1), 47–70. https://doi.org/10.1007/s10479-010-0831-x.
4. Miettinen, K., Hakanen, J., & Podkopaev, D. (2016). Interactive nonlinear multiobjective optimization methods. In S. Greco, M. Ehrgott, & J. Figueira (Eds.) Multiple criteria decision analysis, 2nd edn (pp. 931–980). New York: Springer. https://doi.org/10.1007/978-1-4939-3094-4_22.
5. Miettinen, K., Ruiz, F., & Wierzbicki, A. P. (2008) Introduction to multiobjective optimization: Interactive approaches. In J. Branke, K. Deb, K. Miettinen, & R. Slowinski (Eds.) Multiobjective Optimization: Interactive and evolutionary approaches (pp. 27–57). Berlin: Springer. https://doi.org/10.1007/978-3-540-88908-3_2.
6. Afsar, B., Miettinen, K., & Ruiz, F. (2021). Assessing the performance of interactive multiobjective optimization methods. *ACM Computing Surveys, 54*(4), 85. https://doi.org/10.1145/3448301.
7. Xin, B., Chen, L., Chen, J., Ishibuchi, H., Hirota, K., & Liu, B. (2018). Interactive multiobjective optimization: A review of the state-of-the-art. *IEEE Access, 6,* 41256–41279. https://doi.org/10.1109/access.2018.2856832.
8. Belton, V., Branke, J., Eskelinen, P., Greco, S., Molina, J., Ruiz, F., & Słowiński, R. (2008). Interactive multiobjective optimization from a learning perspective. In J. Branke, K. Deb, K. Miettinen, & R. Slowinski (Eds.) Multiobjective optimization: Interactive and evolutionary approaches (pp. 405–433). Berlin: Springer. https://doi.org/10.1007/978-3-540-88908-3_15.
9. Wang, J., Liu, Y., Sun, J., Jiang, Y., & Sun, C. (2016). Diversified recommendation incorporating item content information based on MOEA/D. In *2016 49th Hawaii international conference on system sciences (HICSS)* (pp. 688–696). https://doi.org/10.1109/HICSS.2016.91. IEEE.
10. Miettinen, K., & Mäkelä, M. M. (1999). Comparative evaluation of some interactive reference point-based methods for multi-objective optimisation. *Journal of the Operational Research Society, 50*(9), 949–959. https://doi.org/10.1057/palgrave.jors.2600786.
11. Miettinen, K., & Mäkelä, M. M. (2002). On scalarizing functions in multiobjective optimization. *OR Spectrum, 24*(2), 193–213. https://doi.org/10.1007/s00291-001-0092-9.
12. Steuer, R. E. (1986). *Multiple criteria optimization: Theory, computation, and application*. New York: Wiley.
13. Lim, B. Y., Yang, Q., Abdul, A. M., & Wang, D. (2019). Why these explanations? Selecting intelligibility types for explanation goals. In *IUI Workshops'19*. https://doi.org/10.1145/1234567890.
14. Gunning, D., Stefik, M., Choi, J., Miller, T., Stumpf, S., & Yang, G.-Z. (2019). XAI-Explainable artificial intelligence. *Science Robotics*. https://doi.org/10.1126/scirobotics.aay7120.
15. Arrieta, A. B., Díaz-Rodríguez, N., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., et al. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion, 58,* 82–115. https://doi.org/10.1016/j.inffus.2019.12.012.
16. Afsar, B., Podkopaev, D., & Miettinen, K. (2020). Data-driven interactive multiobjective optimization: Challenges and a generic multi-agent architecture. *Procedia Computer Science, 176,* 281–290. https://doi.org/10.1016/j.procs.2020.08.030.
17. Branke, J., Deb, K., Miettinen, K., & Slowinski, R. (Eds.). (2008). *Multiobjective optimization: Interactive and evolutionary approaches*. Berlin: Springer.
18. Wierzbicki, A. P. (1982). A mathematical basis for satisficing decision making. *Mathematical Modelling, 3*(5), 391–405. https://doi.org/10.1016/0270-0255(82)90038-0.

19. Buchanan, J. T. (1997). A naïve approach for solving MCDM problems: the GUESS method. *Journal of the Operational Research Society, 48*(2), 202–206. https://doi.org/10.1057/palgrave.jors.2600349.
20. Nakayama, H. (1995). Aspiration level approach to interactive multi-objective programming and its applications. In P. M. Pardalos, Y. Siskos, & C. Zopounidis (Eds.) *Advances in multicriteria analysis* (pp. 147–174). Boston, MA: Springer. https://doi.org/10.1007/978-1-4757-2383-0_10.
21. Wierzbicki, A .P. (1980). The use of reference objectives in multiobjective optimization. In G. Fandel, & T. Gal (Eds.) Multiple criteria decision making, theory and applications (pp. 468–486). Berlin: Springer. https://doi.org/10.1007/978-3-642-48782-8_32.
22. Bishop, C. (2006). *Pattern recognition and machine learning*. New York: Springer.
23. Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems, 50*(3), 559–569. https://doi.org/10.1016/j.dss.2010.08.006.
24. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., et al. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis, 42,* 60–88. https://doi.org/10.1016/j.media.2017.07.005.
25. Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2018). A survey on deep learning for big data. *Information Fusion, 42,* 146–157. https://doi.org/10.1016/j.inffus.2017.10.006.
26. Liu, R., Yang, B., Zio, E., & Chen, X. (2018). Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing, 108,* 33–47. https://doi.org/10.1016/j.ymssp.2018.02.016.
27. Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence (XAI) program. *AI Magazine, 40*(2), 44–58. https://doi.org/10.1609/aimag.v40i2.2850.
28. Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue, 16*(3), 31–57. https://doi.org/10.1145/3236386.3241340.
29. Miotto, R., Wang, F., Wang, S., Jiang, X., & Dudley, J. T. (2017). Deep learning for healthcare: Review, opportunities and challenges. *Briefings in Bioinformatics, 19*(6), 1236–1246. https://doi.org/10.1093/bib/bbx044.
30. Stilgoe, J. (2017). Machine learning, social learning and the governance of self-driving cars. *Social Studies of Science, 48*(1), 25–56. https://doi.org/10.1177/0306312717741687.
31. Siegel, J., & Pappas, G. (2021). Morals, ethics, and the technology capabilities and limitations of automated and self-driving vehicles. *AI & Society*. https://doi.org/10.1007/s00146-021-01277-y.
32. Ackerman, E. (2016). People want driverless cars with utilitarian ethics, unless they're a passenger. IEEE Spectrum.
33. Molnar, C. (2019). Interpretable machine learning: A guide for making black box models explainable.
34. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016) "Why should I trust you?". In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144). New York: ACM. https://doi.org/10.1145/2939672.2939778.
35. Letham, B., Rudin, C., McCormick, T. H., & Madigan, D. (2015). Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics, 9*(3), 1350–1371. https://doi.org/10.1214/15-aoas848.
36. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K.-R. (2010). How to explain individual classification decisions. *The Journal of Machine Learning Research, 11,* 1803–1831.
37. Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S. (2020). Explainable AI: A review of machine learning interpretability methods. *Entropy, 23*(1), 18. https://doi.org/10.3390/e23010018.
38. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 4765–4774). California: Curran Associates, Inc.
39. Shapley, L. S. (2016). *17. A value for N-person games*. Princeton: Princeton University Press.
40. Morgenstern, O., & Von Neumann, J. (1953). *Theory of games and economic behavior*. Princeton: Princeton University Press.
41. Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 11*(6), 712–731. https://doi.org/10.1109/TEVC.2007.892759.
42. Sukkerd, R., Simmons, R., & Garlan, D. (2018). Toward explainable multi-objective probabilistic planning. In *2018 IEEE/ACM 4th international workshop on software engineering for smart cyber-physical systems (SEsCPS)* (pp. 19–25). https://doi.org/10.1145/3196478.3196488. IEEE.

43. Zhan, H., & Cao, Y. (2019). Relationship explainable multi-objective optimization via vector value function based reinforcement learning. arXiv preprint arXiv:1910.01919.
44. Misitano, G. (2020). Interactively learning the preferences of a decision maker in multi-objective optimization utilizing belief-rules. In *2020 IEEE symposium series on computational intelligence (SSCI)* (pp. 133–140). https://doi.org/10.1109/SSCI47803.2020.9308316. IEEE.
45. Corrente, S., Greco, S., Matarazzo, B., & Slowinski, R. (2021). Explainable interactive evolutionary multiobjective optimization. Available at SSRN 3792994.
46. Josè, B. (2009). *Decision theory and rationality*. Oxford: Oxford University Press.
47. Narula, S., & Weistroffer, H. (1989). A flexible method for nonlinear multicriteria decision-making problems. *IEEE Transactions on Systems, Man and Cybernetics, 19*(4), 883–887. https://doi.org/10.1109/21.35354.
48. Miettinen, K., & Mäkelä, M. M. (1997). Interactive method NIMBUS for nondifferentiable multiobjective optimization problems. In J. Clímaco (Ed.) Multicriteria analysis (pp. 310–319). Berlin: Springer. https://doi.org/10.1007/978-3-642-60667-0_30.
49. Deb, K., & Jain, H. (2014). 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. https://doi.org/10.1109/TEVC.2013.2281535.
50. Cheng, R., Jin, Y., Olhofer, M., & Sendhoff, B. (2016). A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation, 20*(5), 773–791. https://doi.org/10.1109/TEVC.2016.2519378.
51. Misitano, G. (2020). INFRINGER : a novel interactive multi-objective optimization method able to learn a decision maker's preferences utilizing machine learning. Master's thesis, University of Jyväskylä, Finland. http://urn.fi/URN:NBN:fi:jyu-202007065235.
52. Liao, X., Li, Q., Yang, X., Zhang, W., & Li, W. (2007). Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization, 35*(6), 561–569. https://doi.org/10.1007/s00158-007-0163-x.
53. 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. https://doi.org/10.1109/ACCESS.2021.3123825.
54. Goodman, B., & Flaxman, S. (2017). European union regulations on algorithmic decision-making and a "right to explanation''. *AI Magazine, 38*(3), 50–57. https://doi.org/10.1609/aimag.v38i3.2741.
55. Keeney, R. L. (2002). Common mistakes in making value trade-offs. *Operations Research, 50*(6), 935–945. https://doi.org/10.1287/opre.50.6.935.357.
56. Wang, H., Olhofer, M., & Jin, Y. (2017). A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. *Complex & Intelligent Systems, 3*(4), 233–245. https://doi.org/10.1007/s40747-017-0053-9.
57. Chander, A., & Srinivasan, R. (2018). Evaluating explanations by cognitive value. In *Machine learning and knowledge extraction* (pp. 314–328). Cham: Springer. https://doi.org/10.1007/978-3-319-99740-7_23.
58. Miettinen, K., Eskelinen, P., Ruiz, F., & Luque, M. (2010). NAUTILUS method: An interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research, 206*(2), 426–434. https://doi.org/10.1016/j.ejor.2010.02.041.
59. Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics, 24*(1), 44–65. https://doi.org/10.1080/10618600.2014.907095.
60. Yang, J.-B., Liu, J., Wang, J., Sii, H.-S., & Wang, H.-W. (2006). Belief rule-base inference methodology using the evidential reasoning approach-RIMER. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 36*(2), 266–285. https://doi.org/10.1109/tsmca.2005.851270.
61. Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32.

II

INTERACTIVELY LEARNING THE PREFERENCES OF A DECISION MAKER IN MULTI-OBJECTIVE OPTIMIZATION UTILIZING BELIEF-RULES

by

Misitano G. 2020

2020 IEEE Symposium Series on Computational Intelligence (SSCI), 133–140

Interactively Learning the Preferences of a Decision Maker in Multi-objective Optimization Utilizing Belief-rules

Giovanni Misitano
Faculty of Information Technology
University of Jyväskylä
P.O. Box 35, Agora, 40014, University of Jyväskylä, Finland
giovanni.a.misitano@jyu.fi

***Abstract*—Many real life problems can be modelled as multi-objective optimization problems. Such problems often consist of multiple conflicting objectives to be optimized simultaneously. Multiple optimal solutions exist to these problems, and a single solution cannot be said to be the best without preferences given by a domain expert. Preferences can be used to find satisfying solutions: optimal solutions, which best match the expert's preferences. To model the preferences of the expert, and aid him/her in finding satisfying solutions, a novel method is proposed. The method utilizes machine learning combined with belief-rule based systems to adaptively train a belief-rule based system to learn a domain expert's preferences using preference information gathered during an interactive process. Belief-rule based systems are explainable generalized expert systems, which have not been used before in the manner described in this paper to model preferences of a domain expert for a multi-objective optimization problem. In the case study conducted, the satisfying solutions found using learned preferences are concluded to be compatible with the preferences of the expert, which support the proposed method's viability as a decision making support tool.**

***Index Terms*—multiple objective optimization, belief-rule based systems, machine learning, Python, preference modelling, decision making**

I. Introduction

Many real-life problems can be modelled as multi-objective optimization problems with multiple conflicting objectives. Such problems can emerge in health care [1] and engineering [2], for example. No single solution exists to these problems and the aid of a domain expert, a decision maker, is involved. The decision maker can give preferences, which can be used to identify satisfying optimal solutions matching the given preferences. To model the preferences of a decision maker, a value function can be used [3]. The value function is not explicitly known and must be inferred from the preferences of the decision maker. If the value function is known, it would allow finding the most satisfying solutions according to the preferences of a decision maker, therefore aiding the decision maker in finding satisfying solutions to a multi-objective optimization problem.

In this paper, a novel method is proposed which utilizes machine learning and belief-rule based systems to learn the value function of a decision maker. Belief-rule based systems are generalized expert systems able to model non-linear systems. Adaptive training is utilized in the proposed method, where preferences given by a decision maker during an interactive process are utilized to adaptively train a belief-rule based system to model the value function of the decision maker. Belief-rule based systems have been chosen to be utilized in modelling the value function because they can be used to build explainable models [4]. Explainability increases the trust humans have in predictions made by computational models [5]. However, the explainability of the method is not explored in this work, and is left for future research.

Some recent works, where the value function of a decision maker is inferred using machine learning can be found in [6] and [7], for example. For more details on modelling preferences for multi-objective optimization problems, see [8].

This paper is structured in the following manner: in Section II the necessary background is given for understanding how the proposed method works, in Section III the proposed method – the INFRINGER method – is presented, in Section IV the software implementation of the INFRINGER method is briefly discussed, in Section V a case study in forest landscape planning is conducted using the INFRINGER method, and in Section VI the results of the case study are discussed. Finally, in Section VII conclusions are made regarding the feasibility of INFINGER as a method for solving multi-objective optimization problems, and how successful the method is in learning the preferences of a decision maker. This paper ends with suggestions for future research topics related to the INFRINGER method.

II. Background

A. Multi-objective optimization

A multi-objective optimization problem consists of multiple objective functions, which are to be optimized simultaneously under certain constraints. Such a problem can be defined as

$$\max_{\mathbf{x}\in X} \mathbf{f}(\mathbf{x}) = \{f_i(\mathbf{x}) \mid i \in [1, m]\}, \tag{1}$$

where $\mathbf{f}$ is a vector of m objective functions f_i to be maximized. Each f_i expects a decision variable vector $\mathbf{x}$ as its argument. The decision variable vectors $\mathbf{x}$ with n elements belong to the feasible set $X \subset \mathbb{R}^n$, which is defined by constraints imposed on $\mathbf{x}$.

The objective functions in (1) are assumed to be conflicting, which means that no solution in X is able to optimize simultaneously all objectives f_i. Instead, multiple optimal solutions $\mathbf{x}^* \in X$ exist, which form a set of optimal objective vectors $\mathbf{f}(\mathbf{x}^*) = \mathbf{z} \in Z$ for all existing optimal solutions $\mathbf{x}^*$. For two vectors $\mathbf{z}, \tilde{\mathbf{z}} \in Z$, the vector $\mathbf{z}$ dominates $\tilde{\mathbf{z}}$ if, and only if, $z_i \geq \tilde{z}_i$ is true for all $i \in [1, m]$, and $z_i > \tilde{z}_i$ is true for at least one $i \in [1, m]$, where the index i is used to denote the ith element in the vectors. The Pareto optimal set Z^{Pareto} is then defined to be the set containing all the vectors $\mathbf{z}$, which are not dominated by any other vector in Z. However, in a real world scenario, the whole extent of the Pareto optimal set for a problem is often not known. Therefore, subsets of the *true* Pareto optimal set Z^{Pareto}, known as *representations* of the Pareto optimal set, are used instead. From the definition of the Pareto optimal set, and as long as all relevant objectives are included in the definition of the multi-objective optimization problem, we can assume that the decision maker is only interested in the vectors present in the set Z^{Pareto}.

Additionally, the nadir point $\mathbf{z}^{\text{nad}}$ and the ideal point $\mathbf{z}^*$ are defined. The nadir point consists of the worst values of each objective in Z^{Pareto}, and the ideal point consists of the best values respectively. The nadir and ideal points are used to convey information on the attainable objective values of (1). It is worth noting that the ideal point is not feasible, that is, it cannot be computed from a decision vector $\mathbf{x} \in X$. On the other hand, the nadir point might or might not be feasible. In practice, the ideal point is trivial to calculate, but because the nadir point depends on the whole extent of the Pareto optimal set, it is often impossible to calculate the *true* nadir point, which is why estimates of the nadir are often used. In this paper, a payoff table was used in the case study to calculate the nadir point of the forest landscape planning problem. The payoff table alongside an alternative method for approximating the nadir point are discussed in [9].

Methods for solving multi-objective optimization problems can be divided into three categories based on how the methods incorporate a decision maker's preferences. Methods utilizing preferences before solving the multi-objective optimization problem are known as *a priori* methods. Methods that make use of preference information after the optimization process are known as *a posteriori* methods. In this paper, an *interactive* method is developed. In interactive methods, the preference information is used and updated during the optimization process. This requires for the decision maker to actively take part in the optimization process by expressing his/her preferences based for example on candidate solutions, which are computed and presented to him/her during the process. For a more detailed introduction to multi-objective optimization and multi-objective optimization methods, see [3], and for a recent review discussing interactive multi-objective optimization methods, see [10].

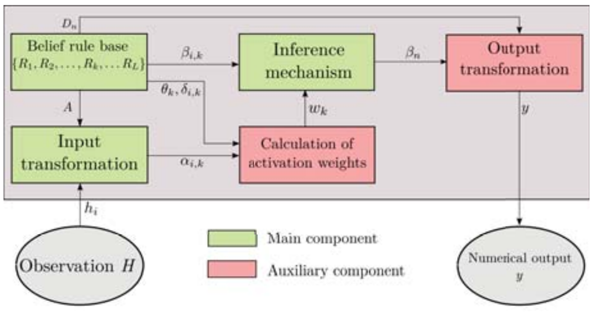


Fig. 1. The general structure of a BRB system and the relation between its components.

B. The value function

To model the preferences of a decision maker, a value function of the form

$$\mathbb{R}^m \to \mathbb{R} : f_{\text{val}}(\mathbf{z}) \tag{2}$$

is assumed to exist. The value function maps each objective vector $\mathbf{z}$ to a scalar value. The higher the value of (2) is for an objective vector $\mathbf{z}$, the more preferred $\mathbf{z}$ is according to the preferences of the decision maker. By maximizing (2), the most preferred objective vector(s) according to the decision maker's preferences can be found.

Assuming that a decision maker will always prefer more to less regarding the objectives in a problem (1), the value function (2) can be regarded to be monotonically increasing as a function of the elements z_i. This means that when one of the elements z_i is varied and the others are kept constant, the value of (2) should increase as z_i is increased.

It is unrealistic to assume a decision maker to be able to tell what his/her value function is explicitly. Therefore, the value function is otherwise inferred utilizing preference information given by the decision maker. In this paper, the value function is inferred utilizing a belief-rule based system and pair-wise comparisons of vectors in Z^{Pareto} conducted by a decision maker. Pair-wise comparisons are chosen because they are a relatively simple way for a human decision maker to convey preference information.

C. Belief-rule based systems

Belief-rule based (BRB) systems presented in this subsection are based on the RIMER methodology discussed in [11] and [12]. The BRB systems described in this paper consist of a belief-rule base, an inference mechanism, and a way to transform observed data into inputs used in the rules of the rule base. The BRB system can be trained by formulating a cost function, which is then minimized to find a set of optimal parameters for the BRB system.

The general structure of a rule base is summarized in Figure 1. Concluding this subsection, the cost function and its minimization problem are discussed The reader is advised that

$$\beta_n = \frac{\prod_{k=1}^{L}(w_k\beta_{n,k} + 1 - w_k\sum_{i=1}^{N}\beta_{i,k}) - \prod_{k=1}^{L}(1 - w_k\sum_{i=1}^{N}\beta_{i,k})}{\sum_{j=1}^{N}\prod_{k=1}^{L}(w_k\beta_{j,k} + 1 - w_k\sum_{i=1}^{N}\beta_{i,k}) - (N-1)\prod_{k=1}^{L}(1 - w_k\sum_{i=1}^{N}\beta_{i,k}) - \prod_{k=1}^{L}(1 - w_k)} \tag{5}$$

many of the symbols already used and defined in the previous subsections are reused and redefined here. This has been a deliberate choice by the author as not to inflate the number of different symbols used in this paper and to keep the notation similar to the notation found in the source literature.

The rule-base in a BRB system consists of L IF...THEN... rules R_k, where an expression follows the IF, and an action – referred to as a consequent – follows the THEN. An input to a rule is a vector $\mathbf{x}$ of attributes with T elements $x_i, i \in [1, T]$. The attribute values x_i are restricted to the values present in the antecedents of the rule base defined to be a set of T vectors $A_i, i \in [1, T]$ with each vector consisting of referential values. Then, $A_{i,j}$ denotes the jth available referential values for the ith attribute x_i, where the number of these referential values may vary from rule to rule. The vectors of referential values are aggregated in a set of vectors $A = \{A_i \mid i \in [1, T]\}$ called the packet antecedent of the rule base.

Similarly, the output of each rule is restricted to N referential values in a packet consequent $D = \{D_i \mid i \in [1, N]\}$. The element D_i indicates the ith available referential value for the consequents in the rule base.

Each rule R_k is also associated to a vector of belief-degrees $\beta_{i,k}$, where the index k indicates to which rule the degrees are associated to, and the index i indicates the belief-degree associated to the ith referential value $D_i \in D$. The value of the belief-degree indicates the likelihood for a referential value D_i being the consequent of a rule when the expression of the rule is true. Additionally, each rule R_k has an associated rule weight θ_k indicating how important the rule is in the rule base, and an associated set of attribute weights $\delta_{i,k}$, indicating how important the ith attribute is in the expression of the rule.

The ideas presented so far can be summarized in the definition of a single belief-rule with the AND-connective '$\wedge$' aggregating the expressions:

$$R_k : \begin{array}{l} \text{IF } x_1 \text{ is } A_1^k \wedge x_2 \text{ is } A_2^k \wedge \ldots \wedge x_{T_k} \text{ is } A_{T_k}^k, \\ \text{THEN } \{(D_i, \beta_{i,k}) \mid i \in [1, N]\}, \\ \text{with an associated rule weight } \theta_k, \\ \text{and attribute weights } \{\delta_{i,k} \mid i \in [1, T_k]\}, \end{array} \tag{6}$$

where A_i^k is an antecedent referential value vector, such that $A_i^k \in A_i$, and T_k is the number of attributes in the input of rule R_k. The parameters $\beta_{i,k}, \theta_k, \delta_{i,k}$, and A, can be set by an expert or learned.

Observed data H consisting of the elements $h_i, i \in [1, T]$, is transformed to a belief distribution before it can be utilized in a BRB system. Given a packet antecedent A, h_i can be transformed into the belief distribution

$$\alpha_i = \left\{ \min\left(\max\left(\frac{A_{i,j+1} - h_i}{A_{i,j+1} - A_{i,j}}, 0 \right), \max\left(\frac{h_i - A_{i,j-1}}{A_{i,j} - A_{i,j-1}}, 0 \right) \right) \middle| j \in [1, |A_i|] \right\}, \tag{7}$$

where $|A_i|$ is the number of elements in A_i, $A_{i,0} = A_{i,T}$, and $A_{i,|A_i|} = A_{i,1}$. That is to say, that the nth element in a belief distribution α_i denoted by $\alpha_{i,n}$ indicates how well an observed attribute h_i matches the referential value $A_{i,n}$ of a rule base.

Once all h_i have been transformed to a belief distribution α_i, an activation weight can be calculated for each rule as

$$w_k = \theta_k \prod_{i=1}^{T_k} \left(\alpha_{i,j}^k\right)^{\bar{\delta}_{i,k}} \Big/ \sum_{l=1}^{L} \left[\theta_l \prod_{i=1}^{T_l} \left(\alpha_{i,j}^l\right)^{\bar{\delta}_{i,l}} \right], \tag{8}$$

where $\bar{\delta}_{i,k} = (\delta_{i,k})/(\max_{i \in T_k}(\delta_{i,k}))$. The sub-indices j in the expressions $\alpha_{i,j}^{k/l}$ are determined by each belief-rule R_k so that the values in $\alpha_{i,j}^{k/l}$ convey the information on how the observed input H matches to the rule's antecedent values A^k. Then, the activation weights indicate how important each rule is given a certain input to the rule base.

Using the computed activation weights, the combined belief degrees β_n can be calculated using (5). A combined belief degree β_n conveys the likelihood for an input $\mathbf{x}$ to be mapped to the consequent referential value $D_n \in D$ in the rule base. Using a function $\mathbb{R} \to \mathbb{R} : u(D_i)$, the combined belief degrees can be transformed into a numerical output as

$$y = \sum_{i=1}^{N} u(D_i)\beta_i. \tag{9}$$

The belief-degrees $\beta_{i,k}$, the rule weights θ_k, the attribute weights $\delta_{i,k}$, and the antecedent referential values A_i, can be denoted by $P(\beta_{i,k}, \theta_k, \delta_{i,k}, A_i)$. A minimization problem is then defined

$$\min_{P \in \mathbf{P}} \xi(P), \tag{10}$$

where ξ is a cost function used to train the BRB system, and $\mathbf{P}$ denotes the set of feasible parameters P, which is defined by the following constraints:

$$\sum_{i=1}^{L} \theta_i = 1, \tag{11}$$

$$\sum_{i=1}^{L} \beta_{i,k} = 1, k \in [1, L], \tag{12}$$

$$0 \geq \delta_{i,k}, i \in [1, T] \text{ and } k \in [1, L], \tag{13}$$

$$A_{i,n} \leq A_{i,n+1}, i \in [1, T] \text{ and } n \in [1, |A_i| - 1], \tag{14}$$

where (11) and (12) are normalizations, (13) limits the attribute weights to positive values to keep the multiplicand in (8) less or equal to one, and (14) is used to keep the antecedent referential values ordered, which is assumed in (7).

By formulating the cost function ξ to reflect a difference between the numerical output (9) of the BRB system and a desired output, an optimization problem defined in (10) can be formed using the formulated cost function, and can then be solved for a set of optimal parameters $\tilde{P} \in \mathbf{P}$, which are then used to update the BRB system to better reflect a desired output effectively training the BRB system.

III. Method

A. The INFRINGER method

An interactive method for solving multi-objective optimization problems, the INFRINGER[1] method, is proposed. The method models a decision maker's preferences as a value function, which is learned utilizing a BRB system. The BRB system in the method is adaptively trained using a cost function, which is formulated based on preference information gathered during an interactive process where a decision maker is asked to conduct pair-wise comparisons of Pareto optimal objective vectors.

As its input, the INFRINGER method requires: (i) a set of objective vectors representing a Pareto optimal set Z^{Pareto} in a multi-objective optimization problem; (ii) the ideal point $\mathbf{z}^*$ and the nadir point $\mathbf{z}^{\text{nad}}$ of the set Z^{Pareto}; and (iii) a fitness threshold $\gamma_{\text{th}} \in [0,1]$ used as a termination criterion. The output of the method consists of: (i) a trained BRB system able to model the decision maker's preferences using a learned value function; and (ii) the vector in Z^{Pareto} with the highest value according to the learned value function. It is assumed that the vectors in Z^{Pareto} consists of real valued objectives scaled between $[0,1]$ using the nadir and ideal points, such that the nadir point's objectives are zero and the ideal point's objectives are one.

In the INFRINGER method, it is assumed that the given set of objective vectors representing the Pareto optimal set is sufficiently diverse and spread-out in such a way that the set can be assumed to represent the whole extent of the Pareto optimal set. This kind of input set is easily computed for a discrete data-based multi-objective optimization problem modelled after a small and limited data set. However, for non-discrete problems or problems modelled using very large amounts of data, it is not always possible to know if a representation of the Pareto optimal set is comprehensive enough to warrant the assumptions made in the INFRINGER method. Therefore, the INFRINGER method may act more as an a posteriori multi-objective optimization method instead of an interactive method, if the given representation of the Pareto optimal set is not comprehensive enough.

B. Initialization

The decision maker is shown the nadir and ideal points and is then asked whether he/she would like to give a reference point $\bar{\mathbf{z}}$, a vector consisting of desired objective values. If the decision maker does not wish to give a reference point, the reference point is set to be in the middle of the nadir and ideal points.

Next, the decision maker is shown the approximation of the Pareto optimal set alongside the reference, nadir, and ideal points. This is done to give the decision maker an idea of the available objective values.

A BRB system is then initialized with a packet antecedent A with m referential value vectors. Each A_i corresponds to the ith objective, and consists of three referential values: $A_i = \{z_i^{\text{nad}}, \bar{z}_i, z_i^*\}$, where the index i refers to the ith component in each of the vectors. The Cartesian product of the referential value vectors in A is then taken resulting in 3^m new vectors labelled as A^k, where $k \in [1, 3^m]$. These vectors are then used to construct $L = 3^m$ belief-rules (6), where the rule R_k has the antecedent referential values A^k.

The referential values for the consequent D in each rule are chosen to be $D = \{0, 0.25, 0.5, 0.75, 1\}$, because it is assumed that the numerical output of the BRB system modelling a value function is limited to the continuous range $[0,1]$. The choice of five referential values in D has been a result of trial and error – the choice has shown to yield a good balance between computational efficiency and performance of the BRB system.

To define the belief-degrees β_i for each rule R_k, an initial value function (2) must be assumed. The value function is assumed to be

$$f_{\text{p}}(\mathbf{z}) = \textstyle\sum_{i=1}^{m} z_i/m, \tag{15}$$

which is chosen, because it is simple and clearly coherent with the assumptions made in Section II-B. The initially assumed value function (15) could also be chosen differently without impairing the described method. Using the principle (15), and an identity function $u(D_i) = D_i$ in (9), a numerical output y^k can be computed for each A^k. This numerical output can then be transformed into the belief degrees of each rule R_k using (7) by substituting α_i with βi, $A_{i,*}$ with D_*, and h_i with y^k. The rest of the parameters in the initial BRB system are set to be $\theta_k = 1/L$, and $\delta_{i,k} = 1$, for all indices i, k, because no further assumptions are made regarding the modelled value function.

The initial belief-rule base has now been fully defined, and a numerical output can be computed for an objective vector $\mathbf{z}$ utilizing (7), (8), (5), and (9). This output represent a value associated to $\mathbf{z}$ according to the value function modelled by the BRB system. As a short hand notation

$$\mathfrak{B}(\mathbf{z}) = \text{"value for } \mathbf{z}\text{"} \tag{16}$$

is used to indicate the numerical output of the BRB system.

C. Pair-wise comparisons and fitness evaluation

Following the initialization of the BRB system, the decision maker is shown pairs of vectors in Z^{Pareto}. Five pairs are chosen to be shown, because it is a number of pairs low enough to be grasped by the decision maker, and high enough to still offer a decent representation of the available vectors. To choose pairs with enough variety, the standard deviation and mean are calculated from a distribution of values computed using (16) to calculate a value for each vector in Z^{Pareto}. The first three pairs are selected by finding two vectors, both being

[1]Which stands for ***I**nteractive **inf**erence **o**f prefe**r**ences us**ing** belief **r**ules*

a multiple of the standard deviation apart from each other, centered on the mean. The multiples of the standard deviation are one, two, and three, for the first three pairs selected, respectively. The fourth pair consists of the vectors with the highest and lowest values in Z^{Pareto} according to (16), and the fifth pair consists of two random vectors in Z^{Pareto}.

The decision maker is then asked for each pair if he/she prefers the first vector, the second, or if the vectors are equally preferable. According to his/her answers, it is assumed that the value function of the decision maker results in a higher value for the preferred vector, and an identical value for two equally preferred vectors. This idea is contrasted to the output of the BRB system (16), based on which a fitness γ is calculated. The maximum value of the fitness γ is 1, which indicates that the values of $\mathfrak{B}$ computed for both vectors in all the pairs agree with the preferences of the decision maker. A fitness value of 0 indicates the opposite. For example, in case of five pair-wise comparisons conducted, a fitness value of 0.6 would indicate that the BRB system compares three out of five pairs similarly to how the decision maker compared them. The preference information gathered in the pair-wise comparisons is also used in a cost function described in the next subsection.

D. Adaptive training and visualization

To adaptively train the BRB system to model the decision maker's preferences using a value function model, a cost function is defined and minimized. Three important assumptions can be made regarding the value function being modelled by the BRB system: the value of the nadir point should be worse than any other vector in Z^{Pareto}, the ideal point should correspondingly have the highest possible value, and the value function should be monotonically increasing as a function of the objective values. Additionally, a fourth assumption is also made: the modeled value function should reflect the preferences of the decision maker according to the pair-wise comparisons conducted in the manner described in Section III-C. To reflect the assumptions made, a cost function is defined for each assumption, which has a higher value the greater the disagreement between the modelled value function and the made assumption is. The cost function used in the INFRINGER method is therefore defined as the aggregation of four cost functions as

$$\xi(P \to \mathfrak{B}) = \xi_{\text{nadir}} + \xi_{\text{ideal}} + \xi_{\text{mono}} + \xi_{\text{pair}}, \qquad (17)$$

where the notation $P \to \mathfrak{B}$ indicates that the underlying BRB system is updated with the parameters P before the cost function is evaluated, ξ_{nadir} has a higher value the greater the absolute difference between $\mathfrak{B}(\mathbf{z}^{\text{nad}})$ and 0 is, ξ_{ideal} has a higher value the greater the absolute difference between $\mathfrak{B}(\mathbf{z}^{\text{ideal}})$ and 1 is, ξ_{mono} has a greater value the less monotonically increasing $\mathfrak{B}$ is, and ξ_{pair} is the reciprocal of the fitness γ discussed in Section III-C.

After a cost function (17) is defined, a minimization problem (10) is solved to find the optimal parameters $\tilde{P}$. Using the optimal parameters $\tilde{P}$ in the BRB system will result in a modelled value function $\mathfrak{B}$ congruent with the assumptions made regarding (17) in the previous paragraph. The modelled value function should therefore reflect the preferences of the decision maker based on the pair-wise comparisons conducted, and abide the assumptions made regarding the general nature of a value function discussed in Section II-B.

To further improve the modelled value function in its ability to model the decision maker's preferences, more pair-wise comparisons can be conducted, and the cost function ξ_{pair} in (17) can be updated with the new preference information gathered from the comparisons. Previously conducted pair-wise comparisons are kept in the formulation of ξ_{pair} because it is assumed that the preferences of the decision maker remain constant.

To suggest the decision maker a best solution, and to give him/her an idea of where their region of interest in Z^{Pareto} might be according to their preferences learned by the BRB system, the decision maker is shown a plot depicting a ranking of the vectors in Z^{Pareto} according to the learned value function $\mathfrak{B}$. The vector with the highest score in Z^{Pareto} is also indicated in the plot. An example of such a plot can be seen in Figure 3, where colors have been used to indicate the value of each vector in a multi-objective optimization problem with three objectives.

E. Termination

The pair-wise comparisons can be conducted again and the BRB system can be adaptively trained multiple times. After each adaptive training, the fitness γ of the value function being modelled can be re-evaluated. If the calculated fitness is equal or greater compared to the fitness threshold γ_{th}, the INFRINGER method is terminated. Optionally, the method can be terminated if the decision maker decides he/she does not wish to continue – the decision maker gets tired, or is happy with the solution(s) found. The decision maker is then shown a desired number of the top valued vectors found in the representation of Z^{Pareto} according to $\mathfrak{B}$.

As for the learned value function, the optimal parameter $\tilde{P}$ found before termination can be saved for later use, for example if the modelled value function of the decision maker is needed later. The INFRINGER method is summarized in Algorithm 1.

IV. Software

A software framework to build and train BRB systems discussed in Section II-C has been developed in Python [13]. Based on the aforementioned framework, the INFRINGER method described in Section II-C has also been implemented. The developed framework is a first of its kind. The source code for the developed framework and method is available on GitHub. [2].

V. Case study

In the case study described in this section, the developed INFRINGER method, and its software implementation, were used for solving a multi-objective optimization problem in a

[2]https://github.com/gialmisi/desdeo-brb (October 1, 2020)

Algorithm 1 The INFRINGER method

```
INPUT: A representation of the Pareto set $Z^{\text{Pareto}}$, the ideal and nadir points $\mathbf{z}^{\text{nad}}, \mathbf{z}^*$, and a fitness threshold $\gamma_{\text{th}}$.
OUTPUT: A BRB system trained to model a decision maker's preferences using a value function, and a desired number of the highest valued vectors in $Z^{\text{Pareto}}$ according to the value function modeled by the BRB system.
Step 1: Show the decision maker $\mathbf{z}^*$ and $\mathbf{z}^{\text{nad}}$, and ask if he/she would like to supply a reference point $\bar{\mathbf{z}}$ with objective values bound by $\mathbf{z}^{\text{nad}}$ and $\mathbf{z}^*$.
Step 2: If $\bar{\mathbf{z}}$ was given in Step 1, continue to Step 3. Otherwise define $\bar{\mathbf{z}}$ to be an objective vector in the middle of $\mathbf{z}^{\text{nad}}$ and $\mathbf{z}^*$.
Step 3: Initialize a BRB system as described in Section III-B.
Step 4: Conduct pair-wise comparisons of the selected vectors in $Z^{\text{Pareto}}$ as described in Section III-C.
Step 5: Formulate the cost function and minimize it as described in Section III-D. Update the BRB system with the optimal parameters $\tilde{P}$ found.
Step 6: Visualize the ranked $Z^{\text{Pareto}}$ to the decision maker as described in Section III-D using the current value function modelled by the BRB system.
Step 7: Conduct a new pair-wise comparisons of vectors in $Z^{\text{Pareto}}$ and compute a fitness $\gamma$ value as described in Section III-C.
Step 8: If the computed $\gamma \geq \gamma_{\text{th}}$ or the decision maker wishes to stop, go to Step 11. Otherwise continue to Step 10.
Step 9: Update the cost function related to the pair-wise comparisons $\xi_{\text{pair}}$ in (17) and minimize it as described in Section III-D. Update the BRB system with the optimal parameters $\tilde{P}$ found and go to Step 7.
Step 10: Visualize the highest valued vectors in $Z^{\text{Pareto}}$ as described in Section III-E.
```

forest landscape planning problem. A domain expert acted as the decision maker in the case study.

A. The problem

The problem in the case study consisted of choosing different management strategies for different parts of a forest, which are referred to as forest stands. A strategy could have consisted of chopping down all the trees in the stand and sell the timber, for example. Three indicators were chosen to represent the consequences for the whole forest resulting from choosing different strategies for each forest stand. These indicators were: total average income, average stored carbon-dioxide, and average combined habitat suitability index. The listed indicators were chosen to be the three objectives of a problem being solved and are to be maximized.

Simulated data were available in three CSV[3]-files, with values for the three objectives for each forest stand. There were 59 available managements strategies, with data for 1475 forest stands whose development over a one hundred year time horizon was simulated for all the available strategies using SIMO [14]. In the case study, only 20 out of the 59 available strategies were considered to lower the computational time for calculating a Pareto optimal objective vector set from the data.

A comprehensive representation of the Pareto optimal set was calculated based on the available data. The nadir and ideal points of the Pareto optimal set were also computed. The original data used and the source code to compute the Pareto optimal set is available on GitHub[4].

B. Solving the problem using INFRINGER

In what follows, each time a reference is made regarding a Step followed by a number, it refers to a Step present in Algorithm 1. Analyst refers to the author of this paper who was in charge of operating the INFRINGER method since the case study was conducted over a video call.

The fitness threshold γ_{th} required as one of the inputs to the INFRINGER method was set to be 1 by the analyst. This choice was made in an attempt to avoid premature termination of the method and guarantee the BRB systems to be trained at least a couple of times. This was done to improve the accuracy of the modelled value function in modelling the preferences of the decision maker.

According to Step 1, the decision maker was shown the nadir and ideal points, and the Pareto optimal set. The decision maker then wished to give a reference point, which was used for initializing a BRB system according to Steps 2 and 3. Then, pair-wise comparisons were conducted by the decision maker according to Step 4. The interface shown to the decision maker for conducting the pair-wise comparisons can be seen in Figure 2. A spider plot was also available for each pair-wise comparison to aid the decision maker in comparing the two vectors in each pair. An example of a spider plot shown can be seen in Figure 4. Based on the pair-wise comparisons conducted, the cost function was formulated and minimized according to Step 5. The BRB system was updated with the optimal parameters $\tilde{P}$ found.

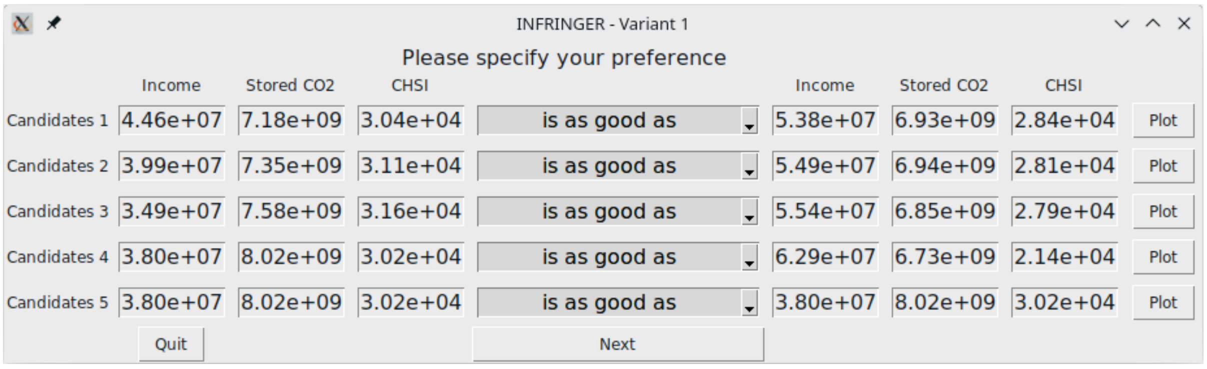
The decision maker was then shown the ranked Pareto optimal set according to Step 6. A fitness was calculated afterwards and compared to γ_{th} according to Step 8. The fitness was less than the threshold, and the method was continued according to Step 9. An additional five iterations of Steps 6 through 9 were conducted before the decision maker wished to stop the method, and the highest valued vector was shown to the decision maker as seen in Figure 3 according to Step 10.

C. General remarks

During the case study, the decision maker had made a couple of important remarks that can be used to assess the potential of the proposed INFINGER method as a tool to assist in decision making: (i) the decision maker said to be happy with

[3]Comma separated values.

[4]https://github.com/gialmisi/forest-opt (October 1, 2020)


INFRINGER - Variant 1

Please specify your preference

	Income	Stored CO2	CHSI		Income	Stored CO2	CHSI	
Candidates 1	4.46e+07	7.18e+09	3.04e+04	is as good as	5.38e+07	6.93e+09	2.84e+04	Plot
Candidates 2	3.99e+07	7.35e+09	3.11e+04	is as good as	5.49e+07	6.94e+09	2.81e+04	Plot
Candidates 3	3.49e+07	7.58e+09	3.16e+04	is as good as	5.54e+07	6.85e+09	2.79e+04	Plot
Candidates 4	3.80e+07	8.02e+09	3.02e+04	is as good as	6.29e+07	6.73e+09	2.14e+04	Plot
Candidates 5	3.80e+07	8.02e+09	3.02e+04	is as good as	3.80e+07	8.02e+09	3.02e+04	Plot

Fig. 2. The interface presented to the decision maker for pair-wise comparison of vectors (candidates) in Z^{Pareto}. For each pair shown, the decision maker may specify a candidate on the left to be either better, worse, or as good as the candidate on the right by choosing an appropriate option from the drop-down menu in the middle. Each pair can be further investigated by clicking the 'Plot' button next to each pair, which will produce a spider plot as shown in Figure 4

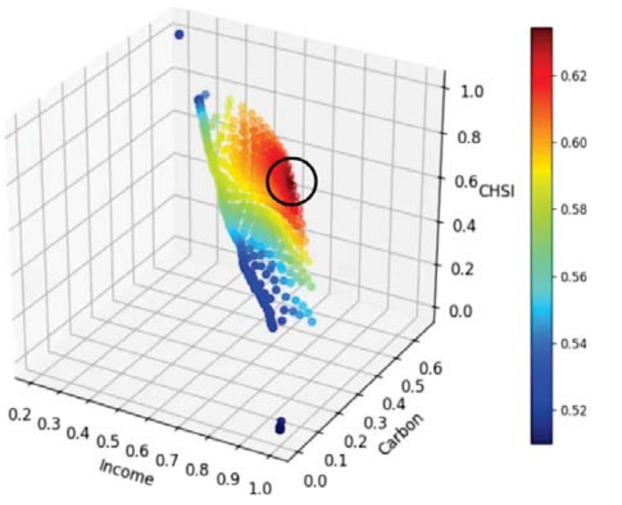


Fig. 3. The ranked Pareto optimal vectors. The highest valued vector, according to the learned preferences, is indicated by the circle. Values are normalized between 0 and 1 according to the limits imposed by the nadir and ideal points. The value of each vector is indicated by the shown color scale. Red tinted vectors have the highest values while blue tinted vectors have the lowest.

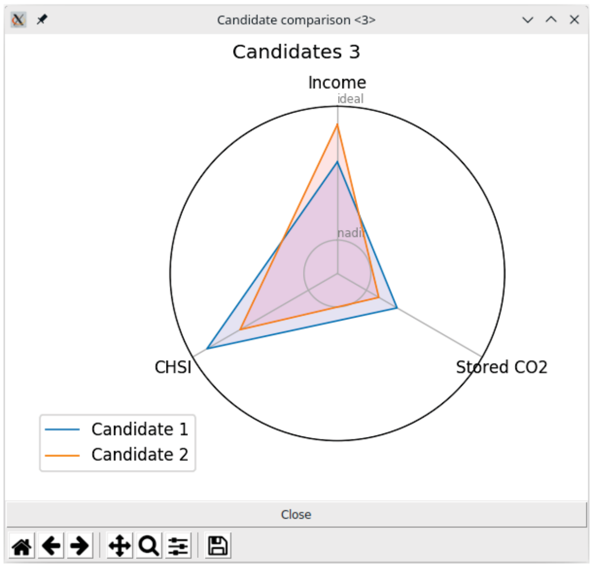


Fig. 4. A spider plot of two vectors in Z^{Pareto} (candidates) shown to the decision maker to aid him/her in comparing the two vectors. In the spider plot shown, the ideal and nadir point's values for each objective are also shown (outer radius and inner radius for the ideal and nadir values, respectively).

the highest valued vectors shown in iterations of Step 6, and he said that he felt that the objective values in the vectors shown were closing to what he was aiming at; (ii) the decision maker said that his preferences had changed during the case study; (iii) the decision maker wished for a fourth preference option to be available in the pair-wise comparisons – there should have been an option of *no preference* indicating that no preference can be given based on the two vectors shown in a pair – and he had chosen the option for equal preference to indicate absence of preference instead. However, overall the decision maker expressed to be happy in the highest valued objective vectors shown.

Additional graphical and numerical data relevant the case study is available. The material is available by request from the author.

VI. Discussion

The first remark in Section V-C, and the notion of the decision maker being overall happy with the highest valued Pareto optimal objective vectors shown, are clear indicators of the INFRINGER method being able to aid a decision maker to find satisfying solutions. The first remark indicates also that the BRB system was able to learn a value function reflecting the decision maker's preferences in the case study.

However, the decision maker had also clearly stated that his preferences had changed during the case study in the second remark – a clear contradiction to the assumption made regarding the preferences of the decision maker made at the end of Section III-D. The change in preferences can be attributed, for example to the decision maker learning about the existing trade-offs between the Pareto optimal objective vectors. This would indicate that incorporating the previously gathered preference information regarding the cost function ξ_{pair} is not correct, and it should be reconsidered whether preference information gathered in previous iterations should be used in training the BRB system in the following iterations as well.

It was also evident from the third remark that an option of no preference should be available in the pair-wise comparisons. During the case study, the option for equal preference had been used, when the decision maker wished to actually give no preference at all. This has probably lead to some error in the learned value function, and the absence of a *no preference* option should be addressed in future research related to INFRINGER.

Moreover, the comments made in Section III-A suggest that the INFRINGER method can be improved to work as an interactive method in cases where a comprehensive representation of the Pareto optimal set is not available. For example, the INFRINGER method could be enhanced by replacing the input consisting of a pre-computed set of objective vectors representing the Pareto optimal set with an evolutionary search method, which can be used to find a set of non-dominated objective vectors according to the current preference model learned in each iteration of the INFRINGER method.

Lastly, many of the choices regarding the INFINGER method have been a result of trial and error, which has been the method of choice for defining the parameters because of the novel nature of the method – there is no existing literature which would suggest choices for optimal parameters for the method proposed in this paper. Some examples of such choices are the number of vectors shown in the pair-wise comparisons, the number of antecedent and consequent referential values in the BRB system, and the choice of the initially assumed value function (15). Additional research should be conducted to find the optimal choices for these parameters.

VII. Conclusion

Based on the results of the case study discussed in Section VI, it is concluded that the proposed novel INFRINGER method is able to aid a decision maker in reaching satisfying solutions in a multi-objective optimization problem, and that the method is able to model the value function of the decision maker. However, it is also evident that the method can be improved further with additional research. Lastly, it must be noted that the BRB system used in the proposed method has the potential to offer explainable results regarding the modelled preferences of the decision maker. The explainability of the learned preferences should therefore also be considered in future works, and may aid in the choosing of the internal parameters of the BRB system.

Acknowledgments

Thanks for Jian-bo Yang, Ling Xu, and Swati Sachan, for the insightful discussion sessions over Skype.[5] Thanks for Kaisa Miettinen and Jussi Hakanen for their guidance related to my master's thesis to which this paper is based on, and for their feedback regarding this paper; and for Jose Malmberg whose code, which he developed in his master's thesis has been partly utilized to compute the Pareto set in the case study.[6] And lastly, thanks for Kyle Eyvindson for acting as the decision maker in the case study.

The research in this paper is related to the thematic research area DEMO: Decision Analytics utilizing Causal Models and Multi-objective Optimization of the University of Jyväskylä.[7] This research has been funded by the Academy of Finland: project number 322221.

References

[1] L. G. More, M. A. Brizuela, H. L. Ayala, D. P. Pinto-Roa, and J. L. V. Noguera, "Parameter tuning of CLAHE based on multi-objective optimization to achieve different contrast levels in medical images," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015.

[2] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004.

[3] K. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1998.

[4] S. Sachan, J.-B. Yang, D.-L. Xu, D. Benavides, and Y. Li, "An explainable AI decision-support-system to automate loan underwriting," *Expert Systems with Applications*, 2019.

[5] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?" in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*. ACM Press, 2016.

[6] R. Battiti and A. Passerini, "Brain–computer evolutionary multiobjective optimization: A genetic algorithm adapting to the decision maker," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 671–687, 2010.

[7] J. Branke, S. Greco, R. Slowinski, and P. Zielniewicz, "Learning value functions in interactive evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 88–102, 2015.

[8] R. L. Keeney, H. Raiffa, and D. W. Rajala, "Decisions with multiple objectives: Preferences and value trade-offs," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 7, pp. 403–403, 1979.

[9] S. Bechikh, L. Ben Said, and K. Ghedira, "Estimating nadir point in multi-objective optimization using mobile reference points," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–9.

[10] 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. 41 256–41 279, 2018.

[11] J.-B. Yang, J. Liu, J. Wang, H.-S. Sii, and H.-W. Wang, "Belief rule-base inference methodology using the evidential reasoning approach-RIMER," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 36, no. 2, pp. 266–285, 2006.

[12] Y.-W. Chen, J.-B. Yang, D.-L. Xu, Z.-J. Zhou, and D.-W. Tang, "Inference analysis and adaptive training for belief rule based systems," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12 845–12 860, 2011.

[13] P. S. Foundation. (2020) Python language reference, version 3.8. [Online]. Available: https://www.python.org

[14] J. Rasinmäki, A. Mäkinen, and J. Kalliovirta, "SIMO: An adaptable simulation framework for multiscale forest resource data," *Computers and Electronics in Agriculture*, vol. 66, no. 1, pp. 76–84, 2009.

[5]From the University of Manchester.

[6]From the University of Jyväskylä

[7]https://www.jyu.fi/demo (October 1, 2020)

III

EXPLORING THE EXPLAINABLE ASPECTS AND PERFORMANCE OF A LEARNABLE EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATION METHOD

by

Misitano G. 2023

ACM Transactions on Evolutionary Learning and Optimization, 4 (1), Article 4

Exploring the Explainable Aspects and Performance of a Learnable Evolutionary Multiobjective Optimization Method

GIOVANNI MISITANO, University of Jyväskylä, Finland

Multiobjective optimization problems have multiple conflicting objective functions to be optimized simultaneously. The solutions to these problems are known as Pareto optimal solutions, which are mathematically incomparable. Thus, a decision maker must be employed to provide preferences to find the most preferred solution. However, decision makers often lack support in providing preferences and insights in exploring the solutions available.

We explore the combination of learnable evolutionary models with interactive indicator-based evolutionary multiobjective optimization to create a learnable evolutionary multiobjective optimization method. Furthermore, we leverage interpretable machine learning to provide decision makers with potential insights about the problem being solved in the form of rule-based explanations. In fact, we show that a learnable evolutionary multiobjective optimization method can offer advantages in the search for solutions to a multiobjective optimization problem. We also provide an open source software framework for other researchers to implement and explore our ideas in their own works.

Our work is a step toward establishing a new paradigm in the field on multiobjective optimization: *explainable and learnable multiobjective optimization*. We take the first steps toward this new research direction and provide other researchers and practitioners with necessary tools and ideas to further contribute to this field.

4

CCS Concepts: • **Theory of computation** → **Bio-inspired optimization**; • **Computing methodologies** → **Rule learning**; • **Information systems** → **Decision support systems**; • **Software and its engineering** → **Open source model;**

Additional Key Words and Phrases: Multiobjective optimization, evolutionary multiobjective optimization, learnable evolutionary models, explainable artificial intelligence

ACM Reference format:
Giovanni Misitano. 2024. Exploring the Explainable Aspects and Performance of a Learnable Evolutionary Multiobjective Optimization Method. *ACM Trans. Evol. Learn.* 4, 1, Article 4 (February 2024), 39 pages.
https://doi.org/10.1145/3626104

1 INTRODUCTION

The field of **Multiobjective Optimization (MOO)** [Miettinen 1999; Sawaragi et al. 1985; Steuer 1989] specializes in solving problems with multiple conflicting objective functions with various tradeoffs. The objective functions depend on decision variables, which are often real valued. MOO problems have many mathematically incomparable optimal solutions known as **Pareto Optimal**

Author's address: G. Misitano, Faculty of Information Technology, University of Jyväskylä, P. O. Box 35 (Agora), Jyväskylä, Finland, 40014; e-mail: giovanni.a.misitano@jyu.fi.

2688-3007/2024/02-ART4
https://doi.org/10.1145/3626104

(PO) solutions. Because the solutions are incomparable without additional information, we must employ the aid of a **Decision Maker (DM)** with domain expertise to identify the most preferred solution. The DM provides preferences, which are utilized to find the best possible solution among the PO ones. MOO problems are an omnipresent kind of problem in the real world, and it is therefore important that we are not only able to understand and solve these problems, but to also provide the adequate decision support to DMs so that they may find the most preferred solution to make the best possible decisions.

The challenge of solving MOO problems is twofold. First, there is the computational and mathematical aspect of finding PO solutions. There are various approaches in this regard, such as scalarization-based (see, e.g., [Miettinen 1999]) and **Evolutionary Algorithms (EAs)** (see, e.g., [Branke et al. 2008; Coello et al. 2007]). By scalarizing a MOO problem, its objective functions are aggregated into one and the problem is reduced to a single-objective optimization problem, a scalarized problem, using some scalarization function. Then, the scalarized problem can be optimized utilizing appropriate, single-objective optimizers. The scalarization function must be selected carefully to guarantee the resulting solution to be PO [Miettinen 1999; Sawaragi et al. 1985]. In turn, EAs work by iteratively evolving a population of solution candidates, gradually improving the population. EAs are based on heuristics inspired by Darwinian evolution and cannot guarantee the Pareto optimality of the solutions, but they are able to produce multiple, near-optimal solutions, whereas by scalarizing a MOO problem one can find a single, but more accurate solution—under certain assumptions even guaranteed to be PO [Miettinen 1999; Sawaragi et al. 1985]. EAs may also be used to solve scalarized MOO problems. In addition to scalarization-based and population-based methods, other types of set approximation methods like the ones discussed in the work of Talbi et al. [2012] exist as well, but are not considered in our current work.

The second challenge of solving MOO problems lies in how to incorporate and utilize preferences provided by the DM to find the most preferred solution. Preferences can be incorporated in the solution process in three ways [Hwang and Masud 1979; Miettinen 1999]: before (*a priori*) or after (*a posteriori*) the optimization process, or interactively during the optimization process. Methods implementing the latter are known as interactive MOO methods [Miettinen 1999]. The main drawbacks of providing preferences before or after the optimization process are, respectively, the possible lack of knowledge the DM has about the characteristics of the solutions available, such as the ranges of the objective function values and the tradeoffs between them, and the (often) overwhelming amount of available solutions to choose from. Naturally, if a DM is very familiar with the MOO problem being solved, then these drawbacks are diminished. However, interactive MOO methods allow the DM to explore the problem by providing preferences iteratively and seeing what kind of solutions are available. This allows the DM to *learn* about the MOO problem [Miettinen et al. 2008]. Interactive MOO methods require little in terms of prior knowledge the DM has about the problem, and they also vary in the type of preferences they accept, how the preferences are incorporated, and what kind of information is shown to the DM [Afsar et al. 2021; Miettinen et al. 2016; Xin et al. 2018]. That being said, DMs often lack support when providing preferences in interactive MOO, which is still an open research question in the study of interactive methods [Belton et al. 2008; Wang et al. 2016].

Recently, it has been argued that interactive MOO methods can be seen as black-boxes by the DM [Misitano et al. 2022]: preferences go in the method which then generates new solutions, but how? To remedy this, *explainability* was explored as a potential solution to provide the DM with additional insights about how a scalarization-based interactive method maps preferences to generate new solutions in the work of Misitano et al. [2022]. It was shown that explainability can have the potential to support the DM in providing new preferences and reach their most preferred PO solution in fewer iterations.

In our present work, we will explore explainability in the context of EAs—specifically *indicator-based* EAs—for solving MOO problems combining them with **Learnable Evolutionary Models (LEMs)** [Michalski 2000]. LEMs are a class of optimization algorithms that combine EAs with **Machine Learning (ML)** models. The idea is to use heuristics based on Darwinian evolution in a so-called *Darwinian mode* to improve a population of solutions, then occasionally switch to a so-called *learning mode* where ML is utilized to learn a hypothesis describing what characterizes a desired solution. This hypothesis can then be used to further generate potentially desired solutions to add to the population. A LEM algorithm then switches between a Darwinian and learning mode iteratively. In our current work, we utilize an explainable and interpretable ML model in a LEM to solve MOO problems and utilize the explanations to provide new insights to the DM about the solutions computed for a MOO problem by explaining the connection between decision variables and objective function values for solutions that are congruent with preferences expressed by the DM. As a proof of concept, we develop a new interactive MOO method based on the idea of LEMs, which utilizes an indicator-based EA and an explainable ML model.

It is safe to assume for a DM to be mainly interested in the objective function values of a MOO problem [Miettinen 1999]—it is often the case that the number of objective functions in a problem is much lower than the number of decision variables, and the problem is assumed to be modeled in a manner where the objective functions represent important criteria to the DM. However, a DM can often also be interested in some of the decision variable values of the computed solutions [Hakanen et al. 2011, 2013; Kania et al. 2022]. We show that explainability can be leveraged in this aspect as well by providing DMs with additional insights about the ranges of the decision variable values in the proximity of a preferred solution. From a design perspective, where the decision variable values of a problem can also matter, insight about the variables can lead to better decision making, such as by helping DMs discover new designs and practical considerations of the solutions [Deb and Srinivasan 2006; Ray et al. 2022; Smedberg and Bandaru 2022]. Our proposed approach should be of special interest to more technical DMs, such as engineers, who are more accustomed with the implications of different decision variable values in addition to being familiar with the objective function values.

We are the first to explore the explainable aspects in solving MOO problems with LEMs. We provide simple, rule-based explanations to help a DM make a connection between the variables and objective functions in solutions found near a point of interest (from the perspective of the DM) to a MOO problem. To achieve this goal, we implement our own explainable method for solving MOO problems based on LEMs and a simple indicator-based EA. We also provide open source software for others to implement and explore their own similar methods. In our work, we explore the contribution of the ML model used to the performance of our method in finding approximate PO solutions. Based on these contributions, we pave the way toward a new subfield in MOO: **Explainable and Learnable Evolutionary Multiobjective Optimization (XLEMOO)**.

To summarize, the contributions of our present work are the following:

(1) Following the core idea of LEMs, we combine an indicator-based **Evolutionary Multiobjective Optimization (EMO)** with an interpretable ML model to produce an XLEMOO method.
(2) We explore the performance of our **Learnable Evolutionary Multiobjective Optimization (LEMOO)** method in its ability to find optimal solutions. We study this by varying parameters that are specific to the LEM part of our approach.
(3) We demonstrate and discuss the potential benefits of the explanations produced by the interpretable ML model to a DM in our LEMOO method in a showcase as a proof of concept.
(4) We provide an open source Python framework—the XLEMOO framework—to implement and explore further ideas related to XLEMOO.

The purpose of our work is to show that by combining interpretable ML models with EAs into a new method, we can potentially achieve an overall better performance of the method (compared to the EA part alone) and be able to provide valuable explanations regarding the connection between decision variables and objective functions values to a DM. To the best of our knowledge, in the context of applying LEMs to MOO, we are also the first to publish the software used and the results of the experiments conducted in a manner that allows others to reproduce our work and build on top of it promoting the openness and renewal of science.

This article is structured as follows. In Section 2, we present the necessary theory and background information, and give references to the most relevant works related to our work. In Section 3, we present the general structure of a XLEMOO method and briefly discuss the XLEMOO framework. In Section 4, we explore the impact of ML in our LEMOO method from a performance point of view by varying parameters specific to the LEM part of our method. We then move to Section 5, where we present a showcase on how the insights emerging from the explainability of our LEMOO method can help a DM in an interactive MOO context, where a DM provides preferences iteratively during the optimization process. We present the results and implications of the experiments and the showcase in Section 6, where we also discuss the further potential of explainability in the context of LEMOO. The article is then concluded with Section 7, where we draw the main conclusions of our work and propose future research directions in the field of XLEMOO.

2 BACKGROUND

In this section, we provide the background information necessary for the rest of the article. We start with Sections 2.1 and 2.2, where we briefly present the basics of (interactive) MOO and EMO focusing on indicator-based methods, respectively. In Section 2.3, we present the main ideas behind LEMs and how these have been utilized in MOO in the past. Furthermore, we briefly discuss the basics of **Explainable Artificial Intelligence (XAI)** and ML in Section 2.4. We conclude with Section 2.5, where we will review the most notable past works in which explainability has been explored in the context of MOO.

2.1 Concepts of MOO

A MOO problem can be defined as follows:

$$\begin{aligned} \text{minimize} \quad & F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})), \\ \text{s.t.} \quad & \mathbf{x} \in X \subset \mathbb{R}^n, \end{aligned} \tag{1}$$

where $f_1, f_2, \dots, f_k$ are ($k \geq 2$) objective functions, $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$ is a *decision vector* of n decision variables, and X is the set of feasible decision vectors. We assume the objective functions to be real valued—that is, $f_i : \mathbb{R}^n \rightarrow \mathbb{R}^k$. The feasible set is often defined by box-constraints on the decision variables (i.e., upper and lower bounds) or constraint functions, or both. A feasible solution to a MOO problem (1) is one that belongs to the feasible set X. When all k objective functions are evaluated at some feasible decision vector $\mathbf{x}$, the result is an *objective vector* $\mathbf{z}$, with components $z_i = f_i(\mathbf{x})$ for $i = i, \dots, k$, which forms the set of feasible objective vectors Z. We assume all objective functions to be minimized without loss in generality—a function to be maximized can be converted to a minimized function by multiplying it by -1.

While objective vectors cannot be fully compared on a mathematical basis alone, we can define the concept of *dominance* to help us order objective vectors. Suppose we have two feasible decision vectors $\mathbf{x}^1$ and $\mathbf{x}^2$ with respective objective vectors $\mathbf{z}^1$ and $\mathbf{z}^2$. Now, $\mathbf{x}^1$ is said to dominate $\mathbf{x}^2$ if, and only if, $z_i^1 \leq z_i^2$ for all $i = 1, \dots, k$ and $z_j^1 < z_j^2$ for at least *some* $j = 1, \dots, k$. We can then define the set of PO solutions as the subset of feasible solutions, which when evaluated will result in a set of mutually nondominated solutions so that no other feasible solution exists that dominates any of the solutions in the set. The set of objective vectors resulting from evaluating the set of PO solutions is the set of PO objective vectors Z^{Pareto}, which we refer to as a *Pareto set*.

To characterize the Pareto set, we define the concepts of *ideal* and *nadir* points. When minimizing each objective function, the ideal point $\mathbf{z}^*$ represent the lower bounds of the Pareto set. The ideal point can be computed by minimizing each objective function in a MOO problem (1) individually. Similarly, the nadir point $\mathbf{z}^{\text{nadir}}$ represents the upper bounds of the Pareto set. However, computing the nadir point would require knowledge on the whole and true extent of the Pareto set, which is generally unknown. Several methods exist to approximate the nadir point (e.g., [Benayoun et al. 1971; Deb and Miettinen 2009; Deb et al. 2010]).

As mentioned, a MOO problem (1) can be scalarized using a *scalarization function* $s : \mathbb{R}^k \to \mathbb{R}$ that transforms the original MOO problem into a single-objective optimization problem. The achievement scalarizing function [Wierzbicki 1980, 1982] is an example of a scalarization function. The single-objective optimization problem resulting from utilizing the achievement scalarizing function is defined as

$$\min_{\mathbf{x}\in X} s^{\text{ASF}}(F(\mathbf{x}), \bar{\mathbf{z}}, \mathbf{z}^{**}, \mathbf{z}^{\text{nadir}}) = \min_{\mathbf{x}\in X} \max_{i=1,\ldots,k} \left[\frac{f_i(\mathbf{x}) - \bar{z}_i}{z_i^{\text{nadir}} - z_i^{**}} \right] + \rho \sum_{i=1}^{k} \frac{f_i(\mathbf{x})}{z_i^{\text{nadir}} - z_i^{**}}, \tag{2}$$

where $\bar{\mathbf{z}} = (\bar{z}_1, \ldots, \bar{z}_k)$ is a *reference point*, a vector consisting of *aspiration levels,* which are provided by a DM; $\mathbf{z}^{**}$ is a *utopian* point, defined as $z_i^{**} = z_i^* - \varepsilon$, for $i = 1 \ldots k$, where ε is a small positive scalar; and ρ is also a small positive scalar value. The aspiration levels in the reference point represent objective function values a DM wishes to achieve.

The scalarization function in (2) is also able to incorporate the preferences of a DM in the optimization process in the form of aspiration levels in the reference point. As said, the aspiration levels represent objective function values the DM wishes to achieve. When solved, (2) results in a PO solution with an objective vector that is projected to the closest PO solution according to the Tchebycheff [Miettinen 1999] distance of its components to the aspiration levels in the reference point. By utilizing different reference points $\bar{\mathbf{z}}$ in (2), we can find different solutions. The summation term in (2) is known as an *augmentation term* that assures the Pareto optimality of the solution found—actually, so-called proper Pareto optimality (for details, see, e.g., [Miettinen 1999; Wierzbicki 1982]). Many scalarizing functions exist with different properties (see, e.g., [Miettinen and Mäkelä 1999, 2002]).

2.2 Evolutionary Multiobjective Optimization

EMO methods [Branke et al. 2008; Coello et al. 2007] take inspiration from Darwinian evolution to evolve a set (a *population*) of solutions (referred to as *individuals*) concurrently over multiple generations, gradually improving the overall *fitness* of the population. EMO methods can be divided into three subcategories: indicator-based [Zitzler and Künzli 2004], domination-based [Deb et al. 2002], and decomposition-based [Zhang and Li 2007] methods. Although EMO methods can generate multiple solutions at the same time, there is no guarantee for the solutions to be PO due to the heuristic nature. This is why EMO methods can be considered to generate, at best, only approximations of PO solutions and sets. Different EMO methods are more suitable for different problems and may accommodate for different preference information as well. For recent reviews on different EMO methods, see, for example, the work of Antonio and Coello [2017] and Chugh et al. [2019].

We focus our current study on an indicator-based method, which evolves a population of solutions over generations utilizing three simple evolutionary operators: crossover, mutation, and selection. The crossover, or mating, operator combines different individuals to form new ones, the mutation operator applies random mutations to the individuals, and the selection operator is used to select the individuals with a good fitness. In indicator-based methods, a quality (or performance) indicator is used in the selection step to select individuals from the population with good fitness values. The selected individuals then continue to the next generation.

By properly selecting the quality indicator, we can determine what kind of solutions the EMO algorithm will strive to find. Usually, indicators measure the quality of the whole set of found solutions by either comparing the set to another set or by producing an absolute measure of the quality of an individual set [Emmerich and Deutz 2018]. The fitness of a single solution is then determined by its contribution to the overall value of the indicator. When preference information is available from a DM, the fitness of single solutions can be determined directly by utilizing a scalarizing function, such as the one shown in (2).

When incorporating preference information in the indicator used, it is possible to focus the search for solutions in an indicator-based EMO method to a subset of the approximate PO solutions that is of special interest to a DM. In this case, there is no need to try to approximate the whole PO set. Furthermore, when the DM is allowed to change the preferences and provide them iteratively between the executions of the EMO method, we have an interactive method. From a decision-making perspective, interactive methods are important since they allow the DM to learn about the available solutions and the feasibility of their own preferences [Miettinen et al. 2008]. The details of the proposed interactive indicator-based EMO method that incorporates explainability are further fleshed out in Sections 3 and 4.

2.3 Learnable Evolutionary Models

LEMs are a special breed of EAs where the heuristic nature of EAs is augmented with the more deterministic nature of ML methods. Proposed originally in the work of Michalski [2000], LEMs are claimed to help an evolutionary process converge faster, as in fewer iterations, to an optimal solution. Leveraging the power of ML is made possible in EAs because of the large number of individuals generated during an evolutionary process. This allows an ML model to distinguish between *good* and *bad* individuals and formulate a hypothesis on how to further generate new, usually good individuals. This process can boost the overall search for optimal solutions by diversifying the population and by finding better individuals. In the original work [Michalski 2000], it was also suggested that the evolutionary part of a LEM may be replaced by ML, but we will not explore this aspect in our current work. More details on how a LEM works are given in Section 3.1.

In the past, LEMs have been applied for solving MOO problems. We next describe some of these works. Perhaps the earliest work in which LEMs have been applied for solving MOO problems was presented in by Jourdan et al. [2005], where a LEM was applied to solve a MOO problem in water system design. The authors were able to show that a LEMOO method offered notable performance advantages when compared to NSGA-II [Deb et al. 2002]. In the work of Moradi and Mirzaei [2016], a LEMOO method was merged with a knowledge base to successfully automate complementary metal-oxide semiconductor (CMOS) analog circuit design. The inclusion of a LEM was shown to considerably decrease the number of circuit evaluations. Likewise, in the work of Moradi [2018], LEMs were applied for solving a multiobjective robot path-planning problem, and it was shown that the approach led to a higher hypervolume and overall set coverage of the evolved population when compared to other similar methods. The same author further applied their approach to a vehicle routing problem with stochastic demand with similar results [Moradi 2019]. Solving MOO vehicle routing problems was also explored in the work of Niu et al. [2021] with results indicating better performance in terms of computation time and the quality of solutions when compared to the state-of-the art EMO methods.

Although the listed works have been successful in applying LEMs to solve MOO problems, none of the works provide the software for implementing and applying LEMs to solve MOO problems, nor do they provide the data of the experiments that were run. This not only raises issues with the replicability of the results but also makes it very cumbersome for other researchers and practitioners to apply LEMOO in their own works.

In our work, we provide an open source software framework for implementing and experimenting with LEMOOs as an extension of the DESDEO software framework [Misitano et al. 2021] for MOO, and we make the data and means to reproduce the experiments shown in later sections of our paper openly available. We also provide insights for researchers and practitioners alike to start building and experimenting with their own implementations of LEMOO methods.

2.4 ML and XAI

ML [Bishop 2006] is the study and application of software that can learn from data or its surroundings, or both, to make conclusions about new observations. Often, ML is used to predict new information, examples of which include the prediction of skin cancer in patients [Esteva et al. 2017] or the fault diagnosis in rotating machinery [Liu et al. 2018], for instance. However, when used in decision support, ML can be problematic because of its black-box nature. The opacity resulting from this black-box nature is not desirable if we wish to make transparent and justifiable decisions. It is an unfortunate fact that the ML model complexity correlates positively with its predictive power [Gunning and Aha 2019], but exceptions to this rule do exist [Lipton 2018]. The more complex a model is, the harder it is generally to interpret and understand.

The field of XAI [Kamath and Liu 2021; Molnar 2022] focuses on the study of the explainable aspects of **Artificial Intelligence (AI)**, including ML, and the development of XAI methods. Focusing on explainable ML, there are two main ways that explainability can be incorporated into an ML model. First, there is the *model agnostic* approach, which focuses on methods that are able to explain *any* ML model. This is achieved by just observing the input and output of an ML model and trying to build an understanding on the mapping between these two. Examples of model agnostic explainability are SHAP values [Lundberg and Lee 2017] and LIME [Ribeiro et al. 2016]. Another approach to incorporate explainability in ML is using inherently interpretable ML models. These include, but are not limited to, rule- and tree-based models. Since rules and trees are easy to interpret, there is no need to utilize external tools; instead, the rules and trees can be directly observed to build an understanding on how the ML model makes predictions. Nevertheless, sometimes even inherently interpretable ML models can become too complex to be feasibly interpreted by a human. This is the case with decision trees with a depth in the hundreds, for instance. As argued in the work of Lipton [2018], model interpretability is not a monolithic concept; it also depends on the specific model itself. For reviews on recent advancements in explainable ML and XAI methods, see, for example, the work of Arrieta et al. [2020] and Linardatos et al. [2020].

In our work, we utilize skope-rules [Goix et al. 2020] as an ML model to learn highly interpretable, and thus explainable, rule sets. Skope-rules utilize an ensemble of decision trees to learn with a high precision instances of different classes. The trees are then reduced into simple and interpretable `IF...THEN...` rules. For example, if skope-rules have been applied to learn and predict a binary classification (classes 0 or 1) on two real-valued variables a_1 and a_2 ranging from 1 to 10, then a learned rule set could look, for instance, as follows:

RULE 1: IF $a_1 < 8.2$ AND $a_2 > 1.3$ THEN PREDICT 1,
RULE 2: IF $a_1 > 5.5$ AND $a_1 < 6.3$ AND $a_2 < 1.4$ THEN PREDICT 1,
RULE 3: IF $a_2 > 1.1$ THEN PREDICT 1.

Notice that the class of 0 is assumed to hold when no rules apply to an observed sample. Each rule is also associated with a *precision*, which measures how many samples in the training dataset each rule applies to. This precision ranges from 0 to 1, where 1 means a rule applies to all samples and 0 that it applies to none. For example in the preceding first rule, when a_1 is less than or equal to 8.2, and a_2 is greater than 1.3, then the predicted class for the instance is 1. Skope-rules have been applied successfully in the literature to extract accurate rules to increase the explainability of ML applications, for example, in the work of Bologna [2021] and Narteni et al. [2021].

From a societal perspective, explainability is an important concept that must be taken into account when utilizing ML in decision support. The European Union's General Data Protection Regulation (GDPR) includes the notion for humans to have a *right to an explanation* (Recital 71[1]) in the context of algorithmic decision support, including the use of AI and ML [Goodman and Flaxman 2017].

2.5 Explainability in MOO

Since MOO methods serve primarily as decision support tools, it is no surprise that explainability has recently begun to attract attention in MOO as well. Generally, we may speak of *explainable MOO* when talking about MOO methods with incorporated explainability. In what follows, we briefly review how explainability has been applied to MOO in the literature.

In the work of Wang et al. [2016], a MOO approach was utilized to build a movie recommendation system that accounts for both accuracy of the prediction and individual diversity oriented at users. By taking into account the recommended items' content (the movie genre) when defining the lists' diversity, the recommended lists were claimed to be explainable to the user. Explainability was not, however, explored in the MOO method itself but rather in the problem formulation.

A framework to explain different policies in a multiobjective probabilistic planning setting was presented in the work of Sukkerd et al. [2018]. The explanations of the policies were meant to increase end users' confidence in the different policies. The underlying preference structure leading to each policy was also highlighted. The policy justification system was twofold: first, it described which quality attributes had been considered in each policy, and second, it could argue why a particular policy had been generated. The framework was able to argue why a certain policy was superior, or at least not worse than another one. It achieved this goal by providing the user qualitative rather than numerical information.

So-called *relationship explainable* MOO was explored in the work of Zhan and Cao [2019] through an actor-critic reinforcement learning method. The method was able to learn in a quantifiable way the inter-relationships between different objective functions by a novel concept, *marginal weights*. By being quantifiable, the authors claimed that the relationships could also be explainable. This work focused on the study of the tradeoffs between the conflicting objectives in a MOO problem.

Belief-rules were utilized in the work of Misitano [2020] to model the preferences of a DM during an interactive MOO process. The preferences were modeled as a utility function, which was learned through reinforcement learning by utilizing a belief-rule-based system as an ML model. Although the rules learned by the model could be explained, the explainable aspect of the preference model was only discussed as a potential of the proposed method.

Modeling the preferences of a DM was also explored by Corrente et al. [2021], where a dominance-based rough set approach was used to model preferences based on the results of pairwise comparisons conducted by the DM. The preferences of the DM were then highly explainable

[1]https://www.privacy-regulation.eu/en/r71.htm

because they were modeled based on human interpretable rules. In this work, an interactive EMO method was studied.

Lastly, in the work of Misitano et al. [2022], SHAP values [Lundberg and Lee 2017] were utilized to explain the relationship between preferences and computed solutions in an interactive MOO method. Based on the explanations produced, suggestions were also formulated to aid the DM in providing further preferences to help them achieve a desired goal of improving a certain objective function value in the solution. A small case study was also conducted to validate the explanations and suggestions, and it was found that the suggestions helped the DM reach his best solution in fewer iterations and with less guessing.

In our work, we provide to a DM insights about the connection between decision and objective vectors. However, we do not claim to be the first ones to explore this connection. This concept has been studied previously and has been shown to provide important information to a DM when exploring and selecting solutions in a MOO context. For example, the concept of *innovization* coined by Deb and Srinivasan [2006] leverages EMO to not only find optimal solutions but to also help unveil new design principles in engineering problems. This unveiling is based on building an understanding of the relationship between decision variables and objective functions. More recent examples of connecting the decision variables and the objective functions to support better decision making, can be found, for example, in the work of Ray et al. [2022] and Nagar et al. [2022].

Related to innovization, data-mining approaches have also been explored in the past to find design patterns and to discover knowledge in MOO problems. Although these approaches have not been explicitly referred to as *explainable*, these approaches address similar issues that explanations do. These data-mining approaches have been surveyed in the work of Bandaru et al. [2017a], and more recent advancements have been discussed in another work by Bandaru et al. [2017b]. Data mining for knowledge discovery has also been studied very recently in an interactive MOO setting as well in the work of Smedberg and Bandaru [2022].

To the best of our knowledge, our work is the first to build a connection between decision and objective vectors by utilizing explainability in particular emerging from leveraging interpretable ML in a LEM to interactively solve MOO problems. As the explanations vary depending on the ML model used, the software framework we provide makes it readily possible to explore other ML models as well leading to different explanations, which may serve different DMs better than others.

3 LEMOO AND THE XLEMOO FRAMEWORK

In this section, we present the general structure of a LEMOO method in Section 3.1. We then discuss the potential of explainability in LEMOO methods in Section 3.2. We conclude this section by discussing the proposed XLEMOO software framework to implement LEMOO methods with explainable ML models in Section 3.3. In the following sections, Sections 4 and 5, we utilize our own LEMOO method implementation built according to the framework discussed here.

3.1 LEMOO

In LEMOO, a population of solutions to a MOO problem (1) is evolved in two modes: in a *Darwinian mode* and in a *learning mode*. In both modes, the population can be iteratively evolved multiple times before switching to the other mode. The rules to determine when the mode should be switched can be defined in multiple ways. The simplest is to iterate each mode for a fixed number of times. Another option is to switch modes when a certain condition is met—for example, the fitness of the overall population has improved past a certain threshold. In our present work, for simplicity, we will switch modes after a fixed number of times. Let N^D (Darwinian) and N^L (learning) be the numbers of iterations each mode is iterated before switching to the other

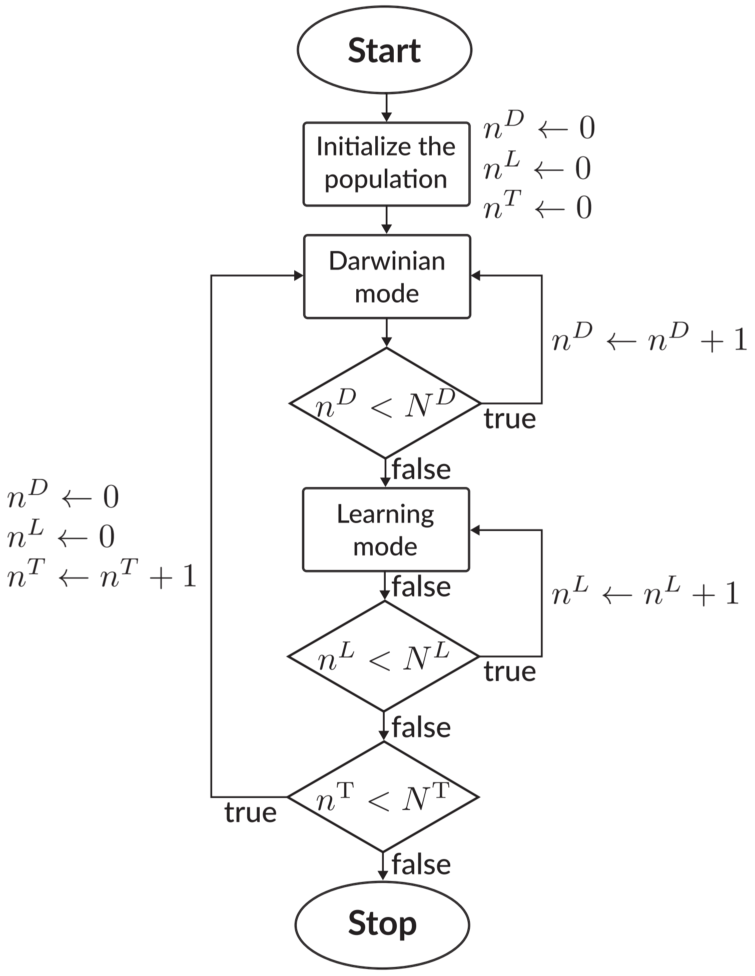


Fig. 1. The basic flowchart of a LEMOO method.

mode. Furthermore, let N^T be the total number of iterations the LEMOO method is iterated over. Iterating a LEMOO method once means evolving the population in a Darwinian mode N^D times and in a learning mode N^L times. This concept is further clarified in Figure 1. According to Figure 1, a LEMOO method starts by initializing the population of solutions. This can be done by utilizing Latin hypercube sampling [McKay et al. 2000], for instance. Whether a LEMOO method starts in a Darwinian mode or a learning mode can also be changed, but we assume that each iteration of a LEMOO method always starts in a Darwinian mode. Notice that if $N^D = 0$ or $N^L = 0$, the respective mode is skipped completely. How often a LEMOO method switches from a Darwinian mode to a learning mode is defined by the learning mode *switching frequency*. A switching frequency defines the number of iterations in a Darwinian mode to be completed before switching to a learning mode. For example, a switching frequency of 10 means that after 10 iterations in a Darwinian mode, the LEMOO method switches to a learning mode, which is equivalent to setting $N^D = 10$. Next, we further discuss the Darwinian and learning modes.

3.1.1 Darwinian Mode. In a Darwinian mode, the population is evolved utilizing an EA method. In our present work, we apply an indicator-based EMO that first performs a crossover operation on the population, then mutates it, and finally selects according to an indicator the best individuals to continue to the next generation. We utilize the scalarizing function in (2) as an indicator and utilize it directly to compute the fitness of individuals in a population. We have chosen the scalarizing function in (2) as the indicator because it incorporates preference information in the form of a reference point, which can be provided by a DM. Thus, we are able to focus the evolutionary process to find solutions that are interesting from the perspective of the DM. The scalarizing function in (2) also allows us to fully order solutions in a population based on the fitness values. Moreover, when utilizing (2) to compute the fitness, dominated solutions will always get a worse fitness when compared to the fitness of nondominated solutions. We are also able to avoid deteriorating effects by utilizing the scalarizing function in (2) as an indicator in the selection step of our indicator-based method, which effectively implements elitism. It is also important to emphasize that in our approach, we do not try to approximate the whole set of PO solutions. Instead, we focus the search on a small subset of PO solutions that are close to a reference point provided by a DM.

Other indicators may also be chosen to compute the fitness in the described approach, but we have utilized the scalarizing function in (2) throughout our current work. The EA used in Darwinian mode may not necessarily have to be an indicator-based EMO. Nonetheless, whichever EA is selected, the learning mode in LEMOO is characterized by its heuristic nature. The main flow of a Darwinian mode is depicted in Figure 2 (left).

3.1.2 Learning Mode. The idea of a learning mode is to employ an ML method to learn characteristics of good solutions. Following the original work [Michalski 2000], ML is utilized for binary classification. Individuals are classified into high-performing and low-performing groups: an *H-group* and an *L-group*, respectively (which we call an *H/L splitting*). These groups are then used as training data for the ML model. This phase is known as *hypothesis forming*. After the ML method has been trained to learn a hypothesis, it is then used to *instantiate* new individuals. The best of the new individuals are then combined with the H-group of individuals to form a new population. Individuals may also be discarded when forming a new population if the size of the population is desired to be kept constant. Depending on the ML model employed, generating new individuals can be either straightforward, such as in the case of rule-based and tree-based ML methods, or it can be more complicated, such as in the case of neural networks. Other than binary classification of the solutions is also possible. For instance, an ML model could be used to learn rules to sort solutions into the different fronts in an EMO algorithm like NSGA-II [Deb et al. 2002]. The model can then be used to generate new solutions with different ranks, possibly improving the variety of the solutions, for example.

The crucial difference on how ML is utilized in LEMOO is that ML is *not* used to predict information; instead, it is used to *describe* and generate new data. One important choice with binary classification to be made when implementing a LEMOO method is to decide how the H- and L-groups are chosen. One option, and the one utilized in our current work, is to pick a percentile of the best and worst individuals based on their fitness values and form the H- and L-groups, respectively. We have utilized the fitness values computed based on the scalarizing function in (2), which reflects the preferences of a DM. Furthermore, we assume the top and bottom percentage to be always equal and represented by an *H/L splitting ratio* or just *H/L split*. The main flow of a learning mode is depicted in Figure 2 (right).

As seen in Figure 2, both Darwinian and learning modes act on a population to improve it. In a Darwinian mode, new solutions are produced based on heuristics and the best solutions are selected based on the solutions' fitness values. However, in learning mode, a hypothesis is learned that describes good individuals and it is used to generate new, potentially good individuals. When the rule learned is interpretable by humans, such as in rule-based ML, we have also access to explanations and explainability.

3.2 The Potential for Explainability in LEMOO

When properly selected, the ML model in a LEMOO method not only gives a boost to the performance of the method but also makes the method explainable for a DM. Explainability is what can really elevate LEMOO methods as decision support tools when compared to traditional EMO

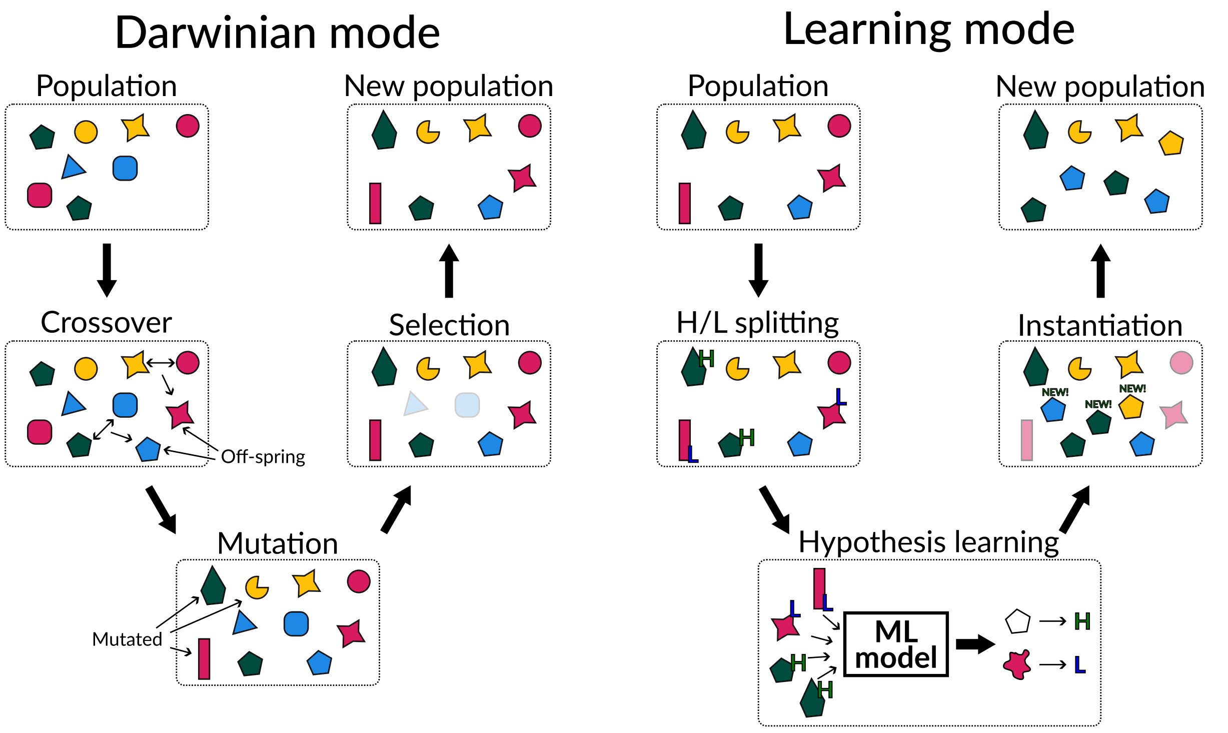


Fig. 2. A pictorial representation of how the two modes of LEM, Darwinian mode and learning mode, act on the population of solutions to create a new population.

methods. For example, by selecting an inherently interpretable ML model, such as a rule-based model, the rules generated by the model can be used to provide the DM with additional insights about the solutions in a population. In our current work, we have decided to utilize skope-rules as an interpretable form of ML because of its low computational demands and the interpretability of the generated rules. Usually, the final population has the solutions with the best fitness values, which makes the ML model trained in the last iteration of a LEMOO method of particular interest from an explainability perspective. For example, if a rule-based ML model has been trained to classify individuals in a population into H- and L-groups based on the individuals' decision vectors, the learned rules can then be shown to a DM to provide additional information about what kind of decision variable values constitute a good solution. In our present work, we have utilized the scalarizing function in (2), which incorporates a DM's preferences. This means that the information extracted from the rules will describe solutions that pertain to the DM's preferences in the decision space of the MOO problem being solved. As mentioned in Section 1, the DM may be expected to be mainly interested in the objective function values of the solutions, but from a design and engineering perspective, the decision variable values can play an important role in the final decision making. We will explore the explanations produced by our LEMOO method in an interactive setting in Section 5 to further clarify the added benefits of explainability. We will also see that the information extracted from the rules can support a DM in providing new preferences addressing an open question in the field of interactive MOO: How to support DMs in providing preferences in the context of interactive MOO methods? How these rules may be generated is discussed in Section 4.

We believe that, when leveraged, the explainable aspects of LEMOO methods can provide similar insights to a DM between the decision variables and objective functions as discussed in Section 1 (c.f. innovization). However, to the best of our knowledge, in none of the past works has the potential for explainability been explored in LEMOO methods to provide a DM with additional insights about the decision vectors in the computed solutions. Furthermore, past works exploring LEMOO have not provided software or other means for researchers and practitioners alike to start exploring and implementing LEMOO methods. Through the experiments conducted in Section 4, we will provide novel insights about LEMOO methods for others to consider when building their own LEMOO methods, and we will show how explainability can be combined with LEMOO methods to provide DMs with important insights about the MOO problem being solved. Thus, we give rise to the new breed of XLEMOO methods, where explainability and LEMOO have been combined to provide DMs with additional information to support better decision making when applying EMO methods. To help others get started with their own implementations and applications of XLEMOO, we provide the open source software framework, *the XLEMOO framework*, as a bedrock for researchers and practitioners to lean on in their future endeavors. This framework is discussed next.

3.3 The XLEMOO Framework

The XLEMOO (software) framework allows users to readily build and apply LEMOO methods to solve MOO problems as described in Section 3.1. The fitness function, the evolutionary operators in a Darwinian mode, and ML model in a learning mode can be easily switched or replaced with custom implementations. The user has also access to the various parameters present in Darwinian and learning modes. Although the framework is primarily built to accustom indicator-based methods in a learning mode, switching to another kind of EMO method is possible with moderately little effort. We have also taken care to document the framework, which eases its use. The framework is implemented in the Python language, which is a widely used language in data analysis, and the framework is aimed at users with a moderate proficiency in the language.

We have implemented the XLEMOO framework as open source software extending the DESDEO framework [Misitano et al. 2021]. The XLEMOO framework is available on GitHub[2] and Zenodo [Misitano 2023a]. To our knowledge, this is the first time a framework for implementing LEMOO methods has been made available to the public. The framework is also suitable for solving single-objective optimization problems. On top of the documentation, we have also provided a Jupyter [Kluyver et al. 2016] notebook for users to get quickly started utilizing the framework.[3]

In the following section, we give a detailed example of our own LEMOO method implemented in the described XLEMOO framework. We have utilized the implementation to run experiments and explore the effects of a learning mode on the performance of a LEMOO method, and also showcase the potential of the explanations emerging from the learning mode of the LEMOO method when the ML model utilized is interpretable.

4 IMPLEMENTATION AND EXPERIMENTS

In this section, we present in more detail the implementation of our LEMOO method utilizing the XLEMOO framework discussed in Section 3. The implementation and experimental setup are presented in Section 4.1. Then, in Section 4.2, we use our implementation to study the effects of the switching frequency and the H/L split in a learning mode on the performance of our method in numerical experiments. Last, we explore the potential for explainability offered by our method and discuss how the explanations (i.e., rule sets) have been generated in Section 4.3. The implementation discussed in this section is later utilized in Section 5 as an interactive MOO method, where the added benefit of explainability to a DM is showcased. The reader interested in the practical benefits offered by our method can skip this section and go to Section 5.

[2]https://github.com/gialmisi/XLEMOO/tree/article_v1.1
[3]https://xlemoo.readthedocs.io/en/article_v1.1/notebooks/Showcase.html

4.1 General Setup

We implemented our LEMOO method utilizing the framework discussed in Section 3.3. The following parameters were chosen in the method's Darwinian and learning modes. This is the same implementation that is utilized later in Section 5.

4.1.1 Darwinian Mode. We implemented a simple indicator-based EMO algorithm with the achievement scalarizing function (2) as the indicator (i.e., the fitness function), which computes a fitness value for computed solutions. The lower the fitness value, the closer the solution is to a supplied reference point $\bar{\mathbf{z}}$, reflecting how well a solution aligns with the preferences of a DM represented by the reference point. In (2), ρ and ε were set to 10^{-6} throughout the experiments. The population size was set to be $N^{\text{pop}} = 50$ to keep computation times feasible. The initial population was initialized using Latin hypercube sampling to achieve a uniform initial coverage of different decision vectors. Three evolutionary operators, as shown in Figure 2 (left), were chosen. For the crossover operator, simulated binary crossover [Deb et al. 1995] was utilized with a crossover probability of 1.0 and a crossover distribution parameter equal to 30; bounded polynomial mutation [Deb et al. 1995] was used as the mutation operator with a mutation probability of $\frac{1}{n}$, where n is the number of decision variables in a MOO problem, and a mutation distribution parameter equal to 20; the selection operator was defined such that after crossover and mutation, the N^{pop} best individuals were chosen to proceed to the next generation, keeping the population size constant in each generation. The population from each generation was saved into an archive. With these operators and their parameters, our method seemed to achieve acceptable results consistently, which motivated their choice. We utilized the EMO operators implemented in the `desdeo-emo` package [Misitano et al. 2021], version 1.4.1.

4.1.2 Learning Mode. We chose skope-rules implemented in the `imodels` Python package [Singh et al. 2021] as the ML model utilized in a learning mode to learn a rule set for binary classification. Our choice was motivated by the computationally fast training of skope-rules, and the interpretability of the generated rules (see Section 4.3 for examples of these rules). The minimum rule precision considered was set to be 0.1, the maximum number of estimators to be learned was set to 30, and we chose to bootstrap both samples and features. The rest of the skope-rules' parameters were kept at their default values as defined in the `imodels` package, version 1.3.6. We chose these values because we found them to work well from a performance perspective in our case. To train the ML model, we considered the populations from all past generations computed before the current learning mode iteration to maximize the available training data. Based on the fitness values of the individuals, we chose a top (best fitness values) and bottom (worst fitness values) percentage of the individuals from all past populations. Individuals with the best fitness values were close to the reference point in (2), whereas individuals with the worst values were farther away. When forming the H- and L-groups, only unique solution, based on their decision variable values, were considered. This was done to boost the learning performance of the ML model. Then, the ML model was trained to classify individuals into either a high-performing or a low-performing group.

The trained skope-rules model was then used to instantiate $\gamma^{\text{inst}} \times N^{\text{pop}}$ high-performing new individuals, where γ^{inst} is an *instantiation factor*, which was set to 10. We chose a relatively high instantiation factor to increase the changes of finding new high-performing solutions during the instantiation. The individuals were instantiated based on the rule sets learned by the ML model. Rules in the rule set were weighted based on their precision when instantiating new individuals: rules with a higher precision were used more often to generate new individuals than rules with a lower precision. Based on a rule, the decision variable values of instantiated individuals were

Table 1. Computed Payoff-table for the Vehicle Crash Worthiness Problem

Vehicle crash worthiness problem			
Minimized	f_1	f_2	f_3
f_1	**1,661.71**	8.30	0.071
f_2	1,675.45	**6.14**	<u>0.26</u>
f_3	<u>1,692.02</u>	<u>10.56</u>	**0.042**

The first column indicates which objective has been minimized on each row. The minimum values in each column for each problem are in bold, whereas the maximum values are underlined.

randomly generated according to the upper and lower limits extracted from the rule. If a rule did not describe an upper or lower limit for a variable, the upper and lower bound based on box constraints of the variable were used instead, respectively. We did not consider problems with function constraints on the variables, but implementing an instantiation strategy to account for function constraints as well is possible in the XLEMOO framework. The new instantiated individuals and the high-performing group were then combined, and the selection operator from the learning mode was used to select the best individuals, keeping the population size constant. The new population generated in the learning mode was also saved in the population archive.

4.1.3 MOO Problem Setup. In the experiments conducted, we considered two real-life based MOO problems. The first one is the *vehicle crash worthiness problem* discussed in Section 5. We will present the results for the vehicle crash worthiness problem later in this section but have included the results for the second problem in the appendix. We made this choice to maintain the clarity of the main text and to not inflate the number of figures present in the text. We utilized the problems as implemented in the `desdeo-problem` package [Misitano et al. 2021], version 1.4.6.

The second problem considered in the experiments, the *car-side impact problem* [Jain and Deb 2013], extended with a fourth objective from the work of Tanabe and Ishibuchi [2020], optimizes the crash safety of a car in case of a side impact. The original problem had three objectives to be minimized: the mass of the vehicle, the pubic force experienced by a passenger in case of a crash, and the average velocity experienced by the V-pillar withstanding the impact load from a crash. The fourth objective is the sum of the 10 constraints the decision variables of the problem are subject to. The seven decision variables of the problem, which model the thickness of the various car parts (in millimeters) that contribute to the crash safety of the vehicle, are subject to the following box constraints: $x_1 \in [0.5, 1.5], x_2 \in [0.45, 1.35], x_3 \in [0.5, 1.5], x_4 \in [0.5, 1.5], x_5 \in [0.875, 2.625], x_6 \in [0.4, 1.2]$, and $x_7 \in [0.4, 1.2]$.

Because the achievement scalarizing function (2) requires information about a problem's ideal and nadir points, we computed the approximations of these points by utilizing our LEMOO method. We ran the method solely in a Darwinian mode for 2,000 iterations, optimizing each objective function separately. The resulting objective vectors have been collected in Table 1 for the vehicle crash worthiness problem forming a payoff-table [Miettinen 1999]. In the payoff-table, the ideal point is located on the main diagonal, whereas the nadir point can be approximated by taking the maximum value from each column. Then, from the payoff-table, we can approximate the ideal and nadir points as $\mathbf{z}^{*}_{\text{crash}} = (1{,}600.0; 6.0; 0.038)$ and $\mathbf{z}^{\text{nadir}}_{\text{crash}} = (1{,}700.0; 12.0; 0.30)$. Notice that from the values shown in Table 1, we have chosen lower values for the ideal point components and higher values for the nadir point components than shown in the table. This was done to assure that the computed objective vectors during the experiments would fall between the ideal and nadir points in both problems, which is required in the achievement scalarizing function (2). We utilized the payoff-table method to compute the approximations for the ideal and nadir points because of the simplicity of the method, and because our experiments do not critically depend on the exact values of the points—it is sufficient that computed solutions fall between the points, and the points can also be updated during the solution process, if needed.

4.2 Exploring the Effects of the Learning Mode Frequency and the H/L Split on LEMOO Performance

4.2.1 Setup. In this first experiment, our goal was to find out the effect of the choice of the H/L split in the learning mode and the frequency of switching from a Darwinian mode to a learning mode on the performance of our LEMOO method. We chose 10 different H/L splits ranging from 5% to 50% in increments of 5%, and 11 different switching frequencies: 2, 4, 5, 8, 10, 20, 25, 50, 100, 200, and 500. For each combination of an H/L split and switching frequency, we ran our LEMOO method so that the total iterations would always sum to 1,000 (i.e., $N^T \times (N^D + N^L) = 1{,}000$). However, as mentioned in Section 3, we iterate the LEMOO in a learning mode always only once (i.e., $N^L = 1$) throughout the experiment. Each run was repeated 10 times for all problems considered to even out statistical fluctuations arising from the heuristic nature of the method. This meant that our LEMOO method was run 1,100 times in total for each problem considered. As for the reference points chosen in our fitness function (2), we chose $\bar{\mathbf{z}}_{\text{crash}} = (1{,}650.0; 7.0; 0.05)$ for the vehicle crash worthiness problem. The reference point was kept constant throughout the experiments so that the results could be compared.

For each run, we measured and collected the following quantities related to the populations in each of the iterations in Darwinian and learning modes: the decision vectors, the objective vectors, the fitness function values for each individual, the best fitness function value, the average fitness function value, the hypervolume (with the nadir point of the studied problem set as the point with respect to the hypervolume is computed), and the cumulative sum of the number of unique decision vectors. We measured the fitness function values to probe the performance of our LEMOO method in finding optimal solutions. Likewise, we measured the hypervolume and the cumulative sums to probe the variety of solutions in the populations. Since we are utilizing the scalarizing function in (2) to compute the fitness, we are not trying to approximate the whole PO set; instead, the search is focused on finding a small subset of the PO set residing near a reference point. It is therefore feasible to assume the subset of the approximate PO set to form a front of relatively continuous, connected, and smooth objective vectors, which makes the hypervolume and cumulative sums appropriate measures for the variety of the solutions in our case. In addition, solutions close to the reference point are assumed to have approximately a linear connection between the decision variables and objective vectors. The measured and collected quantities were saved for each run in files in a JSON format. Snakemake [Mölder et al. 2021] was used to manage the execution of the experiments and to ensure the reproducibility and transparency of them. The data generated in the experiments is openly available on Zenodo [Misitano 2023b], and the steps needed to reproduce the data are documented in the online documentation of the XLEMOO framework.[4]

4.2.2 Results and Analysis. For each combination of an H/L split and learning mode switching frequency, the data generated by the 10 repetitions of the experiment were aggregated by computing the mean and the standard deviation of each measured quantity. By visual inspection, the data seemed to be normally distributed with no significant outliers, which makes the mean and the standard deviation sensible statistical measures for the data under study.

[4] https://xlemoo.readthedocs.io/en/article_v1.1/introduction.html#reproducibility

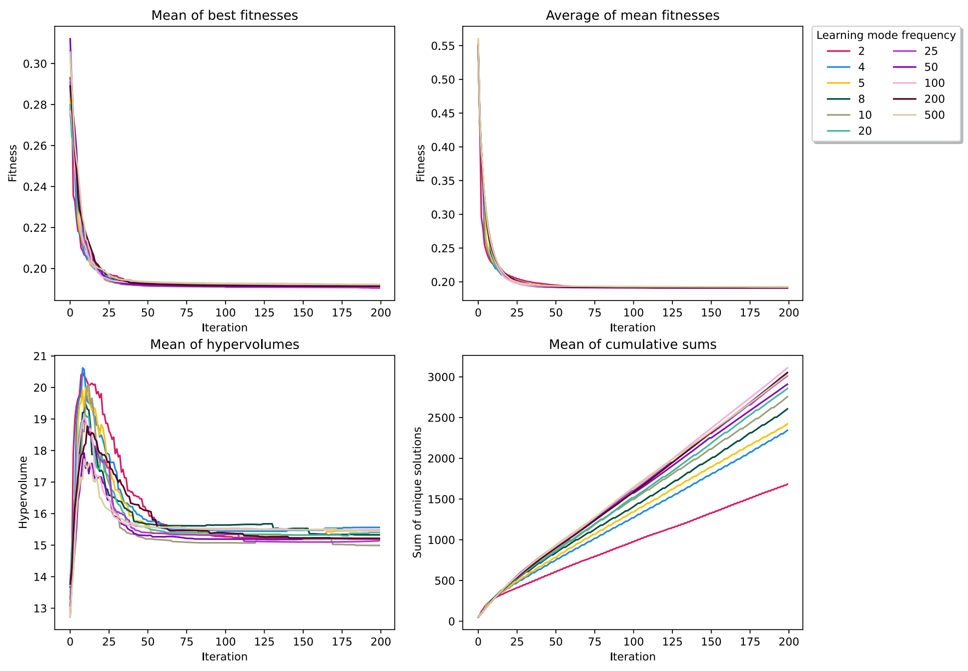


Fig. 3. The aggregated measured data for the vehicle crash worthiness problem after 200 iterations. The data is plotted for all the switching frequencies considered. The H/L split was kept constant at 20.

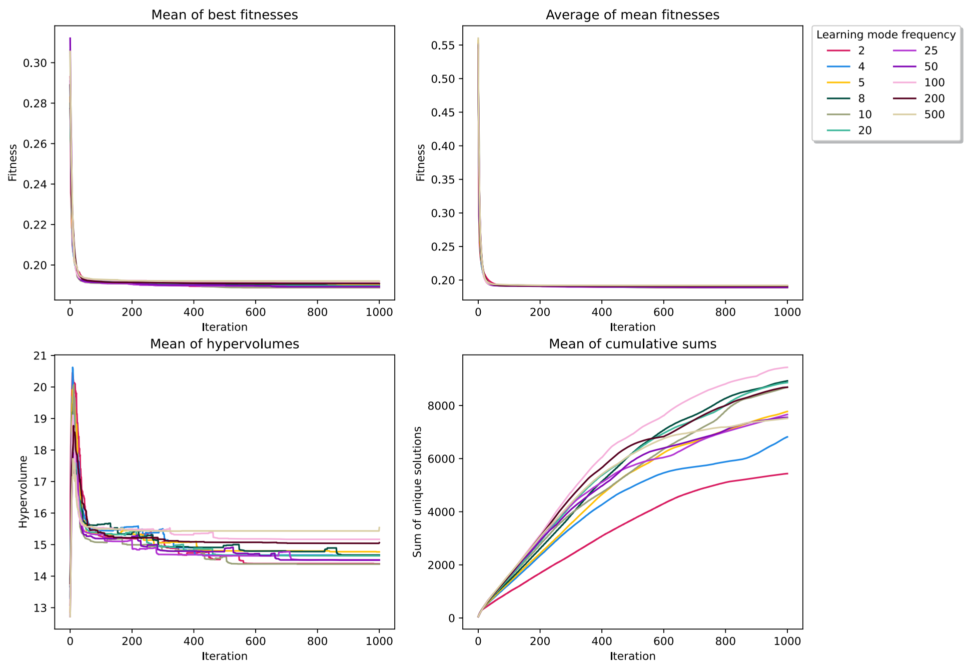


Fig. 4. The aggregated measured data for the vehicle crash worthiness problem after 1,000 iterations. The data is plotted for all the switching frequencies considered. The H/L split was kept constant at 20.

The aggregated data was then further studied by plotting the means of the four measured quantities for each iteration by varying the switching frequency and keeping the H/L split constant. Plots as shown in Figures 3 and 4 were generated and studied for all problems considered. In these figures, the H/L split was set to be 20 to serve as an example. It was quickly noticed that the mean of the best fitness value and the average of the fitness values converged before the 200th iteration in the problems considered, as can be seen by comparing Figures 3 and 4. The mean of the

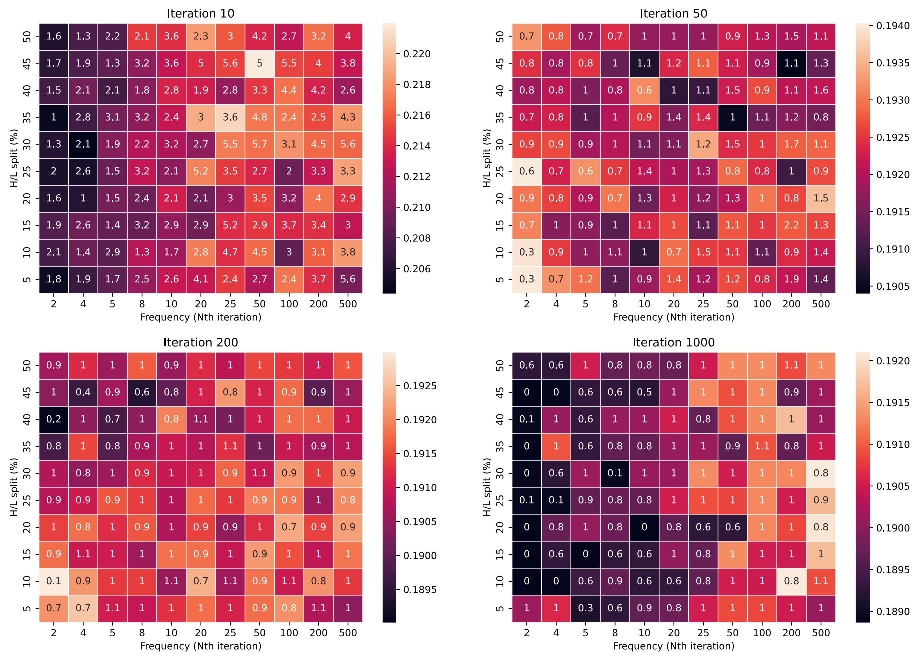


Fig. 5. The mean and standard deviations of the best fitness values for iterations 10, 50, 200, and 1,000 for the vehicle crash worthiness problem.

hypervolume seems to quickly settle before the 200th iteration but clearly keeps decreasing until the last iteration. The mean of the cumulative sums of unique solutions also keeps increasing until the last iteration. The changes after the 200th iteration in all measured quantities seemed approximately constant. The described behavior was true for all H/L splits. Based on these observations, it seems that exploring the early iterations in our LEMOO method in more detail than the later ones makes sense.

As a result of the preceding examination, we decided to study the aggregated data in more detail for iterations 10, 50, 200 and 1,000 by visualizing the data in heatmaps as shown in Figures 5, 6, 7, and 8 for the vehicle crash worthiness problem. In the heatmaps, each cell represents the values of the aggregated measures for an H/L splitting and switching frequency combination. The color of the cell indicates the mean, whereas the numerical quantity inside the cell indicates the standard deviation. The standard deviation is expressed as a percentage of the mean—that is, a quantity's standard deviation of 5, and a mean value of 100, would be expressed as a percentage with a value of 5%. The darker the cell, the lower its values, and the lighter, the higher. For the mean of the best fitness and the average (mean) of the mean fitnesses, a lower value (darker color) is better, whereas for the hypervolume and the cumulative sum of unique solutions, a higher value (lighter color) is better.

4.2.3 Observations. Here we present the results of the experiment and highlight the most obvious observations that can be made based on the data. Our observations are general to all problems considered in the experiment unless specified otherwise. We conclude by giving general remarks about the results. The reader interested only in the final remarks may skip to the end of the observations.

Observing the mean of the best fitness values found in the vehicle crash worthiness problem in Figure 5, we can see that in iteration 10, there is a darker hue on the left side of the heatmap. This means that, on average, the best fitness values found were lower with a lower switching frequency. This trend was also observed in other iterations but was not consistent throughout the problems

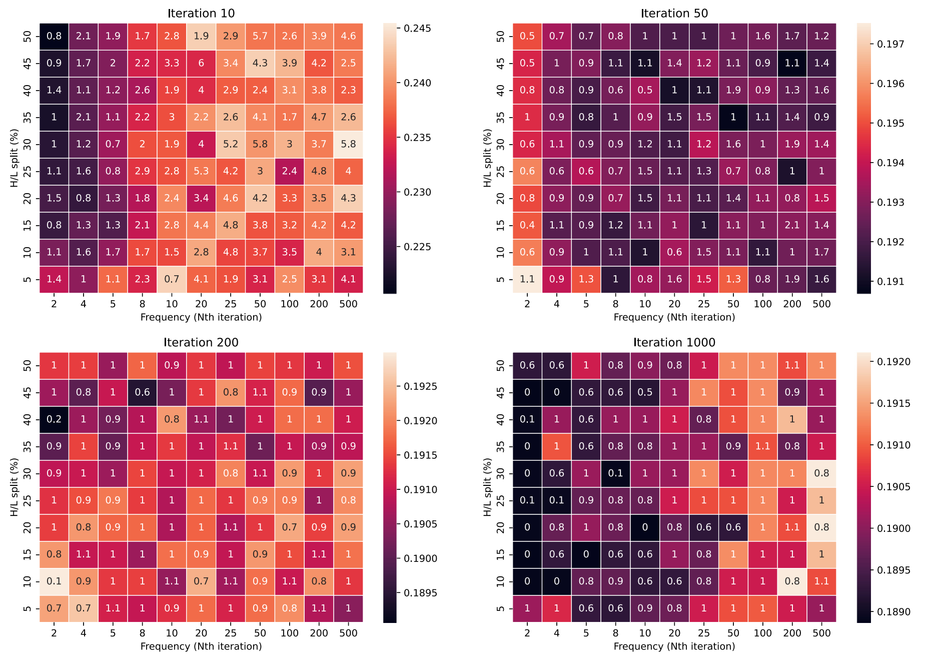


Fig. 6. The mean and standard deviations of the average of the fitness values for iterations 10, 50, 200, and 1,000 for the vehicle crash worthiness problem.

considered—that is, there is a darker hue on the left of the heatmap for iteration 1,000 in the vehicle crash worthiness problem, but this was not generally observed in all problems. We also notice that there is no clear gradient in the vertical directions of any of the heatmaps in Figure 5. This means that the choice of H/L split had no noticeable effect on the performance of our LEMOO method in finding individuals with a better best fitness value.

For the mean of the average fitness values, we notice a similar trend of a darker hue on the left of the heatmap of iteration 10 in Figure 6. A darker hue was again observed on the left of the heatmap for iteration 1,000 for the vehicle crash worthiness problem but was not observed in all problems. We also notice that with a combination of a low switching frequency and H/L split, there might even be a detrimental effect on our method's ability to improve the average of the mean fitness, as seen by the lighter hue of the very left of the heatmap for iteration 50 in Figure 6. We again observe no noticeable change in gradient in the vertical direction in any of the heatmaps, which means that the H/L split had little effect on our method's performance to improve the populations' average means values.

Observing the mean values for the computed hypervolumes in the vehicle crash worthiness problem show in Figure 7, we can in turn note that in the early iterations (10 and 50), there seems to be a lighter hue on the left indicting higher (better) values for the mean value of the hypervolume computed. For the vehicle crash worthiness problem, there seems to be a slightly darker hue in the heatmap of iteration 1,000, indicating lower (worse) hypervolume values, but this was not observed in all problems. And again, the choice of H/L split seems to have had little effect on the mean values of the hypervolume since there is no noticeable gradient in the vertical direction of any of the heatmaps.

If we study the mean values of the cumulative sums of unique solutions found in each iteration for the vehicle crash worthiness problem shown in Figure 8, it seems that in early iterations (10 and 50), with a combination of a low switching frequency and H/L split, our method seems to perform better when finding unique solutions. However, in the other iterations, there seems to be a lighter hue on the right of the heatmaps, indicating a higher (better) value for the cumulative

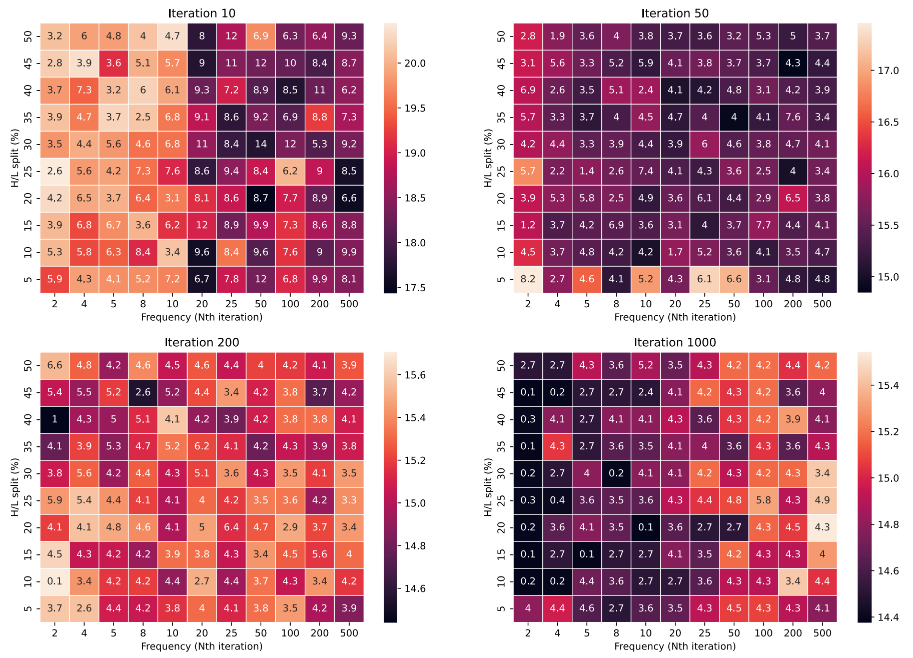


Fig. 7. The mean and standard deviations of the hypervolumes values of the population in iterations 10, 50, 200, and 1,000 for the vehicle crash worthiness problem.

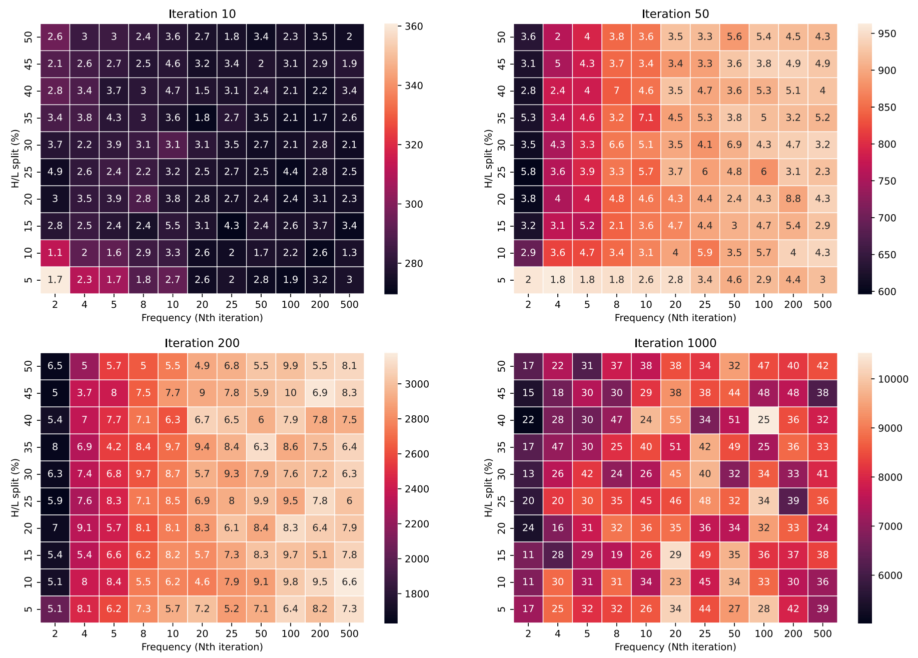


Fig. 8. The mean and standard deviations of the cumulative sums of unique solutions in iterations 10, 50, 200, and 1,000 for the vehicle crash worthiness problem.

sums. Unlike in the observations for the other measures, it seems that the choice of the H/L split might have a minor effect on our method's ability to find more unique solutions.

Last, observing the standard deviations of the measured quantities in the heatmaps in Figures 5 through 8, we can observe mostly low values between 0% to 10%, indicating a relatively low statistical variance of our results. However, an exception to this observation can be seen in iteration 1,000 for the cumulative sums of unique solutions in Figure 8, where the standard deviations raise occasionally above 20%, and in some cases even above 50%. Nevertheless, in general, the statistical fluctuations of our results were low.

Final Remarks. For each of the measured quantities (the best fitness value, the average of the mean fitness value, the hypervolume, and the cumulative sum of unique solutions), the most noticeable effects were observed in early iterations (10 and 50). The effects were positive. In all of the heatmaps, the effect were noticed only when the switching frequency was lower than the iteration represented by the heatmap—for example, in the heatmaps representing results for iteration 50, the noticed effects were present only for switching frequencies lower than 50. For higher switching frequencies in these cases, our LEMOO method has not engaged in a learning mode yet, meaning that the method has operated purely in a Darwinian mode. This is an important observation, as it is clear evidence that the addition of a learning mode has had an obvious effect on the performance of our LEMOO method.

4.3 Explanations in LEMOO

Here, we explore in more depth the explanations emerging from the use of an interpretable ML model in the learning mode of our LEMOO method. We also discuss how rule sets to support DMs can be extracted from our method. These rule sets provide insights about the decision vectors with the highest fitnesses found. We focus on the vehicle crash worthiness problem to give a concrete example of the explanations generated. A showcase on how the rules can benefit a DM in practice is given in Section 5.

4.3.1 Setup and Results. Here, we utilized our LEMOO method described in the previous sections. We chose the number of total iterations to be 200 and the switching frequency to be 20. The H/L split value was set to 20%. These choices were based on the results of the experiments conducted in Section 4.2. The other parameters of our LEMOO method were kept the same as described previously. We ran the method once with four different reference points used in the fitness function (2), which are listed in Table 2. The first reference point $\bar{\mathbf{z}}_1$ was the same as used in the previous experiments. There was no particular criterion in choosing the other reference points other than for the sake of providing more examples of possible rule sets emerging from our method to study.

Each rule in the rule sets was generated as follows. Rules were extracted from the skope-rules model from the final iteration of our LEMOO method. Each rule was then inspected, and the most accurate rule was selected. From the most accurate rules, upper and lower limits for each variable in the vehicle crash worthiness problem were searched for. If there was no rule describing a variable's lower or upper limit, the limit was taken from the final population computed in the LEMOO method—that is, the population was inspected for the variable, and the lowest or highest value found in the population was used as the lower or higher limit, respectively. If two rules extracted from the skope-rules had the same accuracy and described the same variable, then the rule with a stricter limit was selected—that is, if two lower (higher) limits were extracted from the rules, then the lower (higher) limit with a higher (lower) value was selected.

The rule sets generated are shown in Table 2 for each reference point considered. The accuracies of the rules from which the limits were extracted are also shown. An accuracy of −1 means that no rule described a particular variable's lower or higher limit, so the limit extracted from the population was used instead. In Table 2, we also report the lower and upper limits extracted from the final population to be compared with the respective limits extracted utilizing the rules from skope-rules.

To study the spread of the possible solutions computed based on the rules in the rule sets generated, we randomly generated 100,000 decision vectors based on the rules—that is, for each variable,

Table 2. Rule Sets Generated for the Four Reference Points Considered

Rule set 1: $\bar{\mathbf{z}}_1 = (1{,}650.0; 7.0; 0.05)$						
Variable	Lower (R)	Acc.	Upper (R)	Acc.	Lower (P)	Upper (P)
x_1	1.0	(0.485)	1.0	(1.0)	1.0	1.0
x_2	1.37958	(1.0)	1.43105	(1.0)	1.42731	1.42731
x_3	1.0	(0.468)	1.00002	(1.0)	1.0	1.0
x_4	1.0	(0.499)	1.00197	(1.0)	1.0	1.0
x_5	2.31826	(1.0)	2.58524	(1.0)	2.38275	2.38275
Rule set 2: $\bar{\mathbf{z}}_2 = (1{,}600.0; 8.0; 0.07)$						
Variable	Lower (R)	Acc.	Upper (R)	Acc.	Lower (P)	Upper (P)
x_1	1.0	(0.518)	1.00513	(1.0)	1.0	1.0
x_2	1.0	(0.478)	1.0	(1.0)	1.0	1.0
x_3	1.0	(0.404)	1.00004	(1.0)	1.0	1.0
x_4	1.0	(−1)	1.00428	(1.0)	1.0	1.0
x_5	1.0	(0.232)	1.0	(1.0)	1.0	1.0
Rule set 3: $\bar{\mathbf{z}}_3 = (1{,}700.0; 6.5; 0.18)$						
Variable	Lower (R)	Acc.	Upper (R)	Acc.	Lower (P)	Upper (P)
x_1	1.0	(−1)	1.00001	(1.0)	1.0	1.0
x_2	2.99992	(0.997)	3.0	(0.496)	3.0	3.0
x_3	2.56003	(1.0)	2.57444	(1.0)	2.56865	2.56929
x_4	1.0	(0.487)	1.00001	(1.0)	1.0	1.0
x_5	2.97299	(1.0)	2.99937	(−1)	2.99717	2.99937
Rule set 4: $\bar{\mathbf{z}}_4 = (1{,}695.0; 6.1; 0.04)$						
Variable	Lower (R)	Acc.	Upper (R)	Acc.	Lower (P)	Upper (P)
x_1	1.00001	(1.0)	1.00165	(1.0)	1.00001	1.00001
x_2	2.99559	(1.0)	3.0	(0.987)	3.0	3.0
x_3	1.0	(−1)	1.00001	(1.0)	1.0	1.0
x_4	1.08398	(1.0)	1.1352	(1.0)	1.10772	1.10772
x_5	2.99449	(1.0)	3.0	(−1)	3.0	3.0

The lower and upper limits extracted from the skope-rules and final population are listed in columns ‘Lower (R)’ and ‘Upper (R),’ respectively, whereas the limits extracted from the final population alone are listed in columns ‘Lower (P)’ and ‘Upper (P),’ respectively. The limits extracted from a skope-rules are followed by the respective rule’s accuracy listed in the ‘Acc.’ columns, where 1 is the highest possible accuracy (equal to 100%) and 0 is the lowest. If a limit was extracted from the final population instead, then the reported accuracy is −1.

we generated a random value between the lower and upper limits reported in the rule describing the variable. We then plotted a histogram for each set of randomly generated decision vectors showing the distribution of the vector’s fitness value. These histograms are shown in Figure 9.

The process of running our LEMOO method and generating the described rule sets can be found in a Jupyter notebook.[5] This notebook serves also as a good example of how our LEMOO method can be, and has been, implemented in practice.

4.3.2 Observations. Observing the rule sets in Table 2, we can make some observations. First, the rules extracted from skope-rules always describe a wider, or equal, range of possible decision

[5]https://xlemoo.readthedocs.io/en/article_v1.1/notebooks/How_to_extract_rules_example.html

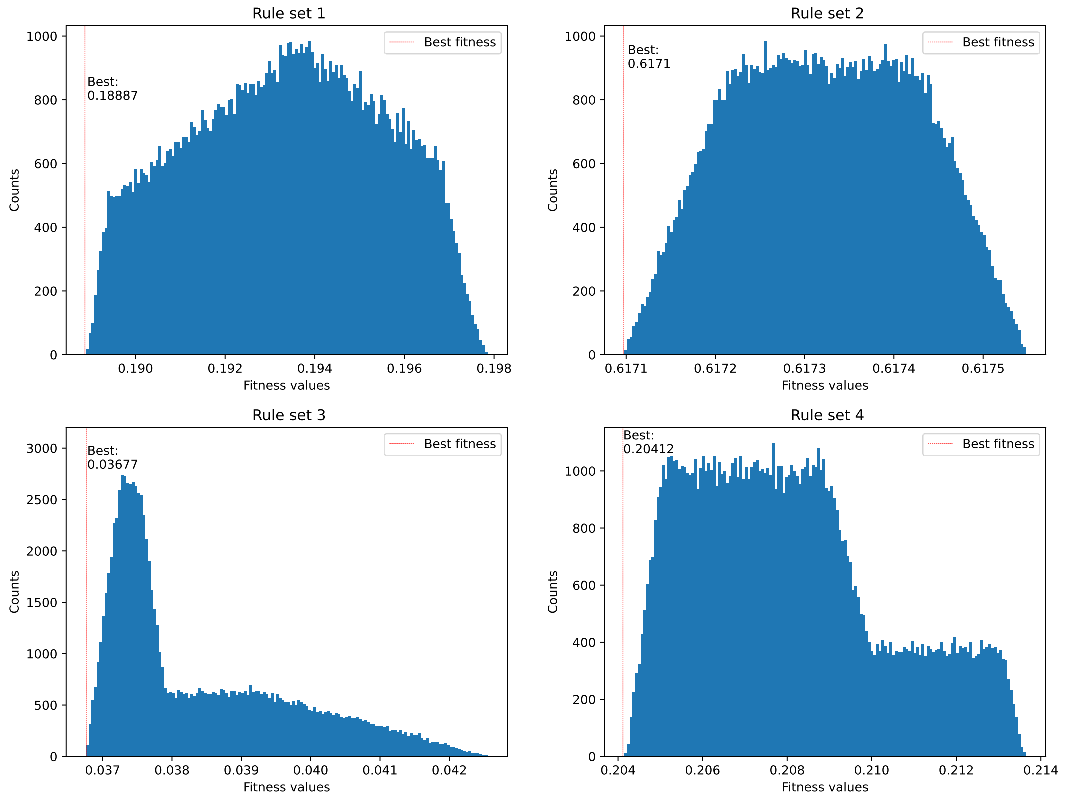


Fig. 9. The distributions of the fitness values of the decision vectors generated based on the rule sets extracted for the car-side impact problem. The best fitness function value found by our LEMOO method for each reference point is also shown in the plots. The number of samples in each histogram is 100,000.

variable values when compared to the rules extracted from the final population alone. Second, when the range is the same, or when there is no range—the upper and lower limits are the same—the accuracies of the upper or lower limits extracted from skope-rules are in the range $[0, 1)$. And third, in most cases, at least one rule extracted from skope-rules was found to describe the lower and upper limits of a variable.

Inspecting the histograms in Figure 9, none of the rule sets describe solutions that have fitness values better than the best fitness value found. This is expected since after 200 iterations, our LEMOO method was observed in Section 4.2 to have already converged in terms of best fitness value and the average of fitness values in a population. Nonetheless, the rules in the rule sets describe solutions with fitness values relatively close to the best. If we assume for the objective vector corresponding to the individual with the best fitness value to be close to the reference point considered when generating each rule set, we can also assume that the objective vectors corresponding to the decision vectors generated based on the rule sets to be close to the reference point. In the histograms for rule sets 1 and 2 in Figure 9, we can see an even distribution for the solution fitnesses, whereas in the histograms for rule sets 3 and 4, we can see a slight bias in the fitness values toward the best fitness value. Out of 100,000 solutions generated based on the rules in each rule set, none of the fitness values are significantly far from the best fitness values. This means that by varying the decision vectors according to the rules in the rule sets, we find solutions with respective objective vectors in the proximity of the objective vector with the decision vector corresponding to the best fitness value. This also means that near the reference points considered, we can approximate the relationship between objective function values and decision variables to be linear.

Final Remarks. The rule sets extracted from the interpretable ML model in our LEMOO method are clearly more informative about the possible ranges of decision variable values close to a solution of interest from the perspective of a DM when compared to the respective information extracted from the final population alone. We also saw that the solutions generated according to the rules in the rule sets are close to the original best solution. This can help a DM explore alternative solutions close to the best one. Since the rule sets were learned based on past populations as well, and not just the final one, we conclude that a better insight about the connection between decision vectors and objective vectors close to a solution of interest can be derived by considering the population history, and that utilizing interpretable ML is one possible approach to generate these insights.

5 SHOWCASE OF THE ADDED BENEFITS OF EXPLAINABILITY

In this section, we demonstrate in a proof of concept the benefits of the explanations extracted from our LEMOO method, and how they can support a DM in solving a MOO problem. More technical details on how the results have been computed and the rules extracted were given in Section 4, where we reported more detailed experiments to explore the performance and explanations provided by our LEMOO method, and presented the implementation of our method in more detail.

5.1 LEMOO as an Explainable Interactive Method

We demonstrate how our LEMOO method, which combines an indicator-based EMO and skope-rules, can be used as a reference point based interactive MOO method. The input is a reference point provided by a DM. The reference point is utilized in the scalarizing function in (2), which functions as the indicator in our method. Our method then computes a population of solutions from which the solution with the best fitness value is shown to the DM. In addition, the method outputs explanations, in the form of rule sets generated by skope-rules, describing the high-performing solutions providing the DM with additional insights about the best-performing solutions generated near the reference point from the perspective of decision variables. This idea is visualized in Figure 10. In addition, the DM has also the option to provide decision vectors, which are evaluated into objective vectors, to explore different solutions inspired by the insights provided by the explanations. This in turn can also provide support to the DM in providing further reference points in subsequent iterations. In practice, an *analyst* operating the method can aid the DM when evaluating decision vectors specified by them. But in our case, we have assumed the DM to be an engineer, who is familiar with the domain of the problem and its technicalities, including the meaning of the decision vectors.

As a showcase of how our method can be utilized as an interactive MOO method, we show three iterations with the author acting as the DM. We also comment on the insights provided to the DM. We consider the *vehicle crash worthiness problem* [Liao et al. 2008] to optimize the crash safety of a car. It has three objectives to be minimized: the frontal mass of the vehicle, the collision acceleration experienced by passengers in case of a full frontal crash, and the toe board intrusion in the case of an off-set frontal crash. The problem has five decision variables, which model the thickness of five reinforced members in the frontal structure of the car. They have box-constraints, where the thickness should be between 1 and 3 mm. The problem has no other constraints.

To support the DM in providing reference points, we assume that before the optimization process, the DM has been informed of the approximated ideal and nadir points of the vehicle crash worthiness problem: $\mathbf{z}^{*}_{\text{crash}} = (1{,}600.0; 6.0; 0.038)$ and $\mathbf{z}^{\text{nadir}}_{\text{crash}} = (1{,}700.0; 12.0; 0.30)$. Details on how these points were approximated are given in Section 4. The explanations provided to the DM during the optimization process are the rule sets shown in Table 3. These describe the solutions with

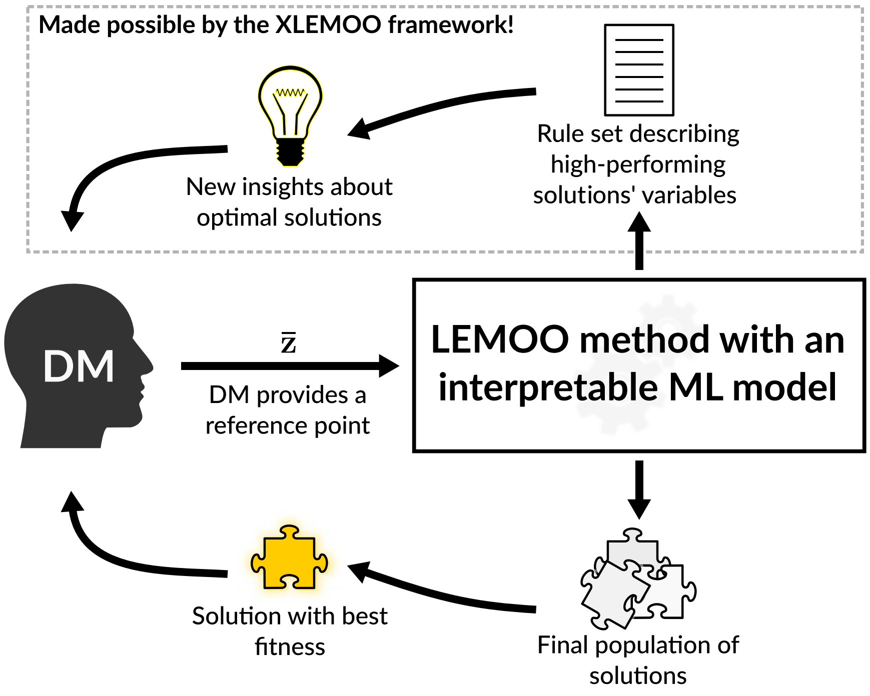

Made possible by the XLEMOO framework!
New insights about optimal solutions
Rule set describing high-performing solutions' variables
DM
$\bar{\mathbf{z}}$
DM provides a reference point
LEMOO method with an interpretable ML model
Solution with best fitness
Final population of solutions

Fig. 10. An example of how our LEMOO method can be used as an interactive MOO method.

the best fitness values found in each iteration by means of lower and upper limits for each decision variable. Each limit has also an accuracy associated with it. The rule sets are extracted from the skope-rules model used in our method. A more detailed description of how the rule sets have been derived is given in Section 4.3. Next, we describe the three iterations and show how our LEMOO method can support a DM in solving the vehicle crash worthiness problem and gaining new insights.

Iteration 1. In the first iteration, the DM provided an optimistic reference point close to the ideal point of the problem: $\bar{\mathbf{z}}_1 = (1{,}610.0; 6.2; 0.041)$, which resulted in the best solution (as defined earlier) $\mathbf{x}_1 = (1.00887; 3.0; 1.0; 1.37784; 1.00168)$ with the objective vector $\mathbf{z}_1 = (1{,}669.28; 7.69556; 0.08126)$. The DM was also shown the rule set derived shown in *Iteration 1* in Table 3. The DM noticed that the objective vector was worse than the reference point in each component. Looking at the rule set in Table 3 for *Iteration 1*, he noticed that the thickness of the second and fourth member could perhaps be lowered while keeping the result otherwise similar to decrease the mass (first objective) by a small amount. Driven by these insights about the optimal solutions, he then tried evaluating the problem with the decision vector $\mathbf{x}'_1 = (1.00887; 2.99985; 1.0; 1.36258; 1.00168)$, which resulted in the objective vector $\mathbf{z}'_1 = (1{,}669.18; 7.66459; 0.08166)$. Although the effect on the mass was minimal, he noticed a more significant improvement in the collision acceleration experienced by passengers (second objective). After learning of the minimal effect changing the variables had on the mass, and the too optimistic value he had set for the mass in the aspiration level of $\bar{\mathbf{z}}_1$, he decided to explore the mass by providing a new reference point instead.

Iteration 2. In the next reference point, the DM decided to increase the aspirations levels for the mass, keep the collision acceleration the same as in $\mathbf{z}'_1$, but increase the aspiration level for the toe board intrusion (third objective) in hopes of achieving a lower mass than in the previous iteration. Thus, he gave the reference point $\bar{\mathbf{z}}_2 = (1{,}680.0; 7.66459; 0.07)$, which resulted in the best solution $\mathbf{x}_2 = (1.0; 3.0; 1.0; 1.25473; 2.99988)$ with the corresponding objective vector $\mathbf{z}_2 = (1{,}677.23; 7.61449; 0.068647)$. The DM noticed that the objective values in the objective vector were all better than in the reference point, but the mass was now clearly too high. After inspecting the rules in the rule set in Table 3 for *Iteration 2*, he noticed that the fourth variable once again had a large range. The DM made a note of this and decided not to try other decision variable values based on the rules in the rule set because the ranges for the variables were narrow and did not encourage the DM to explore the solution in this iteration any further.

Iteration 3. The DM was satisfied with the value of acceleration experienced by passengers (second objective) in the previous objective vector, $\mathbf{z}_2$; however, he still wished to find a solution with a lower mass (first objective) and was ready to trade off for a slightly larger value for the toe board intrusion (third objective). Thus, the third reference point was $\bar{\mathbf{z}}_3 = (1{,}670.0; 7.61449; 0.085)$, which led to the best solution $\mathbf{x}_3 = (1.00286; 2.99999; 1.00000; 1.30009; 1.03374)$ with the objective vector $\mathbf{z}_3 = (1{,}668.83; 7.53771; 0.083098)$. The DM was happy to see a lower mass (first objective) and a lower acceleration (second objective) than in $\mathbf{z}_2$. After inspecting the rule set of *Iteration 3* in Table 3, the DM noticed again that the fourth variable of the problem had the largest range of

Table 3. Three Iterations Featured in Our LEMOO Method's Showcase

Iteration 1
Reference point: $\bar{\mathbf{z}}_1 = (1{,}610.0; 6.2; 0.041)$
Best objective vector: $\mathbf{z}_1 = (1{,}669.28; 7.69556; 0.08126)$
Best decision vector: $\mathbf{x}_1 = (1.00887; 3.0; 1.0; 1.37784; 1.00168)$

Variable	Lower	Acc.	Upper	Acc.
x_1	1.00887	(−1)	1.00973	(1.0)
x_2	2.99985	(1.0)	3.0	(−1)
x_3	1.0	(0.496)	1.0	(0.429)
x_4	1.36258	(1.0)	1.37986	(1.0)
x_5	1.00168	(−1)	1.0028	(1.0)

Iteration 2
Reference point: $\bar{\mathbf{z}}_2 = (1{,}680.0; 7.66459; 0.07)$
Best objective vector: $\mathbf{z}_2 = (1{,}677.23; 7.61449; 0.068647)$
Best decision vector: $\mathbf{x}_2 = (1.0; 3.0; 1.0; 1.25473; 2.99988)$

Variable	Lower	Acc.	Upper	Acc.
x_1	1.0	(−1)	1.00002	(1.0)
x_2	2.99979	(1.0)	3.0	(0.496)
x_3	1.0	(0.993)	1.0	(1.0)
x_4	1.23885	(1.0)	1.25545	(1.0)
x_5	2.99986	(1.0)	2.99988	(0.997)

Iteration 3
Reference point: $\bar{\mathbf{z}}_3 = (1{,}670.0; 7.61449; 0.085)$
Best objective vector: $\mathbf{z}_3 = (1{,}668.83; 7.53771; 0.083098)$
Best decision vector: $\mathbf{x}_3 = (1.00286; 2.99999; 1.00000; 1.30009; 1.03374)$

Variable	Lower	Acc.	Upper	Acc.
x_1	1.00184	(0.996)	1.00558	(1.0)
x_2	2.7135	(1.0)	3.0	(0.48)
x_3	1.0	(0.517)	1.00011	(1.0)
x_4	1.21741	(1.0)	2.12628	(1.0)
x_5	1.02602	(1.0)	1.04323	(1.0)

The reference point provided by the DM, the decision and objective vector corresponding to the best solution, and the rules derived from our method are shown for each iteration. The lower limits are shown in the 'Lower' column and the upper limits in the 'Upper' column. Each limit is followed by its accuracy ranging from 0 (completely inaccurate) to 1 (completely accurate). Refer to Section 4.3 for further details on how the rule sets have been generated.

values according to the rules—this time noticeably larger than in the previous iterations. Although the DM was already happy with the solution found, the insights from the rule set inspired him to modify the decision vector according to the rules in hopes to decrease the mass by lowering the thickness of the members represented by the decision variables x_1, x_2, x_4, and x_5. This led to the modified decision vector $\mathbf{x}_3{}' = (1.00184; 2.7135; 1.00000; 1.21741; 1.02602)$, with the corresponding objective vector $\mathbf{z}_3{}' = (1{,}667.49, 7.54346, 0.094955)$. The DM was quick to note that lowering the thicknesses did not have a significant effect on the mass (first objective) or acceleration (second objective), but it did noticeably increase the toe board intrusion (third objective), which was too large a tradeoff for the DM to make.

In the end, the DM chose $\mathbf{x}_3$ as the final solution since he was most satisfied with the corresponding objective values. However, the DM also learned that changing the thicknesses of the five reinforced members in the frontal structure of the car represented by the decision variables had, in general, a surprisingly small effect on the mass of the car and a more noticeable effect on the toe board intrusion. Furthermore, the DM observed that the rule sets include larger ranges for variable x_4, as indicated in Table 3. The DM thought that this might diminish the impact of the variable on the values of the objective function, particularly within his region of interest. The DM thought that it may be a good idea to revisit the problem formulation to investigate this last point further before making any decisions based on the final solution.

Final Remarks on the Showcase. As we saw, thanks to the explanations (rule sets), the DM got important insights regarding the connections between decision variables and objective vectors, and he got some support to provide new reference points. Moreover, the insights made the DM question the formulation of the MOO problem, since changing the variables did not have the effect he expected. This caused the DM to postpone any decision based on the final solution found. The insights gained from the explanations (rule sets in Table 3) caused the DM to be more critical about the solutions and also explore the decision space of the problem, which, in the end, affected the whole decision making process.

6 DISCUSSION

In this section, we discuss the results of the experiments conducted in Section 4 and the results' main implications in regard to our current work and future research. We also consider the implications of the practical showcase discussed in Section 5. We discuss the results of the experiments related to the performance of our LEMOO method in Section 6.1, and the results regarding the explanation aspects of our method and the showcase in Section 6.2. We then discuss the further potential of the explanations in the context of LEMOO in Section 6.3. Finally, we conclude by discussing the general implications of our work in Section 6.4.

6.1 On the Performance of LEMOO

Based on the observations made in regard to the performance of our LEMOO method in Section 4.2, we can draw the following conclusions. First, a learning mode has a clear positive contribution to the performance of our LEMOO method in finding individuals with better fitness values if the frequency of switching to the learning mode is not too frequent nor too infrequent. This can help a LEMOO method converge faster if compared to a method without a learning mode. However, if the switching is too frequent, the population might not include enough individuals with clearly better fitness values, which makes it difficult for the ML model in the learning mode to be able to build a valid hypothesis for the high-performing group. In turn, if the learning frequency is too infrequent, the populations can become too homogeneous, which again makes it hard for an ML model to learn a distinction between a high-performing and low-performing individual. Interestingly, the choice of an H/L split seems to have little to no effect on the performance of our LEMOO method unless combined with a low switching frequency, in which case it can have a detrimental effect on the performance. The fact that LEMOO offers clear, although not immense, benefits in term of search performance of finding solutions with a better fitness value is in line with the findings reported in previous studies exploring LEMOO methods, which were discussed in Section 2.3. This gives us confidence in the validity of our own experiment as well. But in these previous works, the effect of the switching frequency or H/L split was not studied in detail.

The second conclusion we can make is that a low switching frequency can boost the diversity of the populations found in our LEMOO method, but only during early iterations (up to 200 out of 1,000 in our case). The observations for the computed hypervolumes and cumulative sums in Section 4.2 are the evidence supporting this claim. This makes sense if we take into account the first conclusion we made. In other words, because with a lower switching frequency, our LEMOO method converges faster, and it also means that the populations become saturated with more similar individuals in earlier iterations, leading to a decrease of the learning mode's performance in the hypothesis forming. We did, however, observe that a low H/L split combined with a low switching frequency can lead to a slight boost in the hypervolumes and cumulative sums of the populations, but if we also take into the account the observation that this same combination of frequency and split can have a detrimental effect on the fitnesses of the individuals found, the tradeoff between finding good fitness values and a diverse population with a low frequency and split becomes evident. The potential of LEMOO boosting the diversity of computed populations was also observed in the works discussed in Section 2.3; however, the potentially detrimental effect of a low switching frequency was neither reported nor studied, or both, in these previous works.

It is important to also mention the limitations of our first experiment. First, we only considered two MOO problems in Section 4.2, we did not vary any parameters of the Darwinian mode of our method, we did not vary the ML model used in the learning mode of our method, we did not consider other fitness functions than the scalarizing function in (2), and we did not vary the parameters of the ML model used. We did, however, explore these aspect during internal testing. We chose to report the results for only two problems because we found no significant differences in the results for other problems tested. This was also true for the fitness function chosen—we tried other scalarizing functions as well with no significant changes in the performance of our LEMOO method. Not surprisingly, the parameters in the Darwinian mode of our method did have a noticeable effect on the performance, but the results showed still similar trends when compared to what we reported in this study. The choice of our ML model to be skope-rules was mainly motivated by its superior performance in finding rules. We compared it to a decision tree (C4.5 [Salzberg 1994] and CART [Breiman 2017]), boosted rule set [Freund and Schapire 1997], and slippery rule set [Cohen and Singer 1999] but simply found their performance subpar compared to skope-rules. One of the main factors in choosing an ML model, apart from its interpretable nature, is its performance. It must be fast enough so that training it and generating new solutions is comparable to the time it takes for the LEMOO method to finish iterating in a Darwinian mode. Otherwise, the learning mode of a LEMOO method might be too slow to justify its inclusion. In our setting, we found skope-rules to be able to find useful rules in a feasible time, which ultimately led us to choose it as our ML method. The issue of exploring other ML models is also technical since the trained models are used to generate new data, which is an atypical application of the models. This means that it requires extra effort to develop the necessary utilities for the models to be utilized in a LEMOO method. In our XLEMOO framework, we have provided the necessary utilities to use the aforementioned ML models. That being said, our goal was to explore the effect of the H/L split and the switching frequency to a learning mode in our LEMOO method.

We have not compared our numerical results to previous works. This is because similar works with comparable results and reproducible experiments do not exist according to our best knowledge. However, we have taken care to report our results in a way that is reproducible so that future works may be compared to ours. Because we propose our current method to be used as an interactive MOO method, challenges arise since comparing interactive methods to each other is an open research question [Afsar et al. 2021], which is further complicated by the inclusion of explanations. The best works to compare our method to would be the ones presented in the work of Corrente et al. [2021] and Misitano et al. [2022], since they both discuss an interactive MOO method with explanations. However, there is no established way to compare such methods. To best compare these methods, we suggest that in a future study, experiments with human participants should be conducted to compare the interactive methods (as suggested by Afsar et al. [2022]), where the usefulness of the explanations to a human user is also assessed. That being said, our current results can be used as a baseline when developing new and more advanced LEMOO methods, and as an initial guideline when figuring out the H/L splitting ratios and switching frequencies, for instance.

6.2 On the Explanation Aspects of LEMOO

In Sections 4.3 and 5, we studied the potential of the generated explanations in our LEMOO method when the ML model employed in a learning mode is interpretable. In light of the showcase and the observations made, we conclude the following. First, it is evident that the rules extracted from the skope-rules provide much more potential insight for a DM about the decision variable values near a reference point if compared to the rules extracted from the final population alone. It was also observed that certain decision variables were clearly highlighted as being important when different reference points were used. This importance was underlined by the fact that rules defining a sensible—as in not being too narrow—range were found only for a couple of variables in each rule set. This can help a DM learn about the MOO problem by helping them build an understanding of which decision variables could be varied in the final solution while not steering too far from a point of interest (i.e., the reference point). This can guide the DM in providing further preferences, or in making a final decision based on the solutions found, as we saw in the showcase.

In addition, the accuracies of the rules in each rule set observed can prove to be useful for a DM. When a rule defines a very narrow range, or no range at all, if the accuracy of the lower or upper limit is low, it can encourage the DM to explore solutions beyond these more inaccurate ranges. This can help the DM explore regions in the decision space of the MOO problem that could otherwise be left unexplored but still potentially contain solutions of interest.

We believe that our LEMOO method can be best utilized as an interactive method, as was done in Section 5.1, where a DM provides multiple reference points in subsequent runs of our method. This way, the explanations may be exploited to their fullest by the DM helping them learn about the MOO problem and explore the solution space.

Our experiments in Section 4.3 were of course subject to the same limitations as discussed in Section 6.2. Furthermore, to better validate the actual usefulness of the rule sets generated based on our LEMOO method in an interactive setting, experiments with real DMs should be conducted. Although rule sets are, in general, easily interpretable by humans [van der Waa et al. 2021], it would be interesting to compare different kinds of explanations in an interactive MOO context. The way explanations are communicated to the DM in a MOO is also an unexplored area (e.g., how to best visualize them?). Empirical experiments, such as the one presented by Afsar et al. [2022], could be adapted to explore how explanations, and their usefulness, are perceived by DMs.

6.3 Further Potential of Explainability in LEMOO

In the showcase given in Section 5, we saw how the explainability provided by our LEMOO method was able to offer support to a DM in gaining insights of the connection between decision variables and objective vectors in providing new reference points, and in assessing the formulation of the MOO problem. For providing new reference points, it is also possible to present the high-performing individuals in the final population of our method to the DM in the objective space of the problem, effectively providing the DM with not only further solution candidates but also a description of the candidates in the objective space. Showing the high-performing solution candidates in the objective space can support a DM in providing further reference points as well, which addresses the open issue in interactive MOO methods—that is, the lack of supporting DMs in providing preferences during an interactive solutions process.

Although a more technical DM may appreciate the implications of explainability regarding a MOO problem formulation, the explanations can also offer valuable information to an analyst formulating the MOO problem. An analyst can explore a formulated problem by trying to generate an approximation of its whole PO set of solutions instead of just a subset near a reference point, as we have done in our current work. A whole set can be approximated by choosing the indicator appropriately in an indicator-based method, or the EMO method in the LEMOO approach can be switched to some other method altogether. Nonetheless, assuming an indicator-based method with rule-based explanations, an analyst can gain important insights about the nature of the whole approximated PO set. For instance, the rules may reveal that in the computed approximation, the ranges of the decision variables may differ significantly from the box-constraints of the problem. In turn, this can cause the analyst to revisit the MOO problem formulation by adjusting the constraints on the variables leading to a computationally more efficient problem formulation (achieved by limiting the search space). Rules significantly differing from the box-constraints of the MOO problem may also indicate that a search process for optimal solutions has been stuck on a certain part of the PO set of solutions—a local minima, for instance—which can be the case with discontinuous PO solution sets. Thus, explanations can enhance the process of formulating and validating MOO problems and assessing the effectiveness of the optimization methods chosen, for example, when modeling MOO problems in collaboration between domain experts and analysts as described in the work of Afsar et al. [2023].

In the example given in Section 5.1, we saw how a single solution could be accompanied by descriptive rules describing similar solutions near it in terms of decision variables. This can also facilitate communicating the found solution to other stakeholders, improving the transparency of the decision making process. This can help DMs in justifying their decisions to not just stakeholders but to other DMs as well, such as in a group decision making process [Lu et al. 2007]. For instance, explanations can offer negotiation support between two or multiple DMs with different preferences. It is possible that different objective vectors may share similarities in the decision space, which can be unveiled by the kind of rule sets discussed in this section. This can support multiple DMs in finding a compromise solution by considering decision variable values.

Thus far, we have considered the usefulness of explanations to a DM or analyst based on the final population and the history of populations preceding it. As we saw in Section 4, the ML model used, in our case skope-rules, was able to boost the search process for optimal solutions during the optimization process based on intermediate populations, and we may also extract the explanations generated on these intermediate populations during the optimization process to provide important insights to a DM. Following the NAUTILUS philosophy[6] [Miettinen and Ruiz 2016] and applying it to the interactive method discussed in Section 5.1, we can choose to iterate the LEMOO method only a few times and produce a suboptimal population of solutions, and show the high-performing solutions of this population to the DM. This allows the DM to provide new preferences, which can then be used in the indicator (in our case, the achievement scalarizing function in (2)) to steer the

[6]Starting an interactive process from the nadir point of a MOO problem and iteratively improving upon it until a solution on the PO set is reached.

optimization process toward solutions of interest from the point of view of the DM before actual convergence is reached in the LEMOO method. This can promote the exploratory aspects of the LEMOO method when employed as an interactive MOO method, and enables the DM to make a tradeoff free decision during the interactive process. Unlike existing NAUTILUS methods, applying the same philosophy to a LEMOO approach also provides information on the intermediate decision variable values, which are not available in NAUTILUS methods. This can make the comparison of intermediate solutions more meaningful from the perspective of the DM.

Although we have not explored the ideas discussed in this subsection explicitly in our current work, the XLEMOO framework described in Section 3.3 enables researchers and practitioners to readily start exploring the discussed ideas and more. Our current work provides a proof of concept on the potential of explainability in the context of LEMOO promoting the overall paradigm of explainable MOO.

6.4 General Implications of Our Study and Moving Forward

It is perhaps a little surprising that LEMs have not been explored more in the context of EMO methods given the potential shown by our and past studies of LEMOO methods. The fact that a lot of individuals are generated in the many populations of an EMO method makes the application of ML models, which usually need significant amounts data to perform well, not only possible but perchance even natural.

Based on our study, a simple addition of a learning mode to an indicator-based EMO method could work as a niching operator to increase the diversity of individuals, or to boost the search performance of the method when searching for individuals with better fitness values. By taking into account the whole population history, a learning mode could also be a way to implement a global search step in an EMO method. Moreover, as long as the populations generated in an EMO method are saved in an archive, the same procedure to generate rule sets described in Section 4.3 could be applied to any EMO method, making the method more explainable. Of course, mixing EMO methods with ML does come with an increase in computation cost, but as we have shown, even simpler interpretable ML models, such as skope-rules, can work when utilized in a learning mode.

Going forward, it would be interesting to see the concept of LEMs explored in other types of EMO methods, such as in dominance-based and decomposition-based methods, and exploit their unique nature. An ML model could be used to learn and explain the dominance rankings in a dominance-based method, or to learn and explain different decomposition strategies in decomposition-based methods. In addition to supporting DMs, these explanations could serve the researchers or analyst operating and implementing the methods to build a better understanding of which parameters to choose to achieve the best possible performance. If explainability is not a goal, then exploring the potential of the more powerful ML models, such as deep neural networks and support vector machines, could also be explored further in LEMOO methods.

Our study was also limited to rule-based explanations. In future studies, exploring different kinds of explanations, such as causal and counterfactual explanations [Molnar 2022], would be interesting. Moreover, explaining aspects of a MOO problem other than the characteristics of decision vectors near a reference point should be studied further. For instance, explaining the connection between preferences and computed solutions, as was done in the work of Misitano et al. [2022] for reference point methods, could be pursued in the future studies of LEMOOs.

The future directions listed here, and especially in Section 6.3, are nothing but the tip of the iceberg. However, the XLEMOO framework discussed in Section 3.3 provides other researchers and practitioners alike with a solid starting point to explore the preceding ideas and more. Because our framework is openly available and implemented as open source software, everyone is free to utilize and apply it, or extend it for others to use as well. This highlights the importance of providing similar, open source, and openly available frameworks in future studies as well, and collaborating on open source software to promote the openness and renewal of research.

7 CONCLUSION

We explored the potential of applying LEMs to solve MOO problems through our implementation of a LEMOO method. We explored the performance aspects of our LEMOO method and showcased the potential it can have in supplying a DM with additional explanations providing them insights about the characteristics of decision variables near a solution of interest. We also discussed and provided the openly available XLEMOO framework for others to utilize and implement their own LEMOO methods with explanations.

Naturally, our work came with limitations, as discussed in Section 6, but also showed great potential in terms of future research. LEMs are, without a doubt, an understudied area in the field of EMO. Furthermore, when an interpretable ML model is utilized in a LEM's learning mode, we have the potential access to more insights about the characteristics of the populations generated during the course of an EMO method. These insights can be expressed in the form of explanations, as we did. To help the rest of the EMO community pursue their own research in studying the application of LEMs in an EMO context, we provide the discussed XLEMOO framework as openly available open source software.

Combining ideas of LEMs with ideas from EMO into LEMOO methods has vast potential to unveil new and exciting ideas in the field of MOO. Moreover, utilizing interpretable ML models in a learning phase in a LEMOO method has the potential to unlock additional insights thanks to the vast number of individuals generated in a typical EMO setting. By leveraging these insights into explanations, we have taken the first step toward establishing a new paradigm in the field of MOO: explainable and learnable MOO. It is our hope that our work will inspire and enable future studies and research to help develop this new field further.

A APPENDIX

Here, we present the main results of the experiments conducted in Section 4.2 for the car-side impact problem. The ideal and nadir points of the problem were approximated from a payoff-table (Table 4). The reference point used for the car-side impact problem was $\bar{\mathbf{z}}_{\text{car-side}} = (20.0; 3.5; 11.0; 0.1)$. The plots showing the aggregated measured quantities as a function of the switching frequency, are shown in Figures 11 and 12. The heatmaps for the aggregated measured quantities are shown in Figures 13 through 16. Refer to Section 4.2 for further details.

Table 4. Computed Pay-off Table for the Car-Side Impact Problems

Car-side impact problem				
Minimized	f_1	f_2	f_3	f_4
f_1	**15.58**	<u>4.43</u>	<u>13.09</u>	2.82
f_2	36.71	**3.59**	11.87	14.19
f_3	<u>39.05</u>	4.05	**10.61**	<u>29.81</u>
f_4	23.72	4.30	12.91	**0.14**

The first column indicates which objective has been minimized on each row. The minimum values in each column for each problem are in bold, whereas the maximum values are underlined.

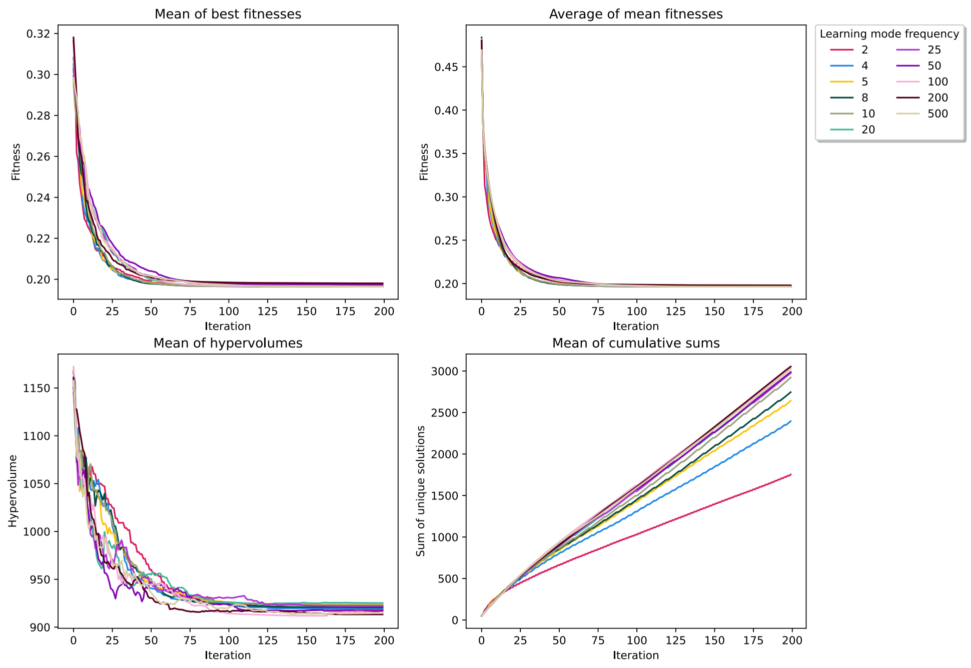


Fig. 11. The aggregated measured data for the vehicle crash worthiness problem after 200 iterations. The data is plotted for all the switching frequencies considered. The H/L split was kept constant at 20.

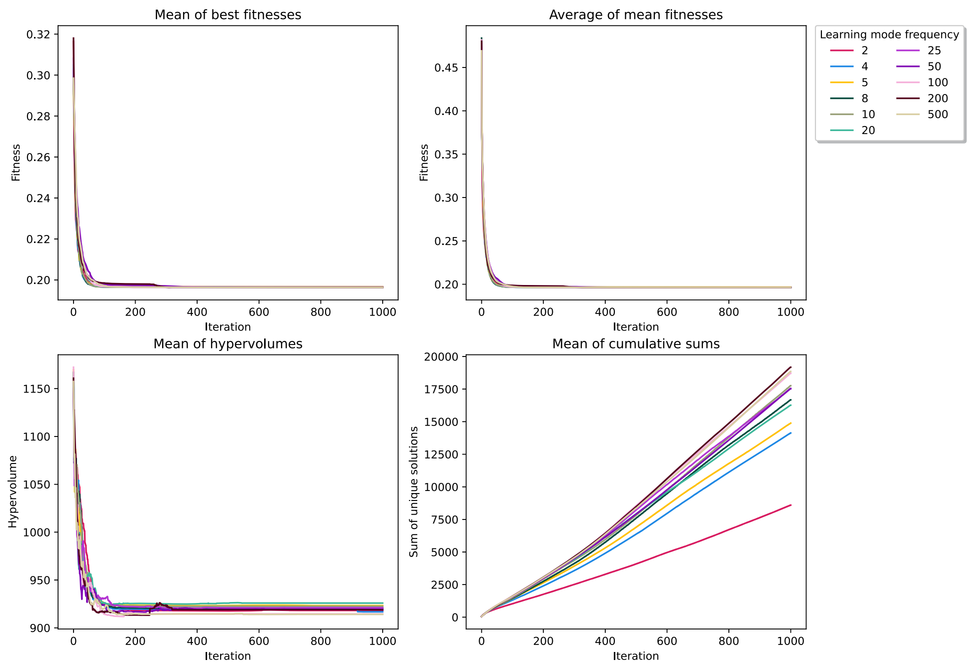


Fig. 12. The aggregated measured data for the car-side impact problem after 1,000 iterations. The data is plotted for all the switching frequencies considered. The H/L split was kept constant at 20.

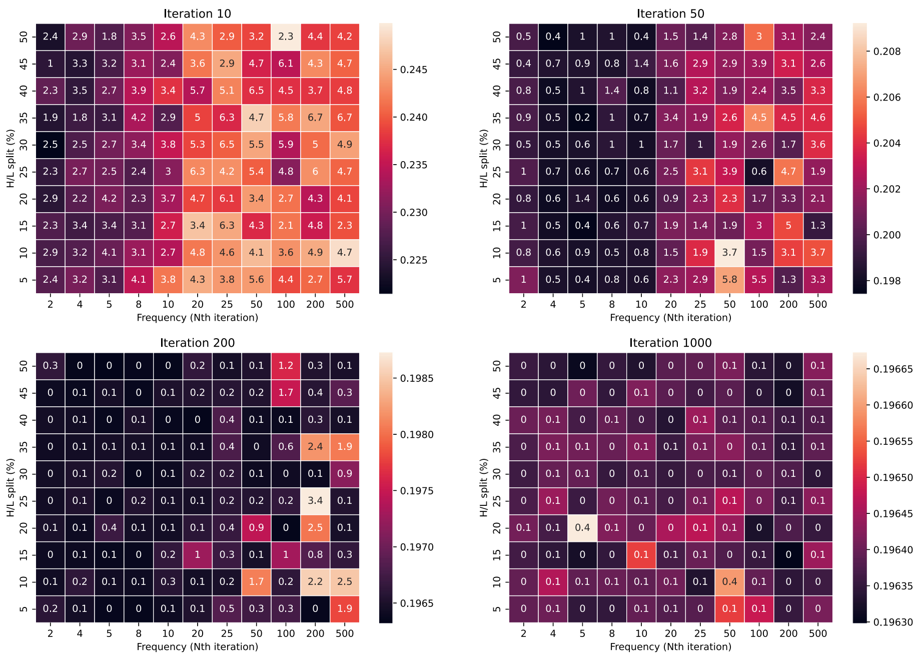


Fig. 13. The mean and standard deviations of the best fitness values for iterations 10, 50, 200, and 1,000 for the car-side impact problem.

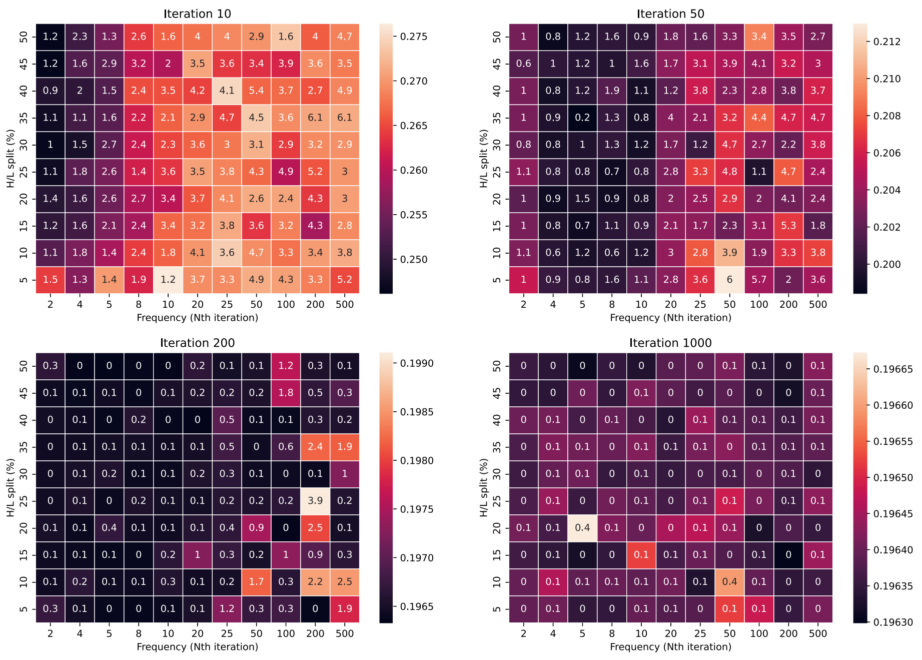


Fig. 14. The mean and standard deviations of the average of the fitness values for iterations 10, 50, 200, and 1,000 for the car-side impact problem.

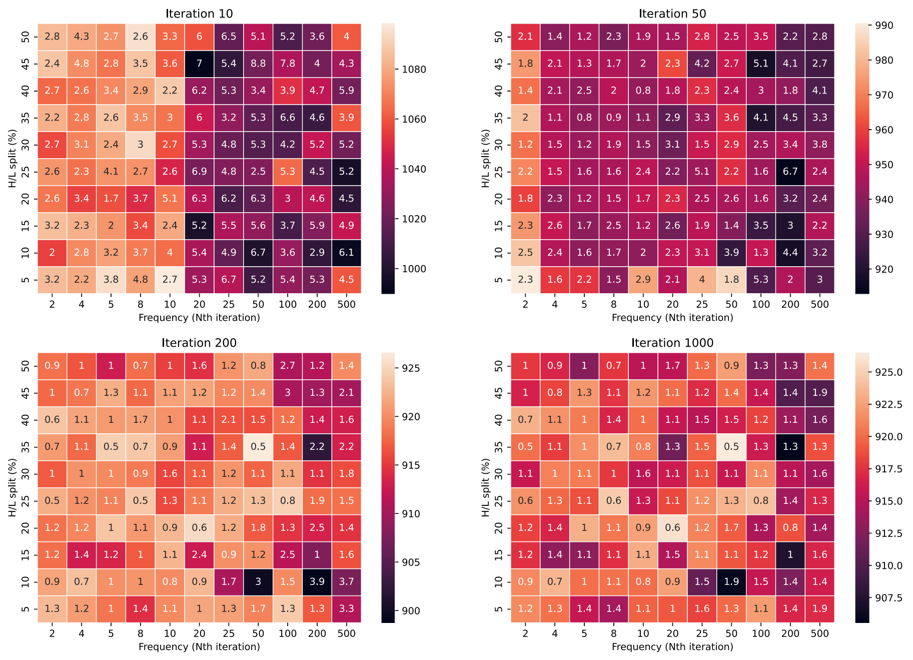


Fig. 15. The mean and standard deviations of the hypervolumes values of the population in iterations 10, 50, 200, and 1,000 for the car-side impact problem.

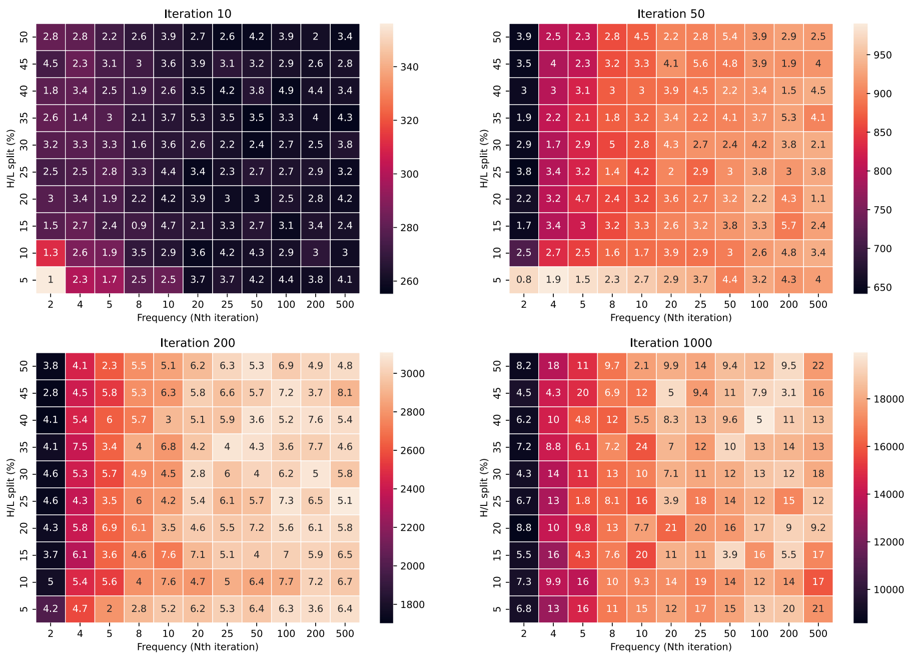


Fig. 16. The mean and standard deviations of the cumulative sums of unique solutions in iterations 10, 50, 200, and 1,000 for the car-side impact problem.

ACKNOWLEDGMENTS

I would like to warmly thank Kaisa Miettinen, Bekir Afsar, Francisco Ruiz, and Babooshka Shavazipour for their valuable comments on the manuscript. This 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.

REFERENCES

Bekir Afsar, Kaisa Miettinen, and Francisco Ruiz. 2021. Assessing the performance of interactive multiobjective optimization methods: A survey. *ACM Computing Surveys* 54, 4 (2021), 1–27.

Bekir Afsar, Johanna Silvennoinen, and Kaisa Miettinen. 2023. A systematic way of structuring real-world multiobjective optimization problems. In *Evolutionary Multi-Criterion Optimization*. Lecture Notes in Computer Science, Vol. 13970. Springer, 593–605.

Bekir Afsar, Johanna Silvennoinen, Giovanni Misitano, Francisco Ruiz, Ana B. Ruiz, and Kaisa Miettinen. 2022. Designing empirical experiments to compare interactive multiobjective optimization methods. *Journal of the Operational Research Society* 2022 (2022), 1–12.

Luis Miguel Antonio and Carlos A. Coello Coello. 2017. Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation* 22, 6 (2017), 851–865.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115.

Sunith Bandaru, Amos H. C. Ng, and Kalyanmoy Deb. 2017a. Data mining methods for knowledge discovery in multi-objective optimization: Part A—Survey. *Expert Systems with Applications* 70 (2017), 139–159.

Sunith Bandaru, Amos H. C. Ng, and Kalyanmoy Deb. 2017b. Data mining methods for knowledge discovery in multi-objective optimization: Part B—New developments and applications. *Expert Systems with Applications* 70 (2017), 119–138.

Valerie Belton, Jürgen Branke, Petri Eskelinen, Salvatore Greco, Julián Molina, Francisco Ruiz, and Roman Słowiński. 2008. Interactive multiobjective optimization from a learning perspective. In *Multiobjective Optimization*. Springer, 405–433.

R. Benayoun, J. de Montgolfier, J. Tergny, and O. Laritchev. 1971. Linear programming with multiple objective functions: Step method (stem). *Mathematical Programming* 1, 1 (1971), 366–375.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer, New York, NY.

Guido Bologna. 2021. A rule extraction technique applied to ensembles of neural networks, random forests, and gradient-boosted trees. *Algorithms* 14, 12 (2021), 339.

Jurgen Branke, Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowiński. 2008. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Vol. 5252. Springer Science & Business Media.

Leo Breiman. 2017. *Classification and Regression Trees*. Routledge.

Tinkle Chugh, Karthik Sindhya, Jussi Hakanen, and Kaisa Miettinen. 2019. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing* 23, 9 (2019), 3137–3166.

Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. 2007. *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer.

William W. Cohen and Yoram Singer. 1999. A simple, fast, and effective rule learner. In *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI '99)*. 335–342.

Salvatore Corrente, Salvatore Greco, Benedetto Matarazzo, and Roman Slowinski. 2021. Explainable interactive evolutionary multiobjective optimization. *SSRN*. Retrieved October 23, 2023 from https://ssrn.com/abstract=3792994

Kalyanmoy Deb and Ram Bhushan Agrawal. 1995. Simulated binary crossover for continuous search space. *Complex Systems* 9, 2 (1995), 115–148.

Kalyanmoy Deb and Kaisa Miettinen. 2009. A review of nadir point estimation procedures using evolutionary approaches: A tale of dimensionality reduction. In *Proceedings of the Multiple Criterion Decision Making Conference (MCDM '08)*. 1–14.

Kalyanmoy Deb, Kaisa Miettinen, and Shamik Chaudhuri. 2010. 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.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and Tamt Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

Kalyanmoy Deb and Aravind Srinivasan. 2006. Innovization: Innovating design principles through optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. 1629–1636.

Michael T. M. Emmerich and André H. Deutz. 2018. A tutorial on multiobjective optimization: Fundamentals and evolutionary methods. *Natural Computing* 17, 3 (2018), 585–609.

Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. 2017. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542, 7639 (2017), 115–118.

Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55, 1 (1997), 119–139.

Nicolas Goix, Vighnesh Birodkar, Florian Gardin, Jean-Matthieu Schertzer, Hoebin Jeong, Manoj Kumar, Alexandre Gramfort, Tim Staley, Tom Dupré la Tour, Boyuan Deng, C, Fabian Pedregosa, Lawrence Wu, Ariel Rokem, Kyle Jackson, and mrahim. 2020. scikit-learn-contrib/skope-rules v1.0.1. Retrieved October 23, 2023 from https://doi.org/10.5281/zenodo.4316671

Bryce Goodman and Seth Flaxman. 2017. European union regulations on algorithmic decision-making and a "right to explanation." *AI Magazine* 38, 3 (2017), 50–57.

David Gunning and David Aha. 2019. DARPA's explainable artificial intelligence (XAI) program. *AI Magazine* 40, 2 (2019), 44–58.

Jussi Hakanen, Kaisa Miettinen, and Kristian Sahlstedt. 2011. Wastewater treatment: New insight provided by interactive multiobjective optimization. *Decision Support Systems* 51, 2 (2011), 328–337.

Jussi Hakanen, Kristian Sahlstedt, and Kaisa Miettinen. 2013. Wastewater treatment plant design and operation under multiple conflicting objective functions. *Environmental Modelling & Software* 46 (2013), 240–249.

Ching-Lai Hwang and Abu Syed Md. Masud. 1979. *Multiple Objective Decision Making: Methods and Applications*. Springer, Berlin, Germany.

Himanshu Jain and Kalyanmoy Deb. 2013. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation* 18, 4 (2013), 602–622.

Laetitia Jourdan, David Corne, Dragan Savic, and Godfrey Walters. 2005. Preliminary investigation of the 'learnable evolution model' for faster/better multiobjective water systems design. In Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science, Vol. 3410. Springer, 841–855.

Uday Kamath and John Liu. 2021. *Explainable Artificial Intelligence: An Introduction to Interpretable Machine Learning*. Springer.

Adhe Kania, Juha Sipilä, Giovanni Misitano, Kaisa Miettinen, and Jussi Lehtimäki. 2022. Integration of lot sizing and safety strategy placement using interactive multiobjective optimization. *Computers & Industrial Engineering* 173 (2022), 108731.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. 2016. Jupyter notebooks—A publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt (Eds.). IOS Press, 87–90.

Xingtao Liao, Qing Li, Xujing Yang, Weigang Zhang, and Wei Li. 2008. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization* 35, 6 (2008), 561–569.

Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2020. Explainable AI: A review of machine learning interpretability methods. *Entropy* 23, 1 (2020), 18.

Zachary C. Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue* 16, 3 (2018), 31–57.

Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. 2018. Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing* 108 (2018), 33–47.

Jie Lu, Guangquan Zhang, Da Ruan, and Fengjie Wu. 2007. *Multi-Objective Group Decision Making*. Imperial College Press.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*. 4768–4777.

Michael D. McKay, Richard J. Beckman, and William J. Conover. 2000. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 42, 1 (2000), 55–61.

Ryszard S. Michalski. 2000. Learnable evolution model: Evolutionary processes guided by machine learning. *Machine Learning* 38, 1 (2000), 9–40.

Kaisa Miettinen. 1999. *Nonlinear Multiobjective Optimization*. Kluwer Academic, Boston, MA.

Kaisa Miettinen, Jussi Hakanen, and Dmitry Podkopaev. 2016. Interactive nonlinear multiobjective optimization methods. In *Multiple Criteria Decision Analysis*. Springer, 927–976.

Kaisa Miettinen and Marko M. Mäkelä. 1999. Comparative evaluation of some interactive reference point-based methods for multi-objective optimisation. *Journal of the Operational Research Society* 50, 9 (1999), 949–959.

Kaisa Miettinen and Marko M. Mäkelä. 2002. On scalarizing functions in multiobjective optimization. *OR Spectrum* 24, 2 (2002), 193–213.

Kaisa Miettinen and Francisco Ruiz. 2016. NAUTILUS framework: Towards trade-off-free interaction in multiobjective optimization. *Journal of Business Economics* 86 (2016), 5–21.

Kaisa Miettinen, Francisco Ruiz, and Andrzej P. Wierzbicki. 2008. Introduction to multiobjective optimization: Interactive approaches. In *Multiobjective Optimization*. Springer, 27–57.

Giovanni Misitano. 2020. Interactively learning the preferences of a decision maker in multi-objective optimization utilizing belief-rules. In *Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI '20)*. IEEE, Los Alamitos, CA, 133–140.

Giovanni Misitano. 2023a. gialmisi/XLEMOO: Article Code. Retrieved October 23, 2023 from https://doi.org/10.5281/zenodo.8318957

Giovanni Misitano. 2023b. XLEMOO Numerical Experiment Data. Retrieved October 23, 2023 from https://doi.org/10.5281/zenodo.8085638

Giovanni Misitano, Bekir Afsar, Giomara Lárraga, and Kaisa Miettinen. 2022. Towards explainable interactive multiobjective optimization: R-XIMO. *Autonomous Agents and Multi-Agent Systems* 36, 2 (2022), 1–43.

Giovanni Misitano, Bhupinder Singh Saini, Bekir Afsar, Babooshka Shavazipour, and Kaisa Miettinen. 2021. DESDEO: The modular and open source framework for interactive multiobjective optimization. *IEEE Access* 9 (2021), 148277–148295.

Felix Mölder, Kim Philipp Jablonski, Brice Letcher, Michael B. Hall, Christopher H. Tomkins-Tinch, Vanessa Sochat, Jan Forster, Soohyun Lee, Sven O. Twardziok, Alexander Kanitz, et al. 2021. Sustainable data analysis with Snakemake. *F1000Research* 10 (2021), 33.

Christoph Molnar. 2022. *Interpretable Machine Learning* (2nd ed.). Christoph Molnar. https://christophm.github.io/interpretable-ml-book

Behzad Moradi. 2018. Multi-objective mobile robot path planning problem through learnable evolution model. *Journal of Experimental & Theoretical Artificial Intelligence* 31, 2 (Nov. 2018), 325–348.

Behzad Moradi. 2019. The new optimization algorithm for the vehicle routing problem with time windows using multi-objective discrete learnable evolution model. *Soft Computing* 24, 9 (2019), 6741–6769.

Behzad Moradi and Abdolreza Mirzaei. 2016. A new automated design method based on machine learning for CMOS analog circuits. *International Journal of Electronics* 103, 11 (2016), 1868–1881.

Deepak Nagar, Palaniappan Ramu, and Kalyanmoy Deb. 2022. Visualization and analysis of Pareto-optimal fronts using interpretable self-organizing map (iSOM). *Swarm and Evolutionary Computation* 76 (2022), 101202.

Sara Narteni, Melissa Ferretti, Vanessa Orani, Ivan Vaccari, Enrico Cambiaso, and Maurizio Mongelli. 2021. From explainable to reliable artificial intelligence. In *Proceedings of the International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 255–273.

Yunyun Niu, Detian Kong, Rong Wen, Zhiguang Cao, and Jianhua Xiao. 2021. An improved learnable evolution model for solving multi-objective vehicle routing problem with stochastic demand. *Knowledge-Based Systems* 230 (2021), 107378.

Tapabrata Ray, Hemant Kumar Singh, Kamrul Hasan Rahi, Tobias Rodemann, and Markus Olhofer. 2022. Towards identification of solutions of interest for multi-objective problems considering both objective and variable space information. *Applied Soft Computing* 119 (2022), 108505.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1135–1144.

Steven L. Salzberg. 1994. *C4. 5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993.* Kluwer Academic.

Yoshikazu Sawaragi, Hirotaka Nakayama, and Tetsuzo Tanino. 1985. *Theory of Multiobjective Optimization*. Elsevier.

Chandan Singh, Keyan Nasseri, Yan Shuo Tan, Tiffany Tang, and Bin Yu. 2021. imodels: A Python Package for Fitting Interpretable Models. Retrieved October 23, 2023 from https://doi.org/10.21105/joss.03192

Henrik Smedberg and Sunith Bandaru. 2022. Interactive knowledge discovery and knowledge visualization for decision support in multi-objective optimization. *European Journal of Operational Research* 306, 3 (2022), 1311–1329.

Ralph Steuer. 1989. *Multiple Criteria Optimization: Theory, Computation, and Application*. Krieger Publishing Company.

Roykrong Sukkerd, Reid Simmons, and David Garlan. 2018. Toward explainable multi-objective probabilistic planning. In *Proceedings of the 2018 IEEE/ACM 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS '18)*. IEEE, Los Alamitos, CA, 19–25.

El-Ghazali Talbi, Matthieu Basseur, Antonio J. Nebro, and Enrique Alba. 2012. Multi-objective optimization using metaheuristics: Non-standard algorithms. *International Transactions in Operational Research* 19, 1-2 (2012), 283–305.

Ryoji Tanabe and Hisao Ishibuchi. 2020. An easy-to-use real-world multi-objective optimization problem suite. *Applied Soft Computing* 89 (2020), 106078.

Jasper van der Waa, Elisabeth Nieuwburg, Anita Cremers, and Mark Neerincx. 2021. Evaluating XAI: A comparison of rule-based and example-based explanations. *Artificial Intelligence* 291 (2021), 103404.

Jinkun Wang, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, and Chunhua Sun. 2016. Diversified recommendation incorporating item content information based on MOEA/D. In *Proceedings of the 2016 49th Hawaii International Conference on System Sciences (HICSS '16)*. IEEE, Los Alamitos, CA, 688–696.

Andrzej P. Wierzbicki. 1980. The use of reference objectives in multiobjective optimization. In *Multiple Criteria Decision Making Theory and Application*. Springer, 468–486.

Andrzej P. Wierzbicki. 1982. A mathematical basis for satisficing decision making. *Mathematical Modelling* 3, 5 (1982), 391–405.

Bin Xin, Lu Chen, Jie Chen, Hisao Ishibuchi, Kaoru Hirota, and Bo Liu. 2018. Interactive multiobjective optimization: A review of the state-of-the-art. *IEEE Access* 6 (2018), 41256–41279.

Huixin Zhan and Yongcan Cao. 2019. Relationship explainable multi-objective optimization via vector value function based reinforcement learning. *arXiv preprint arXiv:1910.01919* (2019).

Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (2007), 712–731.

Eckart Zitzler and Simon Künzli. 2004. Indicator-based selection in multiobjective search. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*. 832–842.

Received 15 November 2022; revised 16 September 2023; accepted 28 August 2023

IV

DESDEO: THE MODULAR AND OPEN SOURCE FRAMEWORK FOR INTERACTIVE MULTIOBJECTIVE OPTIMIZATION

by

Misitano G., Saini B. S., Afsar B., Shavazipour B., Miettinen K. 2021

IEEE Access, 9, 148277–148295

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]:

$$\begin{aligned} \min \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x})) \\ \text{s.t.} \quad & \mathbf{x} \in S, \end{aligned} \tag{1}$$

where $f_i \colon S \rightarrow \mathbb{R}$ $(i = 1, \dots, k)$ are objective functions and $\mathbf{x} = (x_1, \dots, x_n)^T$ 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.

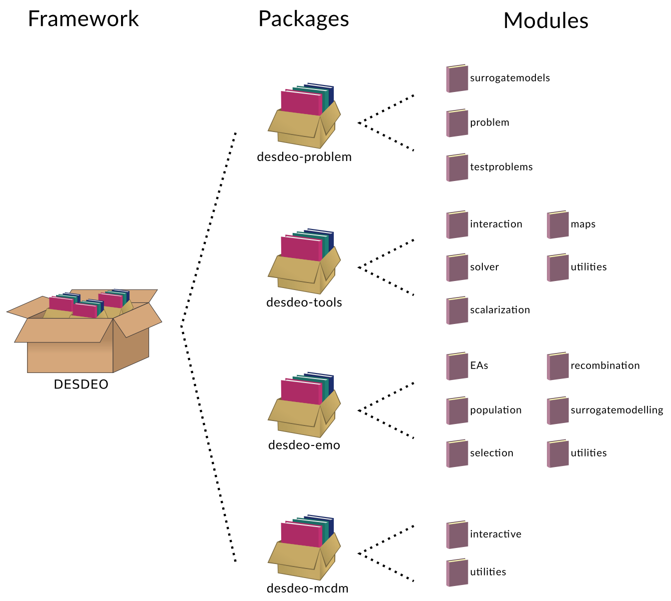


FIGURE 1. The main structure of the DESDEO framework with packages and modules included in each package. Further packages and modules can be added as needed.

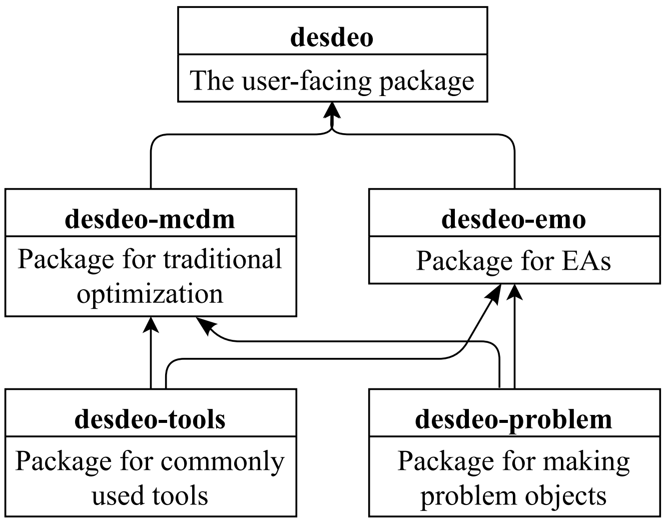

desdeo
The user-facing package
desdeo-mcdm
Package for traditional optimization
desdeo-emo
Package for EAs
desdeo-tools
Package for commonly used tools
desdeo-problem
Package for making problem objects

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:

$$\begin{aligned}
\min \quad & f_1(\mathbf{x}) = -4.07 - 2.27x_1 \\
\min \quad & f_2(\mathbf{x}) = -2.60 - 0.03x_1 - 0.02x_2 \\
& \qquad - \frac{0.01}{1.39 - x_1^2} - \frac{0.30}{1.39 - x_2^2} \\
\min \quad & f_3(\mathbf{x}) = -8.21 + \frac{0.71}{1.09 - x_1^2} \\
\min \quad & f_4(\mathbf{x}) = -0.96 + \frac{0.96}{1.09 - x_2^2} \\
\min \quad & f_5(\mathbf{x}) = \max\{|x_1 - 0.65|, |x_2 - 0.65|\} \\
\text{s.t.} \quad & 0.3 \le x_1, x_2 \le 1.0,
\end{aligned} \tag{2}$$

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.

```python
import numpy as np

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.

```python
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.

```python
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

```python
mo_problem = MOProblem(variables=variables, objectives=objectives)
```

SOURCE CODE 4. Defining the multiobjective optimization problem object of problem (2).

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.

```python
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.

```python
1 from desdeo_mcdm.interactive.NIMBUS import NIMBUS
2
3 nimbus = NIMBUS(mo_problem)
4
5 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
...	...	...	...	...	...	...

```
1  import pandas as pd
2  from desdeo_problem.problem import DataProblem
3
4
5  training_data =
   ↪   pd.read_csv("./data/River_pollution.csv",
   ↪   comment="#")
6
7  problem = DataProblem(
8      data=training_data,
9      variable_names=["x_1", "x_2"],
10     objective_names=["f_1", "f_2", "f_3", "f_4",
   ↪       "f_5"],
11     bounds=pd.DataFrame(
12         [[0.3, 0.3], [1.0, 1.0]],
13         columns=["x_1", "x_2"],
14         index=["lower_bound", "upper_bound"]),
15 )
```

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*.

```
1  from desdeo_emo.EAs import RVEA
2
3  evolver = RVEA(
4      problem,
5      interact=True,
6      use_surrogates=True
7  )
8
9  (_, _, refp_request, _), _ = evolver.requests()
10
11 # references given to refp_request
12
13 (_, _, 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.

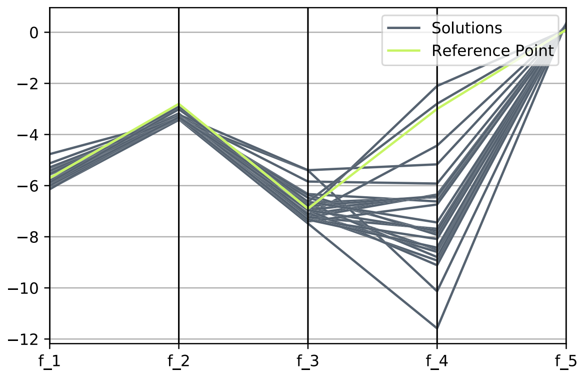


FIGURE 3. Solutions obtained for the data-driven river pollution problem after using RVEA for one iteration.

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.

```python
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.

```python
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.

```python
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.

```python
from desdeo_mcdm.interactive import
  ↪  ReferencePointMethod

rp_method = ReferencePointMethod(mo_problem,
  ↪  mo_problem.ideal, mo_problem.nadir)
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.

```python
response = {
    "reference_point": np.array([-6.06739, -2.79173,
    ↪  -5.96145, -6.57333, 0.30863]),
}

initial_request.response = response

rp_request = rp_method.iterate(initial_request)

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.

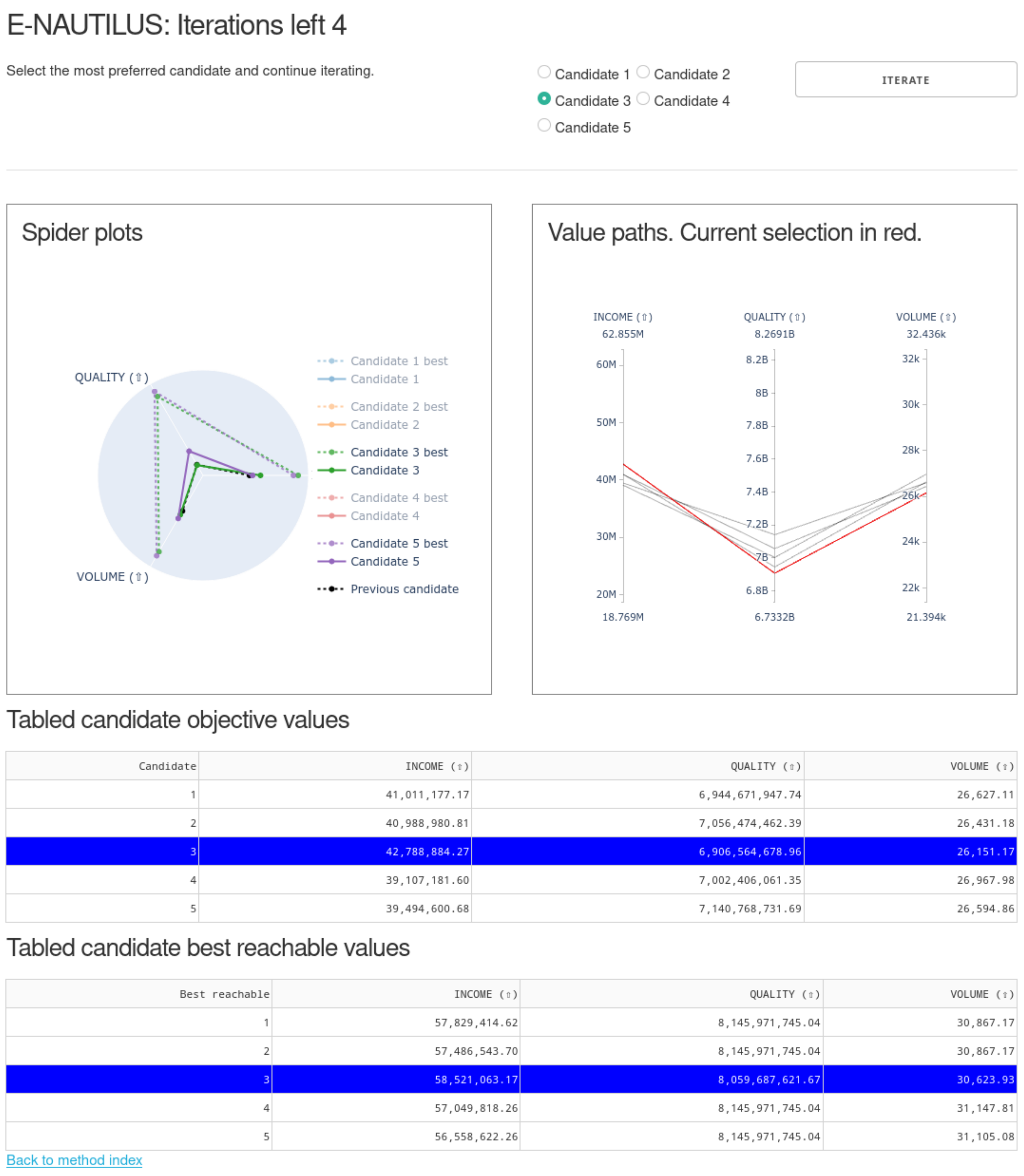
F. SOFTWARE APPLICATIONS BUILT UTILIZING DESDEO

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



E-NAUTILUS: Iterations left 4

Select the most preferred candidate and continue iterating.

( ) Candidate 1 ( ) Candidate 2 [✓ Candidate 3] ( ) Candidate 4 ( ) Candidate 5

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 multi-agent 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.

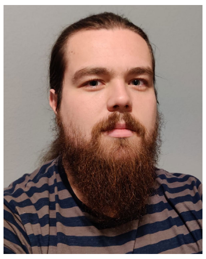
[56] B. Afsar, K. Miettinen, and F. Ruiz, ''Assessing the performance of interactive multiobjective optimization methods: A survey,'' *ACM Comput. Surveys*, vol. 54, no. 4, p. 85, 2021.

[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.

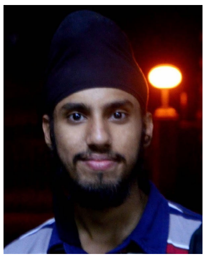
[58] S. Huber, M. J. Geiger, and M. Sevaux, ''Simulation of preference information in an interactive reference point-based method for the bi-objective inventory routing problem,'' *J. Multi-Criteria Decis. Anal.*, vol. 22, nos. 1–2, pp. 17–35, Jan. 2015.

[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.

• • •