Samuli Kurikka

Extending the DESDEO Software Framework: Development and Evaluation of a User Interface for the NAUTILUS Method

Master's Thesis in Information Technology

June 11, 2024

University of Jyväskylä

Faculty of Information Technology

Author: Samuli Kurikka

Contact information: sajojoku@student.jyu.fi

Supervisor: Kaisa Miettinen, and Bhupinder Saini

Title: Extending the DESDEO Software Framework: Development and Evaluation of a User Interface for the NAUTILUS Method

Työn nimi: DESDEO-ohjelmistokehyksen laajentaminen: NAUTILUS-menetelmän käyttöliittymän kehittäminen ja arviointi

Project: Master's Thesis

Study line: Tietotekniikka

Page count: 65+1

Abstract: This thesis provides a holistic overview of the development and implementation of a user interface for the NAUTILUS method within the DESDEO software framework. Multiobjective optimisation involves solving problems with multiple conflicting objectives, resulting in various Pareto optimal solutions involving different trade-offs. NAUTILUS is an interactive multiobjective optimisation method that assists a decision maker in finding one's most preferred Pareto optimal solution without requiring to make any trade-offs. The interactive solution process using the NAUTILUS method typically starts from an inferior starting point that allows for possible improvements across all objective functions. The decision maker iteratively approaches the set of Pareto optimal solutions, using ranks or weights as forms of providing preference information. This research employs a design science methodology, culminating in the creation and evaluation of an artefact—the implementation of the user interface for the NAUTILUS method. The user interface will adhere to the design principles of the DESDEO framework, such as modular structuring. This approach ensures seamless integration within the DESDEO software framework, allowing the components presented in this thesis to be utilised in DESDEO. Additionally, it enables the reusable components from DESDEO to be incorporated into the NAUTILUS user interface.

Keywords: NAUTILUS, multiobjective optimisation, decision maker, user interface, DES-

DEO

Suomenkielinen tiivistelmä: Tässä tutkielmassa annetaan kokonaisvaltainen kuvaus NAUTILUS-menetelmän käyttöliittymän kehittämisestä ja toteutuksesta DESDEO-ohjelmistokehykseen. Monitavoiteoptimointi etsii ratkaisua ongelmiin, joissa on yleensä ristiriitaisia tavoitteita. Ratkaisun löytämiseksi päätöksentekijä on joutunut tekemään kompromisseja. NAUTILUS on interaktiivinen monitavoiteoptimoinnin menetelmä, joka auttaa päätöksentekijää löytämään hänelle parhaan Pareto-optimaalisen ratkaisun ilman, että hänen tarvitsee tehdä kompromisseja. Interaktiivinen ratkaisuprosessi NAUTILUS-menetelmällä alkaa tyypillisesti lähtökohdasta, joka mahdollistaa parannukset kaikissa tavoitefunktioissa. NAUTILUS-menetelmän avulla päätöksentekijä lähestyy iteratiivisesti Pareto-optimaalisten ratkaisujen joukkoa, ja menetelmä varmistaa tavoitefunktioiden arvojen parantumisen jokaisella iteraatiokierroksella. Tutkimus noudattaa suunnittelutieteellistä metodologiaa, ja se huipentuu artefaktin luomiseen, eli käyttöliittymän toteutukseen NAUTILUS-menetelmälle. Käyttöliittymä noudattaa DESDEO-ohjelmistokehyksen suunnitteluperiaatteita, kuten modulaarista rakennetta. Tämä lähestymistapa takaa saumattoman integraation, mahdollistaen tässä tutkielmassa esiteltyjen komponenttien käytön DESDEO:ssa. Lisäksi se mahdollistaa DESDEO:n uudelleenkäytettävien komponenttien sisällyttämisen NAUTILUS-menetelmän käyttöliittymään.

Avainsanat: NAUTILUS, monitavoiteoptimointi, päätöksentekijä, käyttöliittymä, DESDEO

List of Figures

Contents

1 Introduction

According to Miettinen (1999), multiobjective optimisation addresses optimisation problems that involve multiple conflicting objective functions that must be optimised at the same time. Because of these conflicts, multiobjective optimisation problems usually do not have a single optimal solution, but instead, a set of mathematically equivalent solutions called Pareto optimal solutions. Consequently, solving a multiobjective optimisation problem often necessitates the involvement of a decision maker to determine the most preferred solution from Pareto optimal ones, as multiobjective optimisation is fundamentally centred around assisting a decision maker in finding the most preferred solution that satisfies their preferences. To identify the most preferred solution, the decision maker must carefully analyse the problem and progressively deepen their understanding of both the problem and its potential solutions.

In order for the decision maker to find Pareto optimal solutions, they can use various methods. As outlined by Miettinen (1999), one way to classify multiobjective optimisation methods is based on when a decision maker and preference information take part in the solution process. The decision maker can use methods that requires preference information after a representative set of solutions has been identified. Alternatively, they can use methods that ask preference information before the optimisation process begins. The decision maker can also be involved in the process throughout, providing preference information interactively. Methods where the user provides preference information throughout the process are called interactive methods.

According to Miettinen, Ruiz, and Wierzbicki (2008), interactive methods, are based on a continuous engagement between the decision maker and the optimisation process. The process unfolds through an iterative solution pattern, where each iteration brings the decision maker closer towards identifying their most preferred solution. The decision maker does not need to have any global knowledge of what Pareto optimal solutions can be achieved and can learn from the solution process, adjusting the preference information accordingly. To unleash the full potential of interactive methods, it is essential for the decision maker to have something to interact with during the process. This necessity highlights the importance of developing effective user interfaces for such methods.

The essence of this thesis is the implementation and evaluation of an interactive multiobjective optimisation method. The method in question, introduced by Miettinen et al. (2010), is called NAUTILUS. NAUTILUS is part of a family of NAUTILUS methods introduced by Miettinen and Ruiz (2016), and here we concentrate on the development of the very first in the series. Members of the NAUTILUS family incorporate an approach that rethinks traditional interactive multiobjective problem solving where the search for a Pareto optimal solution begins from an inferior starting point, and improvements in each objective function are possible.

The user interface for the NAUTILUS method will be implemented into an open-source software framework called DESDEO (Misitano et al. 2021). Among other multiobjective optimisation methods, some members of the NAUTILUS family already have a user interface implementation in DESDEO. This thesis extends that work by focusing on the implementation and evaluation of the first NAUTILUS method within this framework. The research centres on the question of how to successfully implement the user interface of the NAUTILUS method in the context of DESDEO.

In Chapter 2, we explore the background concepts relevant to the thesis. We introduce concepts of multiobjective optimisation, such as Pareto optimality and interactive methods. We also introduce the DESDEO software framework and explain the research approach of this work in detail. Some concepts of web development are also introduced, as the user interface is implemented using web technologies. Chapter 3 focuses on the NAUTILUS method in detail. We provide an overview of the method, detailing its operational framework and its approach to solving multiobjective optimisation problems. To aid in clarity, we provide a comprehensive walk-through of the NAUTILUS algorithm.

In Chapter 4, we explore the detailed implementation steps necessary for developing the user interface for the NAUTILUS method. This chapter begins by identifying key aspects for addressing the research question. We discuss the development challenges, including the information exchange process between the user and the user interface, and how this exchange is constructed and handled in the background. The chapter concludes with a description of the user interface's structure, including how users interact with the user interface through iterations, how they provide preference information, and how the results of solving the opti-

misation problem are presented.

In Chapter 5, we evaluate the user interface developed in this thesis. The evaluation begins with a case study introducing an example multiobjective optimisation problem, where we use NAUTILUS to find a preferred solution to that problem. Subsequently, we conduct a detailed assessment of the user interface, evaluating how effectively the developed ways of providing preference information supports the decision maker in the decision-making process. We evaluate the quality of information exchange and the interface's accessibility to a broad range of users, considering their experience and expertise levels. Additionally, we explore how well the user interface integrates within the DESDEO software framework.

In Chapter 6, we summarise what has been accomplished and the results of this thesis. We review the key highlights of this research, as well as the areas needing improvement. We also discuss potential future research related to this thesis. In Chapter 7, we conclude the thesis.

2 Background

This chapter introduces the main concepts of multiobjective optimisation relevant to this thesis, and some mathematical foundations that underpin multiobjective optimisation problems. The concepts of interactive, *a priori* and *a posteriori* methods are introduced. The DESDEO software framework is examined, where the user interface of the NAUTILUS method is implemented. Next, the concepts related to user interface design and implementation, emphasising on user interfaces in the context of multiobjective optimisation are discussed. Lastly, guidelines for design science research are examined, relating to the aspects needed in a design science research project, and how to evaluate the user interface that is created.

2.1 Multiobjective optimisation

According to Miettinen (1999), multiobjective optimisation refers to the systematic process of optimising more than one objective function, which usually are in conflict. One of the main characteristics of multiobjective optimisation is the absence of a single unique solution. Instead, a collection of mathematically incomparable solutions, called Pareto optimal solutions, is encountered. In this section, the main concepts of multiobjective optimisation, such as Pareto optimality and preference information are presented, and how a multiobjective optimisation problem is formulated. This section also includes a brief touch on different kinds of multiobjective optimisation methods.

To begin our discussion on multiobjective optimisation, let us address a question of what is multiobjective optimisation. To illustrate this, let us consider the following case: a head architect, who works at an architecture firm, is tasked with designing and planning the construction of an office building for a client. The architect could face three objectives:

1. Construction cost: the architect aims to minimise the upfront cost of constructing the building, which is crucial for staying within the budget.
2. Energy consumption: the architect aims to minimise the building's energy use through design strategies and technologies, promoting long-term sustainability.
3. Project profitability: as the firm wants to make profit, the architect aims to maximise

the profit by making a competitive offer that would be accepted by the client, but also ensures the greatest possible financial return for the firm.

These objectives are naturally in conflict, as measures of choosing energy-efficient materials may increase initial costs, while reducing long-term energy consumption. On the other hand, reducing construction costs might lead to a less sustainable building, which could also affect the profitability of the project, in cases where the market for office buildings favours environmentally friendly designs. Thus, a decision made in favour of one objective might lead to deterioration in at least one of the other objectives. These are called trade-offs, that the architect must carefully consider to reach an optimal solution that in this case balances all three objectives: construction cost, energy consumption, and profitability.

The architect must make decisions based on the available choices and information, placing him/her in the role of a decision maker. A decision maker is an individual who has the insight of the background of the problem, and is the one who provides preference information. Preferences can be understood as specific inputs that are used to solve multiobjective optimisation problems. Preferences can take various forms. The architect can prioritise the construction cost, aiming to find designs that primarily reduce upfront expenses. The architect may also prioritise both construction cost and project profitability equally, where these objectives together will have a higher effect to the outcome compared solely to energy consumption. This imaginary case above is an extremely simplified example of an optimisation problem, as real-life problems are typically much more complex. Nonetheless, this serves as a useful representation of what a real-life multiobjective optimisation problem might entail.

Next, we will formally define a multiobjective optimisation problem. Following the description provided by Miettinen (1999), a general multiobjective optimisation problem is described as follows:

$$\begin{aligned} \min \quad & \{f_1(x), \ldots, f_k(x)\} \\ \text{subject to} \quad & x \in S \subset \mathbb{R}^n, \end{aligned} \tag{2.1}$$

where the decision maker aims to concurrently minimise $k (k \geq 2)$ *objective functions* $f_i : S \to \mathbb{R}$. The vector of objective function values, called an objective vector is denoted by

$f(x) = (f_1(x), \ldots, f_k(x))^T$. The decision variable vectors, also known as *decision vectors* are denoted by $x = (x_1, \ldots, x_n)^T$, where n is the number of decision variables. Decision vectors are part of a *feasible set* $S \subset \mathbb{R}^n$, that is formed by constraints. Constraints can be equality or inequality constraints, and they are defined as part of the problem. Finally, we define a feasible objective set $Z = f(S)$ in the objective space $\mathbb{R}^k$ and it is denoted by the image of the feasible set S.

For simplicity, we assume that all objectives functions are to be minimised. If there is a need for an objective function to be maximised, we negate that objective function. As Miettinen (1999) outlines, and as mentioned above, multiobjective optimisation problems do not have a single optimum. Instead, a set of optimal solutions exists, called a Pareto optimal set. A solution is deemed Pareto optimal if improving the value of any objective function is not possible without a simultaneous decline in the value of one or more other objective function values. In other words, a decision vector x^* that is in the feasible set S, is Pareto optimal if there is no other $x \in S$, such that $f_i(x) \leq f_i(x^*)$ for all $i = 1, \ldots, k$, with the inequality being strict for at least one index. Similarly, an objective vector z^* that is in the feasible objective set Z, is Pareto optimal if there is no other $z \in Z$, such that $z_i \leq z_i^*$ for all i, with the inequality being strict for at least one index.

When solving a multiobjective optimisation problem, knowledge of the ranges for each objective function in the Pareto optimal set is often useful to the decision maker. We can calculate best values, that is, the lower bound z_i^* for each individual objective function by minimising each f_i individually subject to the constraints. These values, when combined into a vector $z^* = (z_1^*, \ldots, z_k^*)$ constitute an *ideal objective vector*, referred from now on as an ideal point. We can also define another vector, z^{nad} that is called a *nadir objective vector*, referred from now on as a nadir point. The nadir point represents upper bounds, that is, the worst values of objective functions in the Pareto optimal set. As noted by Miettinen, Ruiz, and Wierzbicki (2008), there is no exact way to calculate the nadir point. We can use a so-called payoff table to calculate estimations of the nadir point. A payoff table is a matrix that arranges the values of multiple objective functions evaluated at the optimal solutions found for each objective when optimised independently. The diagonal of the payoff table represents the ideal point, and the worst values in each column provide an approximation of the nadir point.

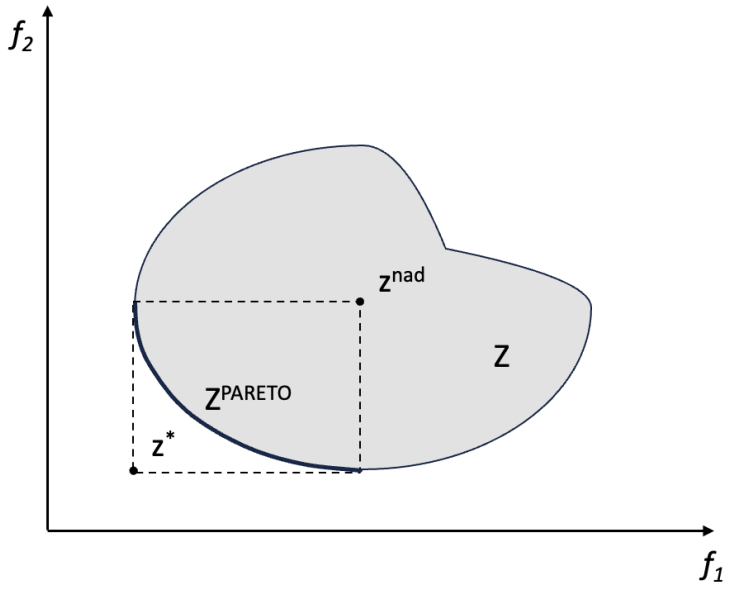

Figure 1. Graphical representation of relevant multiobjective optimisation concepts

However, the accuracy of the resulting point may not always be precise because generally the full extent of the Pareto optimal set is not known.

The concepts of a Pareto optimal set, a feasible objective set, z^{nad} and z^* points are all visualised in Figure 1 using a problem with two objectives, also known as a bi-objective optimisation problem. The filled grey area signifies the feasible objective set. Z^{PARETO} — indicated by the thick line — represents the Pareto optimal set in the objective space.

As Miettinen (1999) outlines, the problem can be considered solved when a feasible decision vector is found that is Pareto optimal and it satisfies the decision maker's requirements. Trying to identify all Pareto optimal solutions is not feasible since there may even be an infinite number of them. For practical purposes, there is a necessity to narrow down this set to one implementable solution. Therefore, methods are needed that can take into account the preferences of the decision maker, because without this information, Pareto optimal solutions cannot be ordered. The decision maker's role in multiobjective optimisation is to determine which solution is the most preferred one and to provide preference information to be used in the solution process. As outlined by Miettinen (1999), preference information can be integrated into the solution process at various stages: before, during, or after the optimisation. The classification of multiobjective optimisation methods presented here is based on the

timing of providing preferences.

2.1.1 Categories of multiobjective optimisation methods

Hwang and Masud (1979) categorised the multiobjective optimisation methods into four distinct classes, based on the decision maker's involvement in the solution process:

1. No-preference methods: The decision maker does not give any preference information.
2. *A posteriori* methods: The decision maker gives preferences after a representative set of solutions has been are generated.
3. *A priori* methods: The decision maker gives preferences before the optimisation process begins.
4. Interactive methods: The decision maker progressively gives preferences throughout the solution process.

In no-preference methods, the optimal solution is selected without the involvement of the decision maker. Here the decision maker's role is limited to accepting or rejecting the outcome. If preferences are given after computing a representative set of Pareto optimal solutions, the corresponding methods are named *a posteriori* methods. Once the Pareto optimal solutions are presented, the decision maker chooses their most preferred solution. According to Miettinen (1994), this type of methods have the advantage of providing the decision maker with a wide range of solutions, leading to a good understanding of the trade-offs between the objective functions. However, it can be time-consuming and resource-intensive to compute a sufficient number of solutions, and the decision maker may face challenges in evaluating a large number of options.

Conversely, if the decision maker states their preferences before optimisation, these preferences are said to be given *a priori*. Hence, the methods are called *a priori* methods. As outlined by Miettinen (1994), the primary challenge with *a priori* methods is that the decision maker may find it difficult to specify their preferences in advance. This could lead to their preferences being overly optimistic or pessimistic, potentially resulting in a set of irrelevant solutions for the decision maker.

2.1.2 Interactive methods

As described by Miettinen (1999), interactive methods are designed to address the limitations of both *a priori* and *a posteriori* methods by integrating the decision maker's preferences iteratively into the optimisation process. Miettinen and Mäkelä (2006) state that the goal of interactive methods is to identify a single preferred solution from the set of Pareto optimal solutions.

According to Miettinen, Ruiz, and Wierzbicki (2008), the iterative nature of interactive methods allows the decision maker to operate without a predefined global preference structure. This iterative nature is a key advantage, as it gives the decision maker insight into the problem's scope and its constraints. In essence, interactive methods address the limitations of *a priori* and *a posteriori* methods by generating only those Pareto optimal solutions that are relevant to the decision maker's evolving preferences. This reduces computational demands compared to *a posteriori* methods. Interactive methods also eliminate the complexity of comparing numerous Pareto optimal solutions at once, which is a significant disadvantage when using *a posteriori* methods.

The main characteristic of interactive methods is their ability to provide a dynamic and user-centric approach in solving multiobjective optimisation problems. As outlined by Miettinen, Ruiz, and Wierzbicki (2008), these methods involve an iterative solution algorithm or a solution pattern. An iteration is essentially one step taken along the solution process, often using the preference information provided by the decision maker. During these iterations, the decision maker is engaged directly, offering preference information progressively throughout the process. As a result, the decision maker learns more about the Pareto optimal set and what are the trade-offs among the objective functions.

As Miettinen, Ruiz, and Wierzbicki (2008) outline, the phases of preference elicitation and the solution generation alternate. This iterative dialogue continues until the decision maker identifies the most preferred solution, a predetermined stopping criterion is met, or it becomes apparent that no satisfactory solution can be attained under the current conditions. With each iteration, the decision maker is presented with information derived from the latest optimisation phase, which can then be applied in what follows.

Miettinen, Ruiz, and Wierzbicki (2008) highlight that interactive methods are advantageous due to their ability to streamline the decision-making process. Instead of overwhelming the decision maker with the representation of all Pareto optimal solutions, only a relevant subset of solutions is presented that potentially aligns with the decision maker's preferences.

Moreover, the flexibility of interactive methods allows the decision maker to refine, specify, and alter their preferences as the process unfolds. This is crucial as it acknowledges the often iterative nature of preference formation and decision-making. By allowing preferences to evolve organically during the solution process, the decision maker's understanding of the problem deepens.

2.2 DESDEO

DESDEO (Misitano et al. 2021) is an open-source software framework dedicated to interactive methods for solving multiobjective optimisation problems. According to Misitano et al. (2021), DESDEO's primary goal is to bridge the gap between interactive methods and their practical application by researchers and practitioners. By offering ready-to-use open-source implementations, DESDEO facilitates several multiobjective optimisation methods for solving complex optimisation problems. Due to its modular architecture and role as a repository for existing implementations, DESDEO actively encourages the addition of new implementations supported by its reusable components.

2.2.1 Structure

Figure 2 illustrates the structure of the DESDEO framework, which is composed of four distinct packages. These packages communicate through predefined channels, ensuring the reusability of components across different layers. This design mirrors the framework's modular structure, ensuring that each layer operates independently and can be modified as necessary.

In the DESDEO framework, the decision maker directly interacts with the elicitation layer, typically through an interface. This layer is responsible for gathering preference information from the decision maker and transforming it into a format compatible with the selected

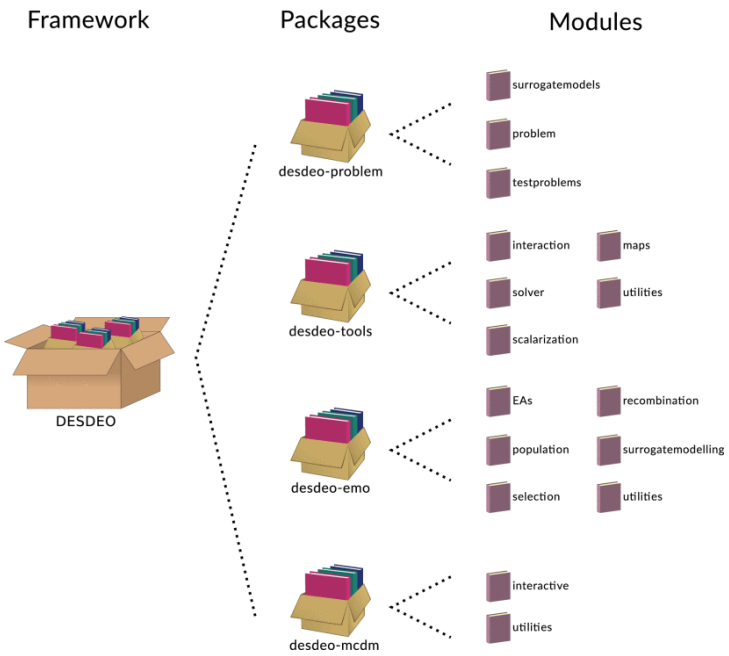

Figure 2. Overview of the DESDEO software framework as outlined by Misitano et al. (2021)

interactive method. Additionally, it compiles information generated during the interactive solution process and formats it for display to the decision maker via the user interface. At the top level, the problem layer contains the problem model, defining the multiobjective optimisation problem to be solved.

2.2.2 Modularity and problem formulation

Each package has its purpose, and as the authors of DESDEO (Ojalehto and Miettinen 2019) point out, the key point of the structure of the DESDEO framework is its generality. This design choice allows various components to be interchangeably replaced as required, and the end-goal for DESDEO's development is to make everything modular. For example, different interactive methods can be utilised to solve the same multiobjective optimisation problem without the need to modify methods' implementations.

Figure 3 illustrates the dependencies between each package in DESDEO through arrows from Misitano et al. (2021). For instance, the `desdeo-mcdm` package relies on both `desdeo-tools` and `desdeo-problem`. This modular architecture enables users to selectively use

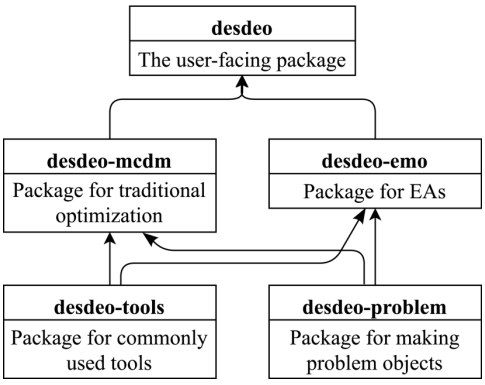

Figure 3. DESDEO's modular structure (Misitano et al. 2021)

parts of the framework as needed. For instance, to construct a multiobjective optimisation problem, one might only use the `desdeo-problem` package, thereby bypassing the integration of other unrelated packages. Furthermore, this modular design simplifies the framework's development by isolating features and functions relevant to interactive multiobjective optimisation within distinct packages.

The focus of this work is the implementation of a user interface for NAUTILUS. As Misitano et al. (2021) explain, the `desdeo-problem` package is responsible for the problem formulations in DESDEO. For example, problems in `desdeo-problem` can be represented as analytical expressions of functions that depend on decision variables. The NAUTILUS method, for which the user interface will be developed, has already been implemented in the `desdeo-mcdm` package.

The formulation of problems is not within the scope of this work. We will assume that the problems provided by the `desdeo-problem` package are correct and ready for use in the development of the NAUTILUS user interface. Once the user selects the desired problem from the `desdeo-problem` package, the NAUTILUS user interface communicates solely with the NAUTILUS method in the `desdeo-mcdm` package.

2.3 Research approach

The research method used in this thesis follows the design science research approach as outlined by Hevner and Chatterjee (2010) and Peffers et al. (2007). The research results in the

creation of an artefact, which is the implementation of the user interface for the NAUTILUS method.

2.3.1 Methodology and research question

In this subsection, we briefly overview five design science methodology activities defined by Peffers et al. (2007). These activities offer a structured road-map for conducting design science research, with each representing a step in the development and assessment of the artefact. We outline these activities here and apply them to the implementation of the NAUTILUS user interface. It should be noted that in this context, the terms 'problem', 'objective', and 'solution' have different meanings compared to their use previously. Here, a problem refers to the lack of a user interface, an objective refers to the creation of this interface, and a solution refers to the implemented user interface.

Activity 1: Problem identification and motivation. The initial step in this research is to present the specific problem and explain the motivation behind seeking a solution. For this study, the problem is identified as the absence of a user interface implementation for the NAUTILUS method. This problem is technical and conceptual. This means that it involves the technical implementation and some documentation of the user interface, as well as understanding the theory behind multiobjective optimisation in order to create an artefact that effectively helps users in solving multiobjective optimisation problems. These matters will be discussed in detail in a later chapter regarding the implementation details of the user interface.

Activity 2: Define the objectives. The second activity is to define objectives for the solution, drawn from the problem. They provide direction to the development process and help to maintain alignment with the intended outcomes. In the context of this research, the objective is to create a user interface that enables the interaction with the NAUTILUS method. The design of the user interface is not part of this work, instead, the main focus is in the implementation. The objective is qualitative: the artefact created addresses the need for a user interface implementation.

Activity 3: Design and development. The third activity is the creation of the artefact

itself. According to Hevner and Chatterjee (2010), artefacts can take various forms such as constructs, models, methods, or instantiations, all of which are construed in a broad sense. In the context of this research, the artefact in question is the user interface for the NAUTILUS method. The development involves specifying the desired functionalities of this user interface. The goal is to implement the artefact so that it can effectively address the problem identified in Activity 1.

Activity 4: Demonstration. The fourth activity of a design science research is the demonstration of the artefact. This is about showcasing the user interface, and demonstrating that it can effectively solve a multiobjective optimisation problem. Demonstrating the artefact's utility is critical in validating its effectiveness and practicality.

Activity 5: Evaluation. The fifth activity is the evaluation of the effectiveness of the artefact in addressing the identified problem. This phase involves assessing how well the user interface implementation fulfils its intended purpose and how well it solves the problem it was designed to solve. This can be understood as a comparison of the solution's objectives, as defined in Activity 2, against the actual outcomes derived from the artefact's application during the demonstration phase.

As said, the aim of this research is to implement a user interface for the NAUTILUS method. Given that the method's implementation is already complete, there is a clear need for a corresponding user interface. Since DESDEO is constantly expanding, adding new features and methods, it is natural to enhance DESDEO's usability with user interfaces. This research has the following research question:

How to successfully implement the user interface of the NAUTILUS method in the context of DESDEO?

2.3.2 Literature review

In this literature review, we examine the relevant literature for this thesis. The literature search was conducted in three main phases, corresponding to the three primary topics of this thesis: multiobjective optimisation, the NAUTILUS method, and design science research. Although this division is broad, there are overlapping areas among these topics. This means

that, for example, a keyword or a relevant combination of multiple keywords used in one phase of the literature search were also utilised in another phase. These phases are not necessarily in chronological order but rather represent a distinction between different literature searches.

Phase 1: Multiobjective optimisation

At the outset of this thesis, I was provided with several references, including a book by Miettinen (1999) and a chapter of a book by Miettinen, Ruiz, and Wierzbicki (2008). Utilising the book by Miettinen (1999), I established a foundation of theories and practices, upon which I built a solid theoretical background for multiobjective optimisation. Subsequently, I expanded my knowledge of interactive multiobjective optimisation through the book by Miettinen, Ruiz, and Wierzbicki (2008).

Given that the main goal of this thesis was to develop a user interface for NAUTILUS—an interactive multiobjective optimisation method—understanding how interactive methods function and how the decision maker is involved in the decision-making process naturally became a central focus of my research. These two books are some of the fundamental pieces of literature in the field of multiobjective optimisation, and they equipped me with the necessary information to understand complex theories in interactive multiobjective optimisation, and laid the groundwork for implementing an effective solution to the research question.

Utilising the literature referenced above, I was able to gather some understanding of what keywords and what combination of them I would use for the search queries. The keywords that I selected were 'multiobjective optimisation' and 'interactive multiobjective optimisation'. I began searching for literature using Web of Science and Scopus databases. Different query structures were used in Web of Science and Scopus due to the syntax specific to each database, but the keywords remained the same. In the first phase of gathering literature, a paper was included in the thesis if it met the inclusion criteria, which required that the title, abstract, or keywords contained any of the search terms or any relevant new combinations of them. The inclusion criteria remained the same for subsequent phases, but naturally involving different keywords. A paper was excluded if it did not address interactive multiobjec-

tive optimisation. From this search, I acquired two papers in addition to the two previously mentioned books. From these sources, I learned about the general taxonomy of interactive methods according to Xin et al. (2018), and gained further insights into interactive methods from the work of Miettinen and Mäkelä (2006).

Phase 2: NAUTILUS

As the background theories and knowledge began to form, I shifted the focus of my research to the NAUTILUS method. At the start of my research, a third key piece of literature was provided to me about the NAUTILUS method, written by Miettinen et al. (2010). This article specifically addresses the NAUTILUS method for which the user interface is being implemented. At the same time I was provided with these pieces of literature, I was informed that there are multiple interactive methods that utilise the same approach as NAUTILUS.

The inclusion and exclusion criteria for this phase were quite simple. Since the development of the user interface is specifically for the NAUTILUS method, I searched for papers using the keyword 'NAUTILUS'. There were no specific exclusion criteria for this phase other than establishing that I had acquired sufficient information from the obtained literature. This search led me to discover the article by Miettinen and Ruiz (2016). Forward and backward searches for this paper were conducted. In a forward search, papers that have cited the current paper are reviewed according to the inclusion criteria. Similarly, in a backward search, we examine the references cited by the current paper.

The process described above led me to discover a family of NAUTILUS methods. Reading about other methods, helped me broaden my knowledge of the NAUTILUS method by Miettinen et al. (2010), including its drawbacks, which other methods aim to improve. These methods that were found were E-NAUTILUS by Ruiz et al. (2015), O-NAUTILUS by Saini et al. (2022), NAUTILUS 2 by Miettinen et al. (2015) and NAUTILUS navigator by Ruiz et al. (2019). While these methods share the principle of starting the optimisation process from an inferior point and gradually moving towards a preferred solution without making trade-offs, they differ, for example, in how the decision maker provides preference information, and in their computational approaches to calculating solutions. Understanding these aspects

is important when implementing the user interface for NAUTILUS, as well as for potential future research. Such knowledge can inform improvements to the existing NAUTILUS user interface or guide the development of user interfaces for other members of the NAUTILUS family.

Phase 3: Conducting design science research

This phase involves the literature search for the development and evaluation of the NAUTILUS user interface. As the theoretical foundation and necessary background information began to take shape, it became clear that this research would be conducted using principles from design science methodology. Since I was implementing a user interface within the context of interactive multiobjective optimisation, I needed references relevant to this context. Additionally, early in this research, I was introduced to the DESDEO software framework, where the NAUTILUS user interface is to be implemented.

I conducted a thorough literature search on the above aspects using keywords such as 'design science', 'design science evaluation', 'visual interactive multi-objective optimization', 'interactive multi-objective optimization user interface' and 'DESDEO'. Similarly, I performed backward searches examining the papers cited by the current paper, as well as forward searches, applying the inclusion and exclusion criteria in each search.

Two foundational papers in the field of design science for conducting a design science research, as well as evaluating the resulted artefact were given at the start of this research, and were authored by Hevner and Chatterjee (2010) and Peffers et al. (2007). From the references listed in these papers, I did a forward search that identified additional articles that provided information for implementing and evaluating an artefact. These articles were written by Venable, Pries-Heje, and Baskerville (2012), Prat, Comyn-Wattiau, and Akoka (2015), and Wang and Wang (2010).

I included three articles written by Hakanen, Miettinen, and Matković (2021), Tarkkanen et al. (2013) and Smedberg and Bandaru (2023). From these articles I found information that met my specific user interface development needs in the context of interactive multiobjective optimisation. Also, as visualisation is a key concept in implementing user interfaces, I

included an article by Smedberg and Bandaru (2023) to gain a deeper understanding of the visualisation techniques used in user interfaces for multiobjective optimisation methods.

I included articles by Misitano et al. (2021) and Ojalehto and Miettinen (2019) about the DESDEO framework. These two articles provided me with the necessary knowledge about the general structure of DESDEO and the design principles employed there, which I have applied in the development of the NAUTILUS user interface. In order to test the NAUTILUS user interface, a case study is required to verify that the interface functions as intended and fulfils its purpose. For this case study, I utilise a problem initially introduced by Narula and Weistroffer (1989) and subsequently modified by Miettinen and Mäkelä (1997).

As I began developing the user interface, I started searching for articles related to web development. I initially used keywords such as 'web application' and 'web application development', which yielded a vast number of articles. To exclude articles, I opted to focus on articles that were not necessarily comprehensive in theory but rather highlighted general trends in web development. This led to an article that was included written by Jazayeri (2007), that would satisfy needs for web development principles.

At this point, the literature structure for my thesis was nearing completion. However, I felt the need to further explore the various ways in which decision makers can provide preference information, so I included a book by Hwang and Masud (1979). Discovering this book effectively completed the literature framework of my thesis.

2.4 Web development

According to Jazayeri (2007), web applications are commonly structured around a client-server architecture, where the server holds data or documents, and the client sends requests to retrieve or interact with this data. This interaction is made possible using the Hypertext Transfer Protocol (HTTP), employing a request-reply mechanism. Specifically, a client may request resources through a GET request or submit data to the server using the POST method. In alignment with these principles, the implementation of the user interface is built upon the same client-server model. This approach ensures that the NAUTILUS method seamlessly integrates into the web-based infrastructure, and more importantly, it is concordance with

methods that are already in DESDEO.

The primary contribution in this thesis is not the creation of a new user interface design, but rather following an existing one. This baseline design serves as a template for our implementation. When using multiobjective optimisation methods, a user interface should collect preference information from the decision maker and present solutions along with relevant data back to them. Regarding preference information input, the implemented user interface of NAUTILUS will be able to handle preference information from the user, as well as to communicate with the server. Regarding the presentation of optimisation results, which can include intermediate solutions between iterations and Pareto optimal solutions, the user interface should display them in a logical and straightforward manner.

Given the modular nature of DESDEO, it is essential to ensure that the NAUTILUS user interface is also developed modularly. This approach supports the specific needs of the NAUTILUS method and allows the components of the NAUTILUS user interface to be utilised across different methods already existing within the DESDEO framework, as well as in user interfaces that will be developed in the future. The ability to have shared components streamlines the development process and scalability of the DESDEO framework.

3 NAUTILUS

This chapter presents an overview of the NAUTILUS method. It explores the operational framework of NAUTILUS and its approach to solving multiobjective optimisation problems. There will also be a detailed walk-through of the NAUTILUS's algorithm. As discussed in Chapter 2, the approach to solving multiobjective optimisation problems typically revolves around finding Pareto optimal solutions. In this context, the decision maker is required to consider various trade-offs between conflicting objectives, necessitating compromises in one objective to achieve gains in another.

3.1 Trade-off free optimisation

According to a prospect theory proposed by Kahneman and Tversky (1979), people do not respond symmetrically to gains and losses; instead, our reactions to losses are more pronounced than those to gains. This could mean that when applying the principles of prospect theory to decision-making, one should account for the likelihood that individuals will weigh potential losses more heavily than equivalent gains. Additionally, as noted by Miettinen et al. (2010), there is a tendency among decision makers to favour a specific solution, known as the anchoring effect. This means that our past experiences can shape and potentially limit our expectations, leading decision makers to gravitate towards solutions influenced by those factors.

Miettinen et al. (2010) introduces the first NAUTILUS method, which is used in the user interface implementation. Later Miettinen and Ruiz (2016) introduce a framework of methods called the NAUTILUS framework that is designed to tackle these problems mentioned above. Methods in the NAUTILUS framework are interactive multiobjective optimisation methods, allowing for a search for Pareto optimal solutions without requiring the decision maker to make trade-offs. The interactive solution process begins either from the nadir objective vector or another point where improvements in each objective function are possible. As outlined by Miettinen and Ruiz (2016), through iterative steps, the decision maker progressively approaches the set of Pareto optimal solutions, achieving improvements in all

objective function values with each iteration.

There are multiple variants of NAUTILUS methods within the framework, including an enhanced version of NAUTILUS known as E-NAUTILUS, developed by Ruiz et al. (2015). Additionally, the framework includes the Optimistic NAUTILUS, known as as O-NAUTILUS developed by Saini et al. (2022). These methods are all part of the NAUTILUS family. Additionally, the NAUTILUS Navigator by Ruiz et al. (2019) is part of the family, though not included in the framework.

These are all continuations and further developments of the initial version of NAUTILUS presented by Miettinen et al. (2010). Each version of NAUTILUS aims to enhance the efficiency of solving multiobjective optimisation problems by improving, for example computational capabilities beyond those of the previous one. The primary focus of this thesis is on the NAUTILUS method that was originally presented by Miettinen et al. (2010). For the remainder of this thesis, any references to the NAUTILUS method will specifically refer to the method described in that article.

In the beginning of solving a problem with NAUTILUS, the decision maker is asked to provide the total number of iterations, denoted as itn. It represents the number of iterations that NAUTILUS uses to go from the nadir point to a Pareto optimal solution. At each iteration, the so-called current iteration point moves closer to the Pareto optimal set. In what follows, we denote by h the current iteration, $h-1$ previous iteration and z^h the corresponding objective vector to that iteration, referred to as the *current iteration point*, and previous iteration point z^{h-1}.

Then we denote by it^h a number representing how many iterations are left. Therefore, we assume that in the first iteration $it^1 = itn$, and as NAUTILUS starts from the nadir point, $z^1 = z^{nad}$. With each iteration, the range of achievable values for each objective function generally shrinks for the current and subsequent iterations. These values are referred to as reachable values, and we denote the lower and upper bounds for these reachable values as $z^{h,lo} = (z_1^{h,lo}, \ldots, z_k^{h,lo})^T$ and $z^{h,up} = (z_1^{h,up}, \ldots, z_k^{h,up})^T$ for each iteration h. These are the upper and lower bounds of what each objective function can have during each iteration. As explained, NAUTILUS starts from the nadir point, so as initial values, we set upper bound

to the nadir point, $z^{1,up} = z^{nad}$, and lower bound to the ideal point, $z^{1,lo} = z^*$.

The decision maker can then provide their preference information to set the direction of approaching Pareto optimal solutions. In NAUTILUS, the preference information can be given in one of two ways in each iteration. The first way can be understood as giving a rank to the objectives functions based on the relative importance of improving each objective value, and the second as distributing a fixed amount of improvement potential to each of the objective function values in z^h. The two ways of providing preference information are defined as follows:

1. The decision maker is asked to rank objective functions based on the relative importance of improving each current objective value in the current iteration point. The decision maker categorises objective functions into classes, in the order of importance for improving the current objective function value. This means that the decision maker allocates the objectives into sets, denoted by J_r. These sets represent the importance levels $r = 1, \ldots, s$, where $1 \leq s \leq k$. If $r \leq t$, this means that improving the objective values in set the J_r is less important than improving them in the set J_t. Each objective function must be assigned to one set. One objective can be assigned to only one index set, although multiple objectives may be assigned to the same set.
2. The decision maker can assign weights to the objectives by using this question as a reference: *Given a hundred points to distribute among the current objective function values, how should they be allocated so that the more points an objective function receives, the greater the desired improvement is for that objective function's value?*

Using z^{h-1} and preference information provided by the decision maker using one of two ways defined above, we denote by x^h a Pareto optimal solution [1]. When we have x^h, we can calculate the current objective value $f^h = f(x^h)$. Using this information, the next iteration point is calculated as:

$$z^h = \frac{it^h - 1}{it^h} z^{h-1} + \frac{1}{it^h} f^h. \tag{3.1}$$

1. For details on how this is done, see Miettinen and Ruiz 2016.

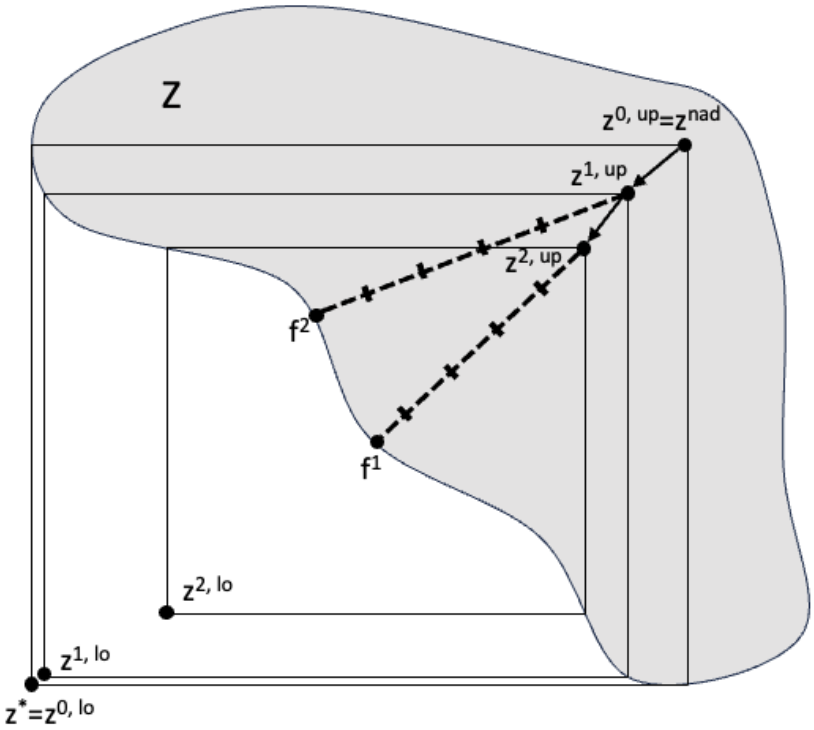

Figure 4. Graphical idea of NAUTILUS, adapted from Miettinen et al. (2010)

Figure 4 illustrates the solution process where three steps have been taken and iteration points are calculated using (3.1). The figure demonstrates how the decision maker's preferences determine the direction of what Pareto optimal solution can be reached with the current preference information. During each iteration, z^h is situated on the line connecting z^{h-1} and f^h. In the figure we illustrate two different possible Pareto optimal solutions denoted by f^1 and f^2 depending on the preference information. Additionally, we can see the shrinking of reachable values. With every iteration, as new current iteration point z^h is calculated, the endpoints of reachable ranges draw closer together.

Consequently, a portion of the Pareto optimal set becomes inaccessible, unless a step back is taken. A step back means that the decision maker can go back to the previous iteration and change the preference information. The figure shows that the decision maker decided to take a step back from z^2 to z^1 and change the preference information. The new direction of what Pareto optimal solution can be reached is illustrated as f^2. It is also possible that the decision maker has no desires to change the preference information when taking a step back. In this case, a new iteration point is calculated with the same preference information but using a

smaller step size, making it z^h and the next iteration is taken from there.

The decision maker is not locked into the amount of iterations, or the preference information that was given at the start. In every iteration, the decision maker has the option to adjust the number of remaining iterations or modify the preference information or alter both. Since NAUTILUS operates iteratively, the process of giving preference information and calculating subsequent iteration points will continue until the last iteration is reached. The last iteration point is a Pareto optimal solution.

At each iteration, the decision maker is provided with distance d^h from the current iteration point to the Pareto optimal set and it is calculated as:

$$d^h = \frac{\|z^h - z^{nad}\|_2}{\|f^h - z^{nad}\|_2} \times 100. \tag{3.2}$$

In (3.2), $\|z^h - z^{nad}\|_2$ represents the Euclidean distance between z^h and z^{nad}. Similarly, $\|f^h - z^{nad}\|_2$ is the Euclidean distance between the f^h and z^{nad}. If the current iteration point is the nadir point ($z^h = z^{nad}$), the Euclidean distance between these two points is zero so then then $d^h = 0$. And if the current iteration point is the same as the current objective vector ($z^h = f^h$), then $d^h = 100$. This means that the greater value d^h has, the closer that corresponding iteration point is to the Pareto optimal set.

3.2 Algorithm

The following is the description of the NAUTILUS algorithm by the authors of NAUTILUS Miettinen et al. (2010).

Step 0. Initialisation. For problem (2.1), calculate estimates of z^* and z^{nad}. Ask the decision maker to give itn. Set $h = 1$, $z^0 = f^{up} = z^{nad}$, $f^{lo} = z^*$ and $it^1 = itn$.

Step 1. Preference information I. The decision maker is tasked with selecting the type of preference information they prefer. The decision maker can choose to rank the objective functions to reflect the relative importance of improving each current objective value, or give percentages indicating how they would like to improve the current ob-

jective function values.

Step 2. New solution. Calculate x^h and the corresponding objective vector f^h.

Step 3. New iteration point. Calculate the new current iteration point using (3.1).

Step 4. Bounds for the next iterations. Update f^{up} and f^{lo}. And calculate the distance d^h to the Pareto optimal set using formula (3.2).

Step 5. Show current iteration point. Show the current iteration point z_i^h $(i = 1, \ldots, k)$, together with the new upper bounds, lower bounds and d^h to the decision maker.

Step 6. Change the number of iterations. If the decision maker wants to change the number of remaining iterations, set it^h accordingly.

Step 7. Preference information II. If the decision maker wants to take a step back, go to step 9. Otherwise, continue.

Step 8. Next iteration. If $it^h = 1$, stop and assign the latest solution x^h and the corresponding objective vector f^h as the final solution. If not, set $it^{h+1} = it^h - 1$ and $h = h + 1$. If the decision maker wants to give new preference information, go to step 1. If the decision maker decides to take a new step in the same direction using the same preference information as in the previous iteration, then set $f^{h+1} = f^h$ and go to step 3.

Step 9. Preference information III. Ask the decision maker for new preference information from the iteration point z^{h-1}. If given, go to step 1. If the decision maker opts to take a shorter step using the same preference information provided in step 1, then set $z^h = \frac{1}{2}z^h + \frac{1}{2}z^{h-1}$ and proceed to step 4.

To summarise the algorithm, after initialising the iterations and providing the initial preference information, starting from the nadir point, NAUTILUS moves into a loop where the decision maker provides preference information to obtain a new iteration point. Based on the decision maker's preferences, they may adjust it^h or take a step back to the previous iteration. From there the decision maker can take a smaller step to the same direction or provide new preference information. If there is no change in the iteration number in the last iteration, the process ends to the last iteration. Figure 5 presents a flowchart of the NAUTILUS algorithm to clarify the steps involved in the algorithm.

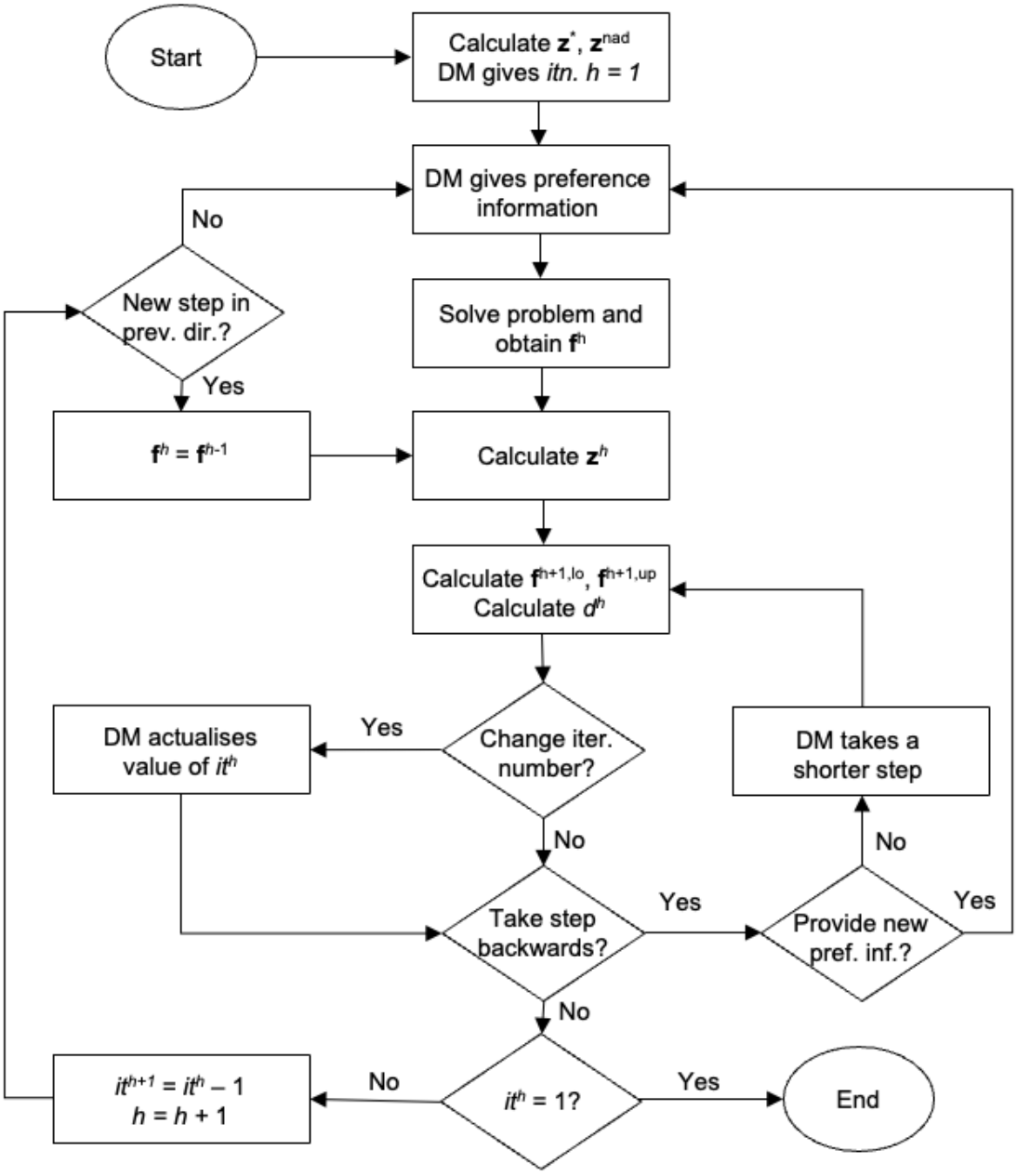

Figure 5. Flowchart of the NAUTILUS algorithm adapted from Miettinen et al. (2010)

3.3 Some comments about NAUTILUS

To conclude this overview of NAUTILUS, let us raise some additional comments as outlined by Miettinen et al. (2010). It should be noted that as the solution process advances from the nadir point towards the Pareto optimal set, the iteration points that are generated exist solely within the objective space, meaning that corresponding decision vectors may not exist in the decision space. Nonetheless, according to Miettinen et al. (2010), it is guaranteed that there is always a feasible solution and an associated objective vector better than the current iteration point.

It is also important to recognise that at each iteration, the Pareto optimal objective vector f^h is calculated and can be made available to the decision maker if they wish to see it. This vector can provide insights into the nature of the Pareto optimal solution that will be achieved if further iterations continue in the same direction. However, revealing this vector could potentially anchor the decision maker's expectations that NAUTILUS tries to avoid. Consequently, it is up to the decision makers whether they would like to have this information.

4 Implementation

In this chapter, we focus on the implementation details in greater depth. We highlight the aspects that are needed to address the research question. The focus of this work is not in the design aspects of the user interface. The user interface is developed using a given design that is expected to evolve during the development phase.

4.1 Problem identification and motivation

This research is conducted as a design science research, at the heart of which is the construction of an artefact. The problem addressed is the absence of a user interface for the NAUTILUS method within the DESDEO software framework. The implementation of the NAUTILUS user interface introduces three challenges. The first challenge is how to implement the information exchange process between the user solving the problem on one device, and a separate server running the optimisation method. The second challenge is how to accommodate different expertise levels of users. The expertise level of a user may range from having never interacted with any multiobjective optimisation method, to being very familiar with using such methods. The third challenge is how to gather and visualise information to the user throughout the optimisation process. By focusing on these challenges when developing the user interface, this research aims to provide a practical solution to the problem discussed above. In the following three subsections, we elaborate on the three challenges.

Information exchange process

From now on, we will refer to the user interface of NAUTILUS as the UI. When the user interacts with the UI, it sends and receives information as requests to and responses from the server. These requests are constructed in the front-end when the user interacts with the UI, and contain information such as preference information, whether to take a step back or to use preference information from the previous iteration. Requests are handled by the NAUTILUS endpoint on the server. An endpoint in this context refers to a specific URL or address on a server where the NAUTILUS method receives requests from and sends responses back to

the UI, facilitating communication between the user interface and server-side processes. The challenge is how to implement the exchange of information between (a) the user, and (b) the UI, as well as between the UI and the NAUTILUS endpoint. The aim is to create an information exchange process that minimises the risk of errors and ensures that the method operates with high efficiency and reliability.

Since the implementation of this artefact focuses specifically on the UI, this imposes another aspect related to this challenge, that is how to handle the different formats of information needed by NAUTILUS. When the user interacts with the UI, they can provide two different kinds of preference information, as discussed in Chapter 3. The preference information needs to be gathered from the user and then integrated into a request that uses the same endpoint for both preference information types.

User's expertise and experience level

The second challenge is to make the UI to cater to users with varying levels of experience in multiobjective optimisation, including those who are new to using NAUTILUS or to solving multiobjective optimisation problems in general. No matter the experience level of the user, the UI should be usable and easy to follow. To address this, the UI should be structured in such a way that it clearly guides the user where to begin and which component to use next. This reduces the time it takes the user to get acquainted with the UI. By disabling components that should not be interacted with prevents misclicks and guides the user how to use the UI. In addition, information boxes or texts are also incorporated. These visual aids would provide context-specific assistance to the user, and help users to take steps along the solution process.

Utilisation of information

Although the primary focus of this thesis is on implementing the UI rather than its design, certain design choices are made during the implementation process. These include identifying the most relevant information for the user and presenting it in a clear, concise, and visually appealing manner. This challenge can be seen as an extension of the previous one.

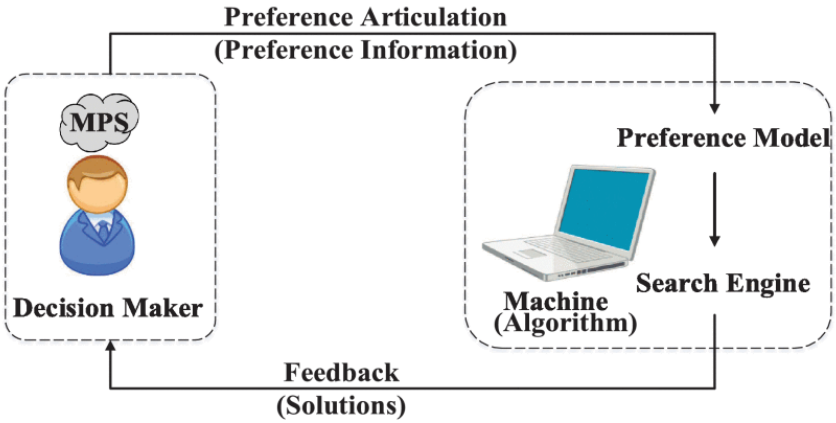

Figure 6. The taxonomy of interactive multiobjective optimisation methods illustrated by Xin et al. (2018)

Meaning, while the previous challenge focused on identifying key aspects that guide and assist the user, utilisation of information focuses on how to translate those aspects into visual elements. For instance, the UI has components to visualise how reachable values shrink after every iteration using color-coded bars. Additionally, the interface allows users to change preferences information type between 'Weight' and 'Rank' modes using a toggle switch.

4.2 Describing the development process

In this section, we provide a detailed description of the development of the UI. According to Misitano et al. (2021), in the development of DESDEO, multiple design choices were made to shape the framework's functionality. Some of these are incorporated to the UI and they are explained in detail in this section. Additionally, we present the structure and contents of the information that is composed and exchanged with the server in the background.

To begin describing the development process, we first highlight the general structure of interactive multiobjective optimisation methods outlined by Xin et al. (2018). In Figure 6, a general taxonomy of interactive multiobjective optimisation methods is illustrated. Accord-

ing to Xin et al. (2018), three key design aspects of interactive multiobjective optimisation methods are apparent, and are illustrated in the figure: the preference information, the preference model, and the search engine.

In the development of the UI, the preference information consists of ranks and weights, as described in Chapter 3. The preference model is the function used inside NAUTILUS that calculates the current objective values, and the search engine is the optimiser used inside NAUTILUS. These two elements together form the algorithm visualised in Figure 6, which is based on the server and is accessed through the NAUTILUS endpoint. This idea is adapted to the NAUTILUS context presented here.

According to Xin et al. (2018), the interaction unfolds as follows: the decision maker provides preference information based on an understanding of the problem. The UI passes this information to NAUTILUS, which constructs the model, and this model exists inside NAUTILUS. In other words, the UI establishes a communication channel between the decision maker and the NAUTILUS method. Lastly, NAUTILUS uses the model to generate solutions that interest the decision maker.

It should be noted, that while Xin et al. (2018) describe the solution process as the algorithm giving solutions to the decision maker before the decision maker gives a new set of preference information, being a general idea of the taxonomy of interactive methods, this is not exactly the case with NAUTILUS. As mentioned in Chapter 3, a decision maker obtains a Pareto optimal solution to a problem in the last iteration, so solutions before the last iteration can be understood as intermediate feedback on the shrinking of the reachable values.

4.2.1 Information exchange structure

In Chapter 2, we discussed how interactive multiobjective optimisation methods differ from other approaches, such as *a priori* and *a posteriori* methods, in terms of the type of interaction and preference information they receive from the decision maker. Because of this, Misitano et al. (2021) have designed the DESDEO framework with a simple and flexible abstraction for interaction.

A sequence diagram is presented in Figure 7, visualising the exchange of information be-

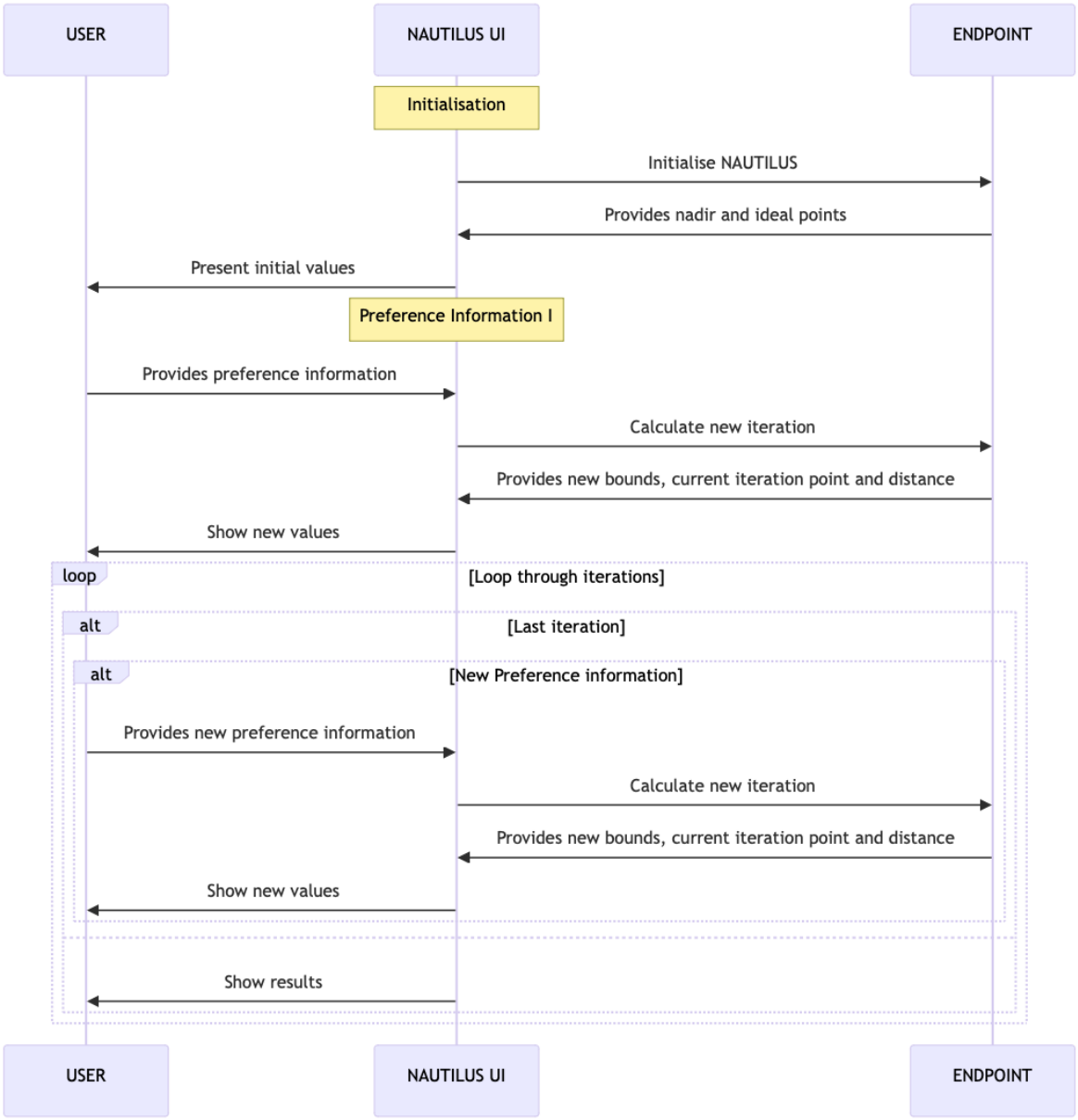

Figure 7. Sequence diagram of information exchange of NAUTILUS

tween the user and the UI, as well as between the UI and the NAUTILUS endpoint. The process of solving the optimisation problem with NAUTILUS begins when the user selects NAUTILUS as the method for solving the problem.

The initiation of the NAUTILUS method sets the stage for subsequent interactions within the endpoint by establishing an instance of NAUTILUS in the server. The initialisation response from the endpoint provides the estimated ideal and nadir points as the initial upper and lower bounds for the Pareto optimal set. The response also contains a message entry, which is not visible to the user. This message is designed to inform developers about how the request object should be constructed at that stage of the process, serving as guidance for those working on implementations.

The initial bounds are shown to the user through the UI. Upon viewing these values, the user first selects the type of preference information—either ranks or weights—as described in Chapter 3. This choice determines the preference handling section of the UI, and then the user can provide the selected preference information in the specified format. Once the user has finalised the preference information, the user proceeds to take the first iteration and the request is sent to the endpoint. In response, the endpoint provides new bounds of reachable values as new information to show to the decision maker for subsequent iterations, and it also includes the current iteration point, which corresponds to the same values as the new upper bounds in the case where the objective function is to be minimised. Also, the distance of the current iteration point to the Pareto optimal set is showed which can be used as a measure of progress to display to the user through the UI.

The previous steps can be repeated until the decision maker reaches the last iteration. At any given point, the user has the opportunity to go back to the previous iteration. If this option is chosen, the most recent iteration and its associated information are removed from the UI. Subsequently, the user has the option to either provide new preference information for the next iteration or proceed with the existing preference information. In the former case, the user gives new preference information and it is added to the request and sent to the endpoint. The endpoint then responds in the same manner as it does for any other iteration providing current iteration point, bounds and the distance. In the latter case, a shorter step is taken in the same direction as the deleted iteration. Information that the user takes a short step

is added to the request and the endpoint then answers, but the distance moved towards the Pareto optimal set in that iteration is shorter.

It should be noted, and as mentioned in Chapter 3, that at any given moment, the user has the opportunity to change the number of iterations to be taken to reach a Pareto optimal solution. If the user changes the number of remaining iterations, no separate request is sent to the endpoint; instead, the new iteration count is simply included in the following request, and the NAUTILUS endpoint calculates the following iteration results accordingly. Upon completion of all iterations, the optimisation results are sent from the endpoint to the UI.

As discussed in Subsection 4.1, certain UI components must be disabled temporarily to assist the user. While the UI is fetching information from the NAUTILUS endpoint, all buttons that are connected to sending requests should be disabled. This prevents the user from making further requests to the endpoint before receiving a response, thus avoiding potential errors that may occur if a new request is sent before the previous one is fully processed. Additionally, buttons connected to sending requests will be enabled once sufficient preference information has been provided. When using ranking as giving the preference information, all objectives must be assigned to at least one rank container. Similarly, when using weights, at least one objective should have a weight greater than zero.

4.2.2 Weight preferences

In Figure 8, the user has set weights as the preference information type and started to provide weights. The weight component also has the information text to guide the user on how to assign the weights. Each objective is associated with a horizontal scrollbar. The scrollbar works as a slider, which users can use to set the weight to an objective. The slider is situated on a scale of 0 to 100, representing the percentage value, and is color-coded corresponding to a specific objective. An input box directly to the right of the scrollbar, displays the numerical value of the weight. This is another option for users who prefer to input an exact number rather than using the scrollbar.

The way weights are then calculated is as follows: first, the values assigned through the sliders or the input boxes for all objectives are summed up. In Figure 8, the input values

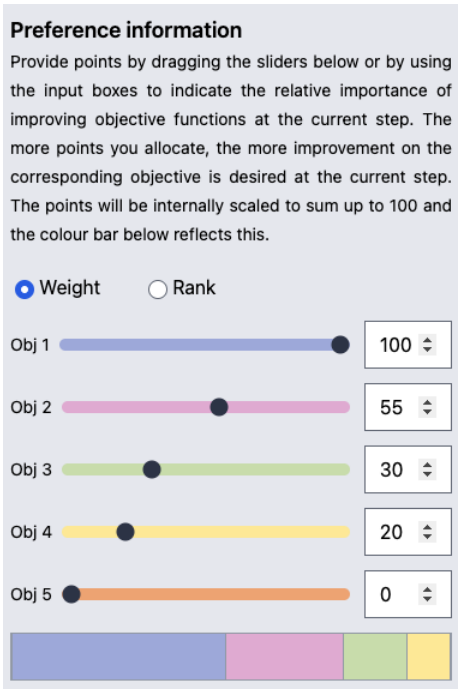

Figure 8. Weights as the preference information type in the NAUTILUS UI

are 100, 55, 30, and 20, which add up to a total of 205. The relative importance of each objective is then calculated by dividing the input value of each objective by this total sum, and then multiplying it by 100. This gives a percentage representation that reflects the weight of each objective relative to the others based on the user input. The calculated weights for the objectives, as shown in the figure, are calculated as follows:

- Objective 1: An input of 100, the calculated weight will be $\frac{100}{205} \times 100 = 48.8\%$.
- Objective 2: An input of 55, the calculated weight will be $\frac{55}{205} \times 100 = 26.8\%$.
- Objective 3: An input of 30, the calculated weight will be $\frac{30}{205} \times 100 = 14.6\%$.
- Objective 4: An input of 20, the calculated weight will be $\frac{20}{205} \times 100 = 9.8\%$.
- Objective 5: With user input being 0, the calculated weight will be 0%.

Directly below the scrollbars, there is a dynamic bar component that visually represents each objective's relative importance. It has the same color codes as the objectives and is displayed as a percentage relative to the others, based on the values set using sliders and input boxes. This bar is updated in real time as the user adjusts the weights.

Figure 9. Ranking as preference information type in the NAUTILUS UI

4.2.3 Rank preferences

Figure 9 demonstrates a situation where the user has decided to use ranking for giving preference information. The ranking component also includes an information text that provides initial instructions on what the user should do. This information is displayed immediately when NAUTILUS is selected as the method to use to inform the user that the ranking is done using a drag-and-drop functionality. In the ranking process, the objectives are visually distinguished by color-coded blocks, with each block corresponding to a specific objective. We use the same colours for the objectives throughout the UI to make it easier for the user to consistently distinguish them. To express their preferences, users can click and hold an objective block, then drag it to their preferred rank's container. Upon releasing the block into one of the labelled rank containers marked as 'Rank 1', 'Rank 2', and so on—the objective is assigned to that rank. To aid in clarity, the UI rank containers include a visual confirmation when the user is hovering on a specific rank. The container borders change color when an objective block is on top of the container.

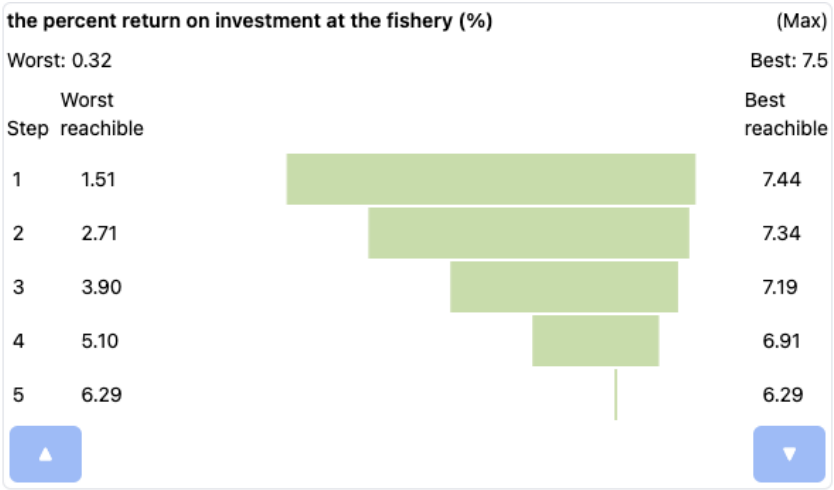

Figure 10. One element of the reachable values section

The user can assign the same rank to any set of objectives by dragging them into the same container, assign a different rank to each objective given that there are as many distinct ranks as there are objectives, or choose any other ranking approach. When an objective is dropped into a container, the decision is not final. The user can easily change the assigned rank at any time by simply dragging and dropping the objective into a different container. Furthermore, the user can reset the ranks by pressing the 'Reset' button, which restores the objectives to their original position above the rank containers. Once the user is satisfied with their ranking preferences, the user proceeds to take a step forward. The process of providing preference information as described, as well as how the user takes steps, are all illustrated in the next chapter.

4.2.4 Reachable values converging to a Pareto optimal solution

According to Smedberg and Bandaru (2023), visualisation plays a crucial role in helping decision makers to choose a preferred solution. It is also essential for delineating the extent of the Pareto optimal set and for tracking the progress of the optimisation. This means that for users with varying levels of familiarity with interfaces like the NAUTILUS UI, the components should help visualise both the upper and lower bounds of the Pareto optimal set, guiding users on what the Pareto optimal solution could be and illustrating the objective values that can be achieved in subsequent iterations.

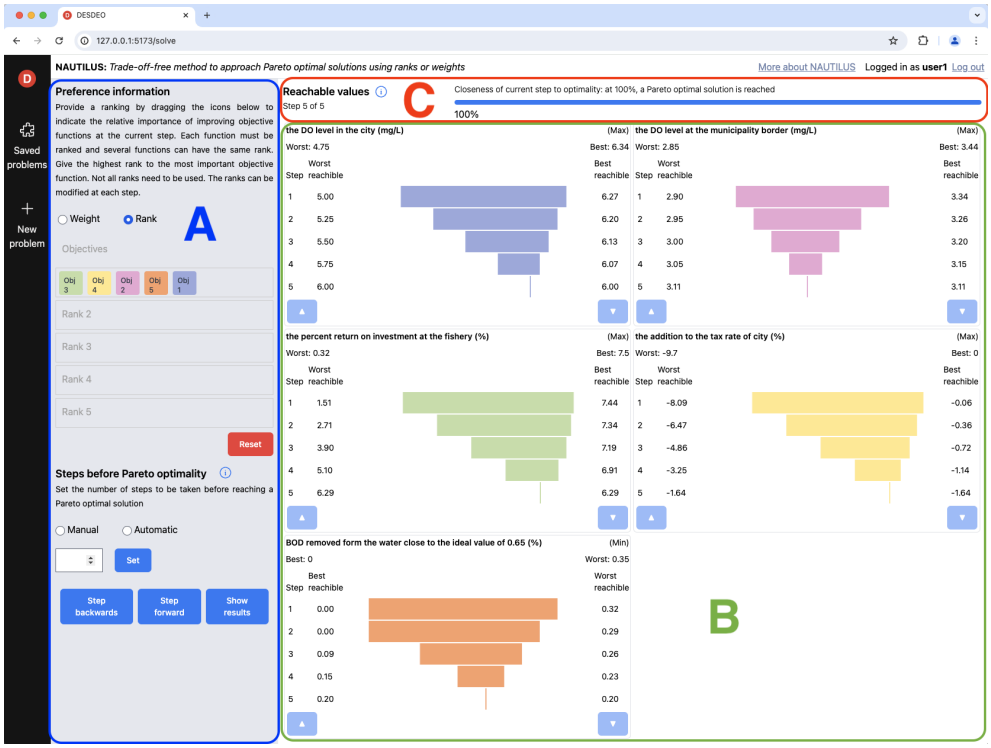

Figure 11. The entire NAUTILUS UI, segmented into three distinct sections.

Illustrated in Figure 10, are iteration bar components, visualising the reachable objective values after each iteration. In the figure after four steps have been taken, the ranges of reachable objective values has shrunk relative to the initial range. The numerical values of the best and the worst reachable values for the respective objectives are in the sides of every iteration bar. Also, the iteration bars accumulate in sequence with each step taken, constructing a visual chronology of the optimisation process. For clarity, the term "step" in the UI refers to iterations. Illustrated in the figure, below iteration bars there are two buttons designed to enable scrolling through the iteration components. Only five rows are displayed at a time, so these buttons provide a scrolling functionality, allowing the user to select which rows are visible when there are more than five iterations set by the user.

Outlined by Smedberg and Bandaru (2023), methods that utilise ideal and nadir points, such as NAUTILUS, these points should be integrated into the UI to provide aid to the users as justified by the reasons explained above. Figure 11 displays the entire user interface, which is segmented into three distinct sections: (A) the section where the user provides preference information, (B) the section that displays objective details and the ranges of reachable values,

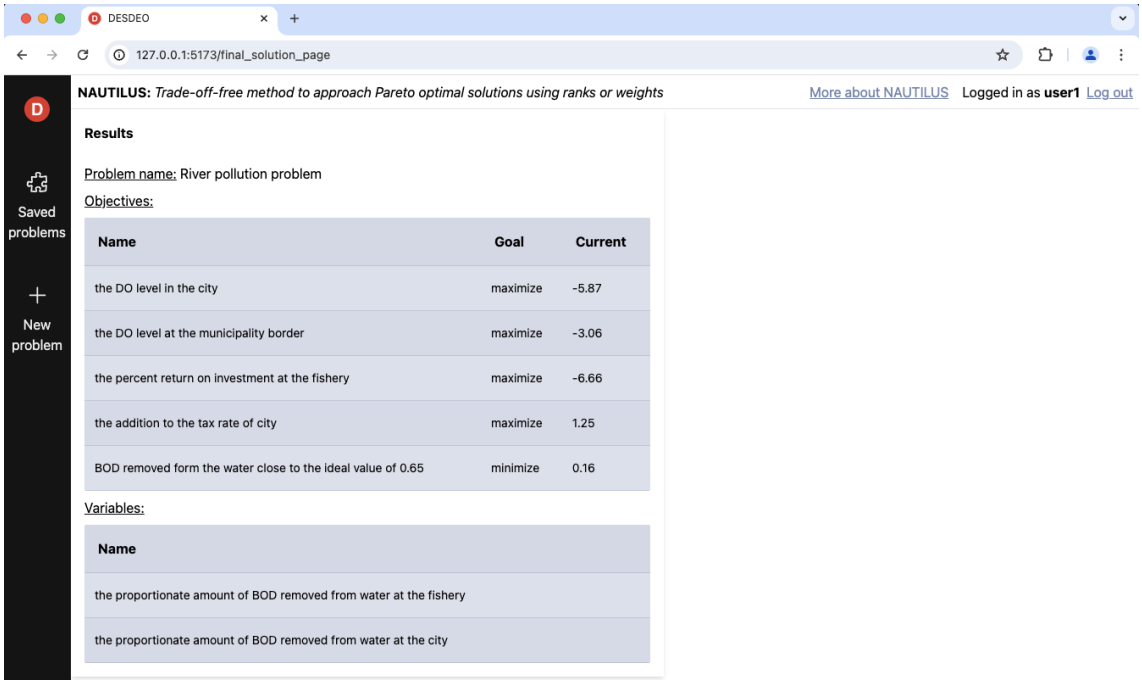

Figure 12. Results view showing the selected Pareto optimal solution

and (C) the section where the current distance to the Pareto optimal solution is displayed.

Section (B) features a grid layout where each of the objectives are displayed. Each objective is identified by its name and unit, and whether the objective function is to be minimised or maximised. Underneath each objective details, iteration bars are displayed, representing the results at that step. As the user takes steps forward, a new iteration bar is displayed.

As users advance through the optimisation process, the progress bar visualised in section (C) is updated to signify the approach towards a Pareto optimal solution. As briefly mentioned earlier, when the user is satisfied with setting their preferences, or simply wishes to use the existing ones, they press the 'Step forward' button at the bottom of section A in Figure 11, to take the next step. When pressed, the UI is updated with a new iteration bar in section C of Figure 11 under the previous one, reflecting the newly reached values. When the user reaches a Pareto optimal solution they press the 'Show results' button illustrated in section A of Figure 11.

Illustrated in Figure 12 is the Pareto optimal solution. This page includes the Pareto optimal objective function value for each objective, detailing the problem name and columns of the

objective names, and whether each objective is to be minimised or maximised for the chosen problem.

5 Evaluation

In this chapter, we conduct the evaluation of the UI. Venable, Pries-Heje, and Baskerville (2016) introduce a strategy on how to effectively carry out this kind of an evaluation. To begin, we focus on assessing how the UI performs and ensure that it functions as intended. This is achieved by solving a problem, and recording the outcomes to confirm that the UI effectively fulfils its designed purpose.

5.1 Case study

In this section, we address an exemplary problem to demonstrate the NAUTILUS UI. This problem was originally introduced by Narula and Weistroffer (1989), and later, a fifth objective was added by Miettinen and Mäkelä (1997). The five objective problem used in this demonstration is as follows: There is a water quality management issue focusing on pollution of a river, where a fishery and a city are sources of water pollution. Water quality is measured by the concentration of dissolved oxygen (DO), while industrial and municipal waste pollution is quantified in pounds of biochemical oxygen demanding material (BOD). Current primary treatment facilities cut down the BOD from the fishery and city output by 30 percent. However, additional treatment facilities would lead to a higher tax rate in the city and a lower return on investment for the fishery. The problem is formulated as follows:

$$
\begin{aligned}
&\max f_1(x) = 4.07 + 2.27x_1 \\
&\max f_2(x) = 2.60 + 0.03x_1 + 0.02x_2 + \frac{0.01}{1.39 - x_1^2} + \frac{0.30}{1.39 - x_2^2} \\
&\max f_3(x) = 8.21 - \frac{0.71}{1.09 - x_1^2} \\
&\max f_4(x) = 0.96 - \frac{0.96}{1.09 - x_2^2} \\
&\min f_5(x) = \max\{|x_1 - 0.65|, |x_2 - 0.65|\}
\end{aligned}
\tag{5.1}
$$

subject to

$$0.3 \leq x_1 \leq 1.0, \quad 0.3 \leq x_2 \leq 1.0.$$

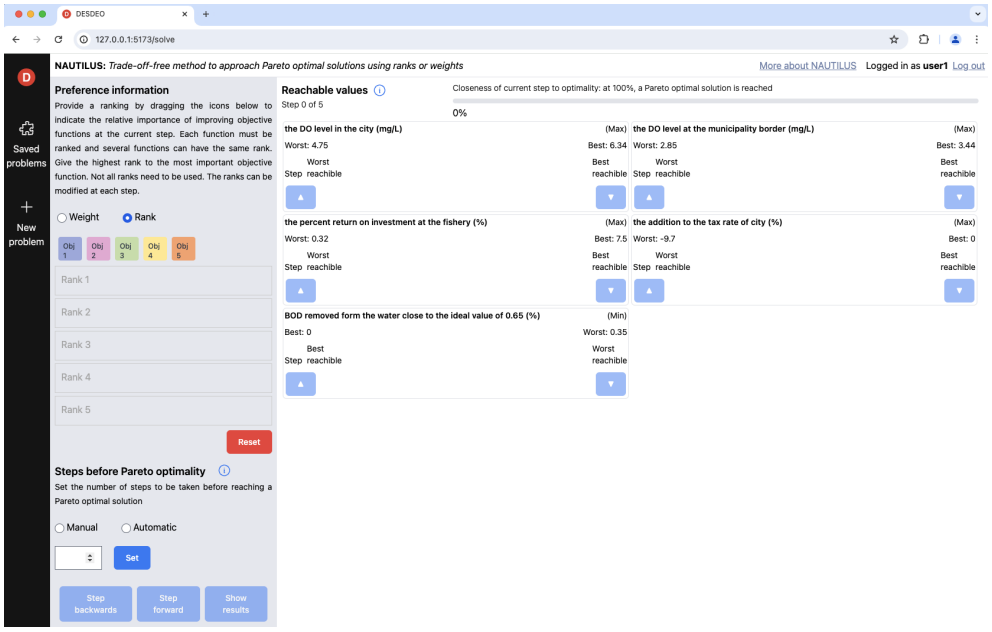

Figure 13. The starting point of the optimisation process visualised

Illustrated in the problem formulation 5.1, are two decision variables in the problem, x_1 and x_2 which represent the respective proportions of biochemical oxygen-demanding substances removed by treatment facilities from the water discharged by the fishery and the city, positioned downstream from each source. As mentioned, there are five objectives in this problem. The objectives f_1 and f_2 describe the water quality subsequent to the fishery and the city, in that order, while the objectives f_3 and f_4 relate to the percentage of return on investment at the fishery and the addition to the city tax, respectively. Objective f_5 describes the percentage of BOD removed from the water close to the ideal value 0.65. For more information about the implementation of this problem, see[1].

We can also see from the problem formulation (5.1), that the objectives from f_1 through f_4 are to be maximised, and objective f_5 is to be minimised. Both decision variables can have values ranging from 0.3 to 1.0. To begin with, the decision maker starts from the nadir point, that is $z^{nad} = (4.75, 2.85, 0.32, -9.70, 0, 35)$, and the corresponding upper bound is $z^* = (6.34, 3.44, 7.50, 0, 0)$. The NAUTILUS UI represents the ideal and nadir points as the lower and upper bounds of the reachable values to the decision maker.

1. https://desdeo-problem.readthedocs.io/en/latest/problems/river_pollution.html

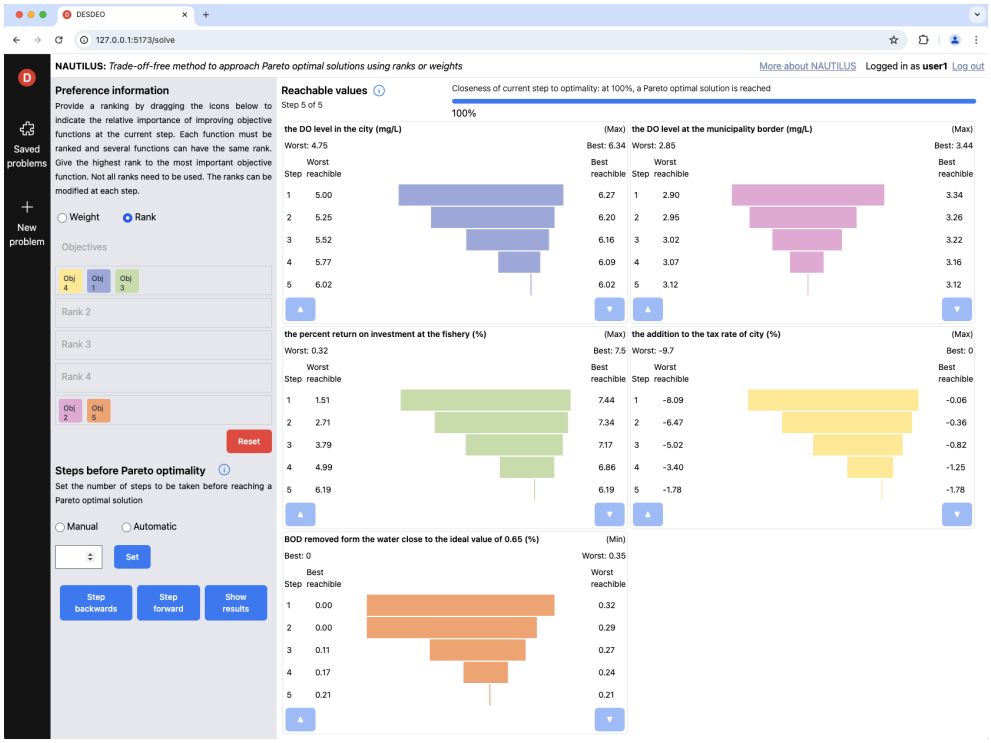

Figure 14. The optimisation process of the case study illustrated.

This starting point of the optimisation process is illustrated in Figure 13, where the NAUTILUS method is initialised and no steps have been taken yet. For the sake of this demonstration, set the number of iterations is five, representing the steps to be taken to reach a Pareto optimal solution, and the current iteration point is the nadir point.

In the first iteration, the decision maker opts to use ranking to provide preference information, and chooses to prioritise the environmental objectives f_1, f_2 and f_5 the most. Thus, the decision maker uses ranking of $(5,5,1,1,5)$.

In Figure 14, the final view of the NAUTILUS UI for the optimisation process, displaying all five iterations, is shown. We use this as a reference to explain the details of this case study's optimisation process.

As can be seen from Figure 14, after completing the first iteration, the decision maker receives the new upper bounds: $z^{1,up} = (6.27, 3.34, 7.44, -0.06, 0)$, and lower bounds $z^{1,lo} = (5.00, 2.90, 1.51, -8.09, 0.32)$, along with the distance to the Pareto optimal set $d^1 = 20$ from the current iteration point.

In the second iteration, the decision maker notices that the reachable values for all objectives have not significantly shrunk. The decision maker opts to slightly modify the ranking of the objectives to improve the return on investment percentage at the fishery. Consequently, a higher rank is assigned to f_3. Now, the ranking being $(5,5,2,1,5)$.

After taking the second step, the decision maker can visualise changes to the outcome based on the preference information used in the NAUTILUS UI. The new upper bounds are $z^{2,up} = (6.20, 3.26, 7.34, -0.36, 0)$, and the lower bounds are $z^{2,lo} = (5.25, 2.95, 2.71, -6.47, 0.29)$. The distance to the Pareto optimal set in the third iteration is $d^3 = 40$. In this iteration, the decision maker observes that the reachable values for all objectives are still relatively favourable.

In the third iteration, the decision maker chooses to use weights to express preference information. During this iteration, the decision maker assigns equal weights of 20 to f_1 and f_2, and a weight of 10 to f_5. This results to weights of $(40,40,0,0,20)$. Now the new upper are $z^{3,up} = (6.16, 3.22, 7.17, -0.82, 0.11)$ and the new lower bounds are $z^{3,lo} = (5.52, 3.02, 3.79, -5.02, 0.27)$.

After this step, the decision maker observes that for objectives f_1 , f_3 and f_4, very good values are achievable. Their respective iteration bars illustrate that the reachable values for these objectives are leaning towards the upper bound values. Since these objectives are to be maximised, this is a very good trend. In contrast, the reachable ranges for f_2 and f_5 are slightly converging towards the middle or are slightly shifted towards their respective worst reachable values. As mentioned, the decision maker wanted to focus on the environmental objectives of this problem. Hence, to improve the reachable values of f_2 and f_5, the decision maker reverts to use ranking, prioritising these two objectives above the rest. The revised ranks are $(1,5,1,1,5)$

After the third step, the new upper bounds are $z^{4,up} = (6.09, 3.16, 6.86, -1.25, 0.24)$, and the new lower bounds are $z^{4,lo} = (5.77, 3.07, 4.99, -3.40, 0.17)$. From these results, the decision maker can observe that, unfortunately, no significant improvements were made on the reachable values of f_2 and f_5, as the reachable values continued to converge relatively equally on both sides. In contrast, values of f_1, f_2 and f_3 are good. For the final iteration, the deci-

sion maker chooses to continue prioritising the improvements for f_2 and f_5, using the same preference information as used in previous iteration: $(1,5,1,1,5)$, where f_2 and f_5 have the highest rank.

At this point we have a Pareto optimal solution, and the decision maker can now evaluate it. This means that if the decision maker is satisfied with this Pareto optimal solution, they may proceed to the final solution page; alternatively, if they wish to explore more Pareto optimal solutions, they can go back some iterations to find a more preferred solution.

Objective f_1 received very good outcome. From the first step, the reachable values gradually converged towards the best achievable values. Reachable values of objectives f_2 and f_3 received good values across the optimisation process. Even though the decision maker did not pay attention to the improvements of these objective values, the nature of the problem likely contributed to these favourable outcomes. The reachable values of f_2 shrink towards the middle fairly evenly across all iterations, indicating a relatively neutral outcome for this objective. Significant changes in the reachable values of f_5 occurred in the third iteration. Due to the preference information provided after the second iteration, the reachable values drastically shifted towards the worst reachable value. Eventually, the Pareto optimal objective value of f_5 settled near the middle of the reachable values. Despite the intention to prioritise f_5, its Pareto optimal objective value ended relatively close to the middle. This outcome is not necessarily a bad one, but if the decision maker chooses to redo the optimisation process, improvements in f_5 objective function values are possible.

Figure 15 illustrates the Pareto optimal solution selected by the decision maker. The decision maker's main focus was on the environmental objectives f_1, f_2, and f_5, and the results are as follows: the dissolved oxygen concentration in the city is $6.02 mg/L$; water quality at the municipality border is measured at $3.12 mg/L$; the percent return on investment at the fishery is 6.19%. The addition to the city tax is -1.78%, which indicates a reduction rather than an increase. Essentially, this negative value means that the tax burden has been lowered. Finally, the percentage of BOD removed from the water is now 0.21%.

This case study serves as a profound demonstration of how the NAUTILUS UI is used to achieve a Pareto optimal solution without trade-offs. We have demonstrated that this imple-

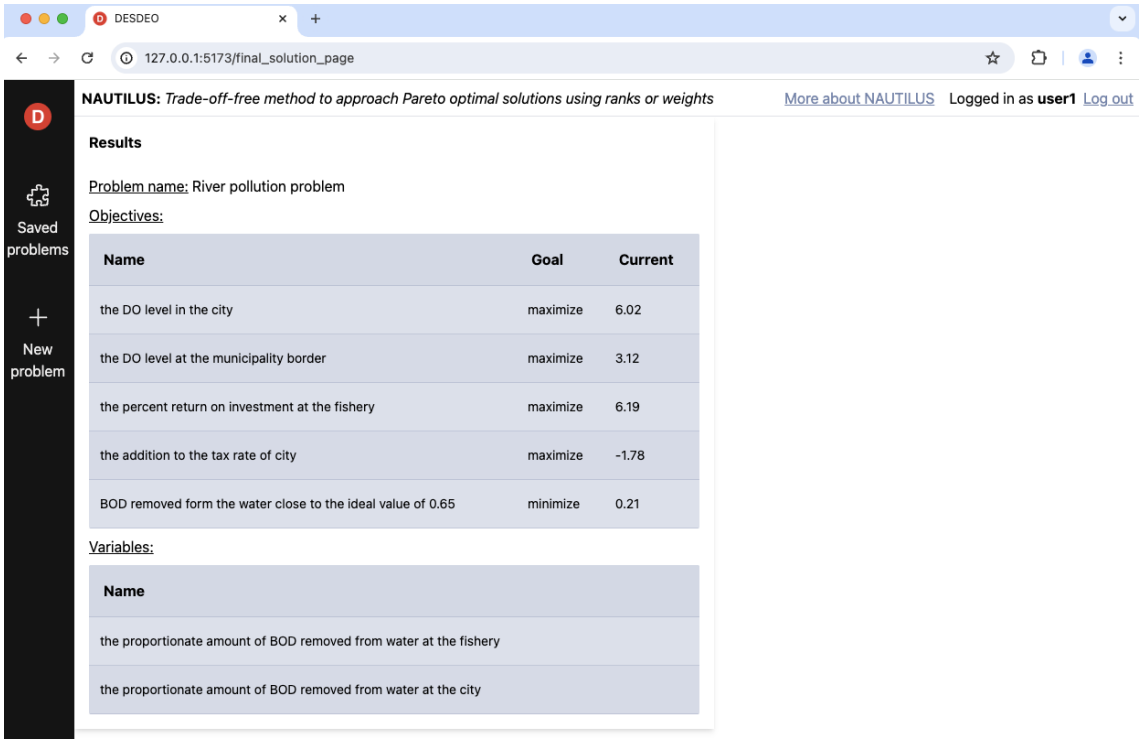

Figure 15. The results view of the selected Pareto optimal solution

mentation is capable of solving a multiobjective optimisation problem and that the NAUTILUS UI works as intended.

5.2 Preference information

According to Tarkkanen et al. (2013), there is a general need to develop intuitive and user-friendly interfaces for interactive multiobjective optimisation methods, as current user interfaces may be too complex for some decision makers. This is why, the way of providing preference information in the NAUTILUS UI has been developed to offer the ease of modification of the preference information, thereby encouraging exploration and adjustment. Engaging users through direct interaction with the interface elements keeps them involved in the task. As discussed in Chapter 2, this type of engagement is in the centre of interactive decision making methods where the the preference information directly affects the solution process.

When the decision maker opts to use ranks, they are choosing a more qualitative method

of preference expression that segments objectives into classes based on their importance for improving the current objective values. To expand what was explained in Chapter 3, ranking does not require specifying the exact extent of the difference in importance between objectives. It is particularly useful when the decision maker prefers a simpler and more intuitive method of stating preferences or when they feel unable to assign precise numerical importance to each objective. Going back to the example case at the beginning of Chapter 2, ranking allows the architect to say, for instance, that reaching a certain threshold in minimising energy consumption is more important than minimising construction costs but does not compel them to quantify how much more important it is.

The drag-and-drop functionality mimics real-world interactions, such as moving objects from one place to another. This familiar action translates well to user interfaces, making it intuitive even for users with limited experience with interacting with different kinds of user interfaces. By enabling users to "physically" manipulate objects, it reduces the cognitive load required to understand abstract concepts. In this case, how to assign objectives to ranks. As users drag objectives, they can see the effects of their actions immediately. This instant feedback helps users understand the relationship between their preferences and the possible optimisation outcomes. For instance, by assigning an objective a higher or lower rank, users sees the results of their actions in the following iteration. The drag-and-drop functionality also makes it easy to rethink preferences without entering complex commands or navigating through multiple menus. If a user makes an error or changes their mind about the priority of an objective, they can simply drag the objective to a new rank container.

Drag-and-drop can visually represent the hierarchy or priority of items, which helps in organising complex information. Users can see a clear layout of where each objective stands in relation to others, aiding in a better mental mapping of their preferences and how they relate to the optimisation process.

In contrast, assigning weights through percentages is a way that distributes a fixed number of points among the objectives, reflecting the desired level of improvement for each. This method allows for more nuanced preference information expression, giving the user the ability to fine-tune the optimisation process to reflect the precise degree of importance assigned to each objective. By assigning more points to an objective, the user is indicating a stronger

preference for improvements in that objective. This approach is preferable when the decision maker is comfortable with, or requires, a more granular control for giving preferences.

When the user distributes points, they are prompted to evaluate each objective not in isolation but in the context of the entire set of objectives. The ability to allocate a fixed pool of points (e.g. 100 points) across various objectives allows users to specify the exact degree of importance or priority for each objective in a quantitative manner.

5.3 Information exchange

Effective information exchange is critical for the interactive multiobjective optimisation methods, where the user is involved in every iteration. This challenge encompasses both the correct gathering of preference information and constructing it into a request that the NAUTILUS endpoint can process and respond to.

As previously discussed, NAUTILUS UI's preference handling considers the two ways of providing preference information, namely weights and ranks. Preference handling is implemented in such way that it allow users to input their preferences as soon as they start interacting with the UI, and to have all relevant components in one single view. In this way the preference information can be processed at the same time as the user interacts with the UI, so as soon as the user wants to take a step forward, the request is sent to the NAUTILUS endpoint. Additionally, the implementation is done in such a way that it requires no changes to the back-end.

A detailed sequence diagram, introduced in Subsection 4.2.1, illustrates the structure of how the information flow between the UI and the server is established. Initially, when a user selects the NAUTILUS method, the server initialises it, and provides ideal and nadir points. These serve as the initial bounds for the Pareto optimal set, guiding the initial and subsequent iterations as well as a way to tell the user what can be achieved in future iterations. As the user interacts with the UI, they can adjust their preferences and decide on the number of remaining iterations. The NAUTILUS UI allows for such adjustments to be made on-the-fly without necessitating additional server requests. Each user input is converted into a request sent to the NAUTILUS endpoint, which then processes these inputs to provide new bounds,

the distance to the Pareto optimal set and the current iteration point. Throughout the iterative process, users can go back to previous iterations to adjust their preferences if the results are not satisfactory. This capability supports the decision-making process, as it allows users to easily explore different Pareto optimal solutions.

5.4 Expertise and experience level

According to Hakanen, Miettinen, and Matković (2021), implementations for interactive multiobjective optimisation methods must be responsive. A decision maker's action should prompt a visible change in the user interface without significant delay, and actions that the decision maker takes should result in a predictable outcome. By maintaining this level of predictability and responsiveness, the UI helps to reinforce the user's understanding of how their actions impact the preferred solution, leading to more efficient and confident decision-making.

The NAUTILUS UI has multiple elements that guide the user. These elements are information text boxes that give the initial instructions of what are the available ways to provide preference information and how to give those preferences. The UI also has information texts that tell how to change the number of steps to take to reach a Pareto optimal solution. The presence of informational text boxes and instructions within the NAUTILUS UI plays an important role in catering to different user levels. For beginners, these guides are crucial for understanding how to input preferences and adjust settings like the number of steps to a Pareto optimal solution. Such features reduce dependency on external help and increase user autonomy. For advanced users, these elements can serve as quick references or reminders, speeding up the interaction process by reducing the need to recall information from memory.

The requirement for the UI to be responsive and for actions to yield predictable outcomes is crucial in supporting users of varying expertise levels. For novice users, the immediate feedback and consistency help in learning and understanding the UI's operation without overwhelming them. For experienced users, the predictability allows for quick, efficient navigation and manipulation of the UI, enabling them to leverage their prior knowledge and expectations effectively.

5.5 Fit within DESDEO

In evaluating the integration of the NAUTILUS UI within the DESDEO framework, the principles articulated by Wang and Wang (2010) and Prat, Comyn-Wattiau, and Akoka (2015) are particularly relevant. Wang and Wang (2010) emphasise the importance of building upon existing artefacts, which ensures that new solutions are grounded in the latest technological advances. This approach leverages current insights and technologies and contributes to the continuous evolution of technology and methodology. Prat, Comyn-Wattiau, and Akoka (2015) reinforce this by describing it as the harnessing of recent technologies, a principle that aligns well with DESDEO's aim to integrate and enhance multiobjective optimisation methods.

DESDEO's modular architecture and its commitment to open-source, ready-to-use implementations encourages the addition of new functionalities, such as the NAUTILUS UI, and supports their integration. This modularity allows for the user interface to be easily incorporated without disrupting existing system components, ensuring that it enhances rather than complicates the integration process.

Furthermore, the flexibility of DESDEO to accommodate various interactive methods without requiring major modifications to existing code base underscores its suitability for integrating the NAUTILUS UI. Also, the NAUTILUS UI includes components that are designed to be usable with both new and existing modules within the DESDEO framework. By successfully fitting within DESDEO's structure, the NAUTILUS UI meets the current need of user interface for the NAUTILUS method and also remains adaptable to future advancements and changes within DESDEO as well as in the field. This alignment ensures that the UI effectively bridges the gap between theoretical methods and practical applications, fulfilling DESDEO's primary goal.

6 Discussions

This chapter summarises the thesis. It begins with an overview of the work completed throughout this project. Following this, we highlight the key achievements of the research. Next, we identify and discuss areas that could benefit from further improvement. Finally, we explore potential avenues for future research.

6.1 Summary of work

Throughout this project, several tasks were continually executed from start to finish. The types of work listed below are not in any specific order:

1. Requirements gathering for the NAUTILUS UI.
2. Familiarising myself with the coding languages used in DESDEO.
3. Refining the design of the UI. This was an was ongoing task. Although the design was predetermined, I had the opportunity to evaluate and modify it, making some independent decisions along the way.
4. Setting up the development environment, and familiarising myself with the back-end architecture as well as the coding languages used in DESDEO.
5. Development work of the NAUTILUS UI.
6. Writing this thesis as a report of the work done.

6.2 Highlights of the project

I am particularly satisfied with the decisions made on the design and the development of the UI. Focusing on usability and intuitiveness contributed positively to the project's success. By making the interface intuitive for giving preference information and visualising the optimisation process in a clear and concise manner, I was able to ensure that the user experience is straightforward and accessible for newcomers, yet responsive enough for experienced users. This approach effectively meets the diverse needs of its audience.

The choice to adopt a modular approach in developing the UI using the same guidelines

that already exist in DESDEO naturally was very beneficial. This design decision helps in integrating the NAUTILUS UI seamlessly within DESDEO, and also enhances the flexibility and scalability of DESDEO. It allowed me to utilise components already implemented within DESDEO, and it allows components from NAUTILUS UI to be reused or adapted for different purposes within the framework.

Given that there was essentially no baseline implementation of any kind of a user interface for the NAUTILUS method in DESDEO before this project, I am very satisfied with how the final version fits visually and functionally within DESDEO. Additionally, starting from scratch and building up to a complete, functional UI provided me with substantial learning opportunities and contributed significantly to my professional growth. The iterative process of development, feedback, and revision helped me in refining development skills, which is critical for my personal and this project's success.

6.3 Areas for improvement

One aspect that could have been handled differently was my personal scheduling of the project. Along the way, I realised that with more careful planning and scheduling, I could have been more efficient in my writing and development work. More strategic time management would have potentially accelerated certain phases of the project and alleviated some of the pressures encountered during deadlines, especially towards the project's final stages.

Another aspect that could have been managed differently was my decision to use the older NAUTILUS endpoint. At the project's outset, I utilised only one endpoint for the NAUTILUS UI. However, during the project, a new endpoint was developed and became available for use. Despite this, due to time constraints and the previously mentioned scheduling challenges, I opted not to switch to the new endpoint. In retrospect, adopting the new endpoint might have simplified several aspects of the development. For instance, the old endpoint required manual handling of the objectives that needed to be maximised; the user interface had to convert these to be minimised before sending requests to the endpoint, and do the conversion again after receiving responses. The new endpoint, conversely, eliminates this need. Additionally, switching to the new endpoint would have allowed the NAUTILUS UI to integrate more

seamlessly with DESDEO, as it would already be utilising the new endpoint.

Another aspect is that I recognised that I could have conducted a more thorough comparison between the NAUTILUS method used in this thesis and other members of the NAUTILUS family. A deeper analysis of the limitations of the NAUTILUS method used here, in contrast to the capabilities and enhancements offered by other methods within the NAUTILUS family, would have provided a more comprehensive understanding of its relative strengths and weaknesses. This comparative analysis could have brought potential adjustments to the NAUTILUS UI I implemented, leading to improved functionality and user experience.

6.4 Future research

The first area for possible future research involves enhancing the navigation capabilities within the developed NAUTILUS UI, particularly addressing the current inability to return to the NAUTILUS UI view from the results view. This would facilitate repeated optimisation cycles on the same problem if the user so desires. Enabling multiple rounds of optimisation over the same problem enables the user in exploring a variety of Pareto optimal solutions, and also helps them to understand the implications of the provided preference information on the results. Future modifications could introduce a functionality that allows users to navigate back to the main view of the user interface after viewing results. This enhancement would support a more exploratory approach, enabling users to refine their preferences based on previous results and potentially discover more favourable solutions through successive optimisations.

A second area for future research involves implementing a way to display the current iteration point in the NAUTILUS UI. According to Miettinen et al. (2010), this point can be shown to the decision maker if desired. Currently, the NAUTILUS UI does not support displaying this point explicitly, even if the user wishes to see it. Current iteration point This point could be very helpful to the decision maker, as it represents the worst values for each iteration bar. For example, the current iteration point could be highlighted within the iteration bar.

The next potential area for future research could the reconsideration of how users provide ranks as preference information. During the implementation of the NAUTILUS UI, I ob-

served a potentially unintuitive aspect of providing preference information as ranks that could confuse some users. In my opinion, the ranking might not be immediately clear to all, because some users may naturally associate a lower rank number with higher priority, similar to how rankings are typically understood in other contexts where 'number one' is considered the best. This misunderstanding could stem from the intuitive interpretation of numbers themselves, where lower numbers are often perceived as superior. Although this is a very small detail and the ranking is explained in the information text, I think future versions of the NAUTILUS UI might benefit from an alternative presentation of the ranks themselves, for example changing the order of the rank containers to start from the highest rank. This could align more closely with common numerical perceptions that some might perceive.

The next area for future research would be the considerations of the limitations with the current endpoint that NAUTILUS uses. This endpoint made me do certain design choices. For example, after initialisation, only one endpoint is used to fetch responses for all iterations. A potential improvement could involve implementing separate endpoints for the step-back functionality, and handling ranks and weights preference information. This would allow for each type of preference information to be processed more effectively, without needing to adapt them to fit the same request format.

Another aspect related to the future research of the endpoint is the absence of units for the objectives. Providing units is crucial for decision makers to fully grasp the solutions and understand the problem as a whole. Including this information can significantly aid in the interpretation and application of the results. Units would be particularly useful in the results view, providing context for the numerical values presented. To clarify, I discuss these aspects as areas for improvement for the current endpoint that the NAUTILUS UI uses. As I mentioned in the previous subsection, a new endpoint is under development, and if these aspects have not yet been considered, they could prove very useful.

Finally, according to Ruiz et al. (2015), the NAUTILUS method incurs a substantial computational costs. Consequently, using the NAUTILUS method becomes impractical for multiobjective optimisation problems with computationally intensive objective and constraint functions, as it subjects the decision maker to significant waiting times. A direct continuation

of the NAUTILUS UI developed in this thesis, would be the implementation of NAUTILUS 2, presented by Miettinen et al. (2015). The user interface components presented in this thesis can be readily applied to the development of the NAUTILUS 2 user interface. Given the limitations discussed above, it would be logical for the development of NAUTILUS UI to transition to NAUTILUS 2.

7 Conclusions

In this thesis, I explored ways to successfully implement the user interface for the NAUTILUS method within the DESDEO software framework. Throughout the development of the user interface, numerous challenges were addressed, including structuring and managing the exchange of information between the decision maker and the user interface, as well as between the user interface and the NAUTILUS endpoint.

The user interface was developed to cater to a broad range of decision makers, from those unfamiliar with interactive multiobjective optimisation methods to experienced ones who regularly engage with multiobjective optimisation. The user interface was constructed to be intuitive and easy to follow, ensuring that decision makers can efficiently provide their preference information and navigate through the optimisation process. This setup allows decision makers to gain insights into the problem and explore various Pareto optimal solutions in search of the most preferred one. During the development of the user interface, I consistently focused on the modular structure and development principles used in the DESDEO software framework. By achieving this alignment, I was able to ensure that the user interface would integrate as seamlessly as possible within the DESDEO software framework.

To ensure that the requirements mentioned above were met, a comprehensive evaluation of the user interface was conducted. The constructed user interface was tested to demonstrate that it indeed meets the criteria. Looking at the finished work, I think it is safe to say that the user interface of the NAUTILUS method has been successfully integrated within the DESDEO software framework. With the suggested areas for improvement and the proposed future research, once implemented, the user interface will become a valuable tool for the multiobjective optimisation community to utilise.

Bibliography

Hakanen, Jussi, Kaisa Miettinen, and Krešimir Matković. 2021. “Task-based Visual Analytics for Interactive Multiobjective Optimization”. *Journal of the Operational Research Society* 72:2073–2090.

Hevner, Alan, and Samir Chatterjee. 2010. “Design Science Research in Information Systems”. In *Design Research in Information Systems: Theory and Practice,* 9–22. Springer.

Hwang, Ching-Lai, and Abu Syed Md. Masud. 1979. *Multiple Objective Decision Making — Methods and Applications: A State-of-the-Art Survey.* Springer.

Jazayeri, Mehdi. 2007. “Some Trends in Web Application Development”. In *Future of Software Engineering (FOSE’07),* 199–213. IEEE.

Kahneman, Daniel, and Amos Tversky. 1979. “Prospect Theory: An Analysis of Decision Under Risk”. In *Handbook of the Fundamentals of Financial Decision Making: Part I,* 99–127. World Scientific.

Miettinen, Kaisa. 1994. “On the Methodology of Multiobjective Optimization with Applications”. PhD thesis, University of Jyväskylä.

———. 1999. *Nonlinear Multiobjective Optimization.* Springer Kluwer Academic Publishers.

Miettinen, Kaisa, Petri Eskelinen, Francisco Ruiz, and Mariano Luque. 2010. “NAUTILUS Method: An Interactive Technique in Multiobjective Optimization Based on the Nadir point”. *European Journal of Operational Research* 206:426–434.

Miettinen, Kaisa, and Marko Mäkelä. 1997. “Interactive Method NIMBUS for Nondifferentiable Multiobjective Optimization Problems”. In *Multicriteria Analysis,* edited by João Clímaco, 310–319. Springer.

———. 2006. “Synchronous Approach in Interactive Multiobjective Optimization”. *European Journal of Operational Research* 170:909–922.

Miettinen, Kaisa, Dmitry Podkopaev, Francisco Ruiz, and Mariano Luque. 2015. “A New Preference Handling Technique for Interactive Multiobjective Optimization without Trading-off”. *Journal of Global Optimization* 63:633–652.

Miettinen, Kaisa, and Francisco Ruiz. 2016. “NAUTILUS Framework: Towards Trade-off-free Interaction in Multiobjective Optimization”. *Journal of Business Economics* 86:5–21.

Miettinen, Kaisa, Francisco Ruiz, and Andrzej P Wierzbicki. 2008. “Introduction to Multiobjective Optimization: Interactive Approaches”. In *Multiobjective Optimization: Interactive and Evolutionary Approaches,* edited by Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Słowiński, 27–57. Springer.

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

Narula, Subhash C, and H Roland Weistroffer. 1989. “A Flexible Method for Nonlinear Multicriteria Decision-making Problems”. *IEEE Transactions on Systems, Man, and Cyberenetics* 19:883–887.

Ojalehto, Vesa, and Kaisa Miettinen. 2019. “DESDEO: An Open Framework for Interactive Multiobjective Optimization”. In *Multiple Criteria Decision Making and Aiding: Cases on Models and Methods with Computer Implementations,* 67–94. Springer.

Peffers, Ken, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. 2007. “A Design Science Research Methodology for Information Systems Research”. *Journal of Management Information Systems* 24:45–77.

Prat, Nicolas, Isabelle Comyn-Wattiau, and Jacky Akoka. 2015. “A Taxonomy of Evaluation Methods for Information Systems Artifacts”. *Journal Of Management Information Systems* 32:229–267.

Ruiz, Ana, Francisco Ruiz, Kaisa Miettinen, Laura Delgado-Antequera, and Vesa Ojalehto. 2019. “NAUTILUS Navigator: Free Search Interactive Multiobjective Optimization Without Trading-off”. *Journal of Global Optimization* 74:213–231.

Ruiz, Ana, Karthik Sindhya, Kaisa Miettinen, Francisco Ruiz, and Mariano Luque. 2015. "E-NAUTILUS: A Decision Support System for Complex Multiobjective Optimization Problems Based on The NAUTILUS Method". *European Journal of Operational Research* 246:218–231.

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

Smedberg, Henrik, and Sunith Bandaru. 2023. "Interactive Knowledge Discovery and Knowledge Visualization for Decision Support in Multi-objective Optimization". *European Journal of Operational Research* 306:1311–1329.

Tarkkanen, Suvi, Kaisa Miettinen, Jussi Hakanen, and Hannakaisa Isomäki. 2013. "Incremental User-interface Development for Interactive Multiobjective Optimization". *Expert Systems with Applications* 40:3220–3232.

Venable, John, Jan Pries-Heje, and Richard Baskerville. 2012. "A Comprehensive Framework for Evaluation in Design Science Research". In *Design Science Research in Information Systems. Advances in Theory and Practice,* edited by Ken Peffers, Marcus Rothenberger, and Bill Kuechler, 423–438. Springer.

———. 2016. "FEDS: a Framework for Evaluation in Design Science Research". *European Journal Of Information Systems* 25:77–89.

Wang, Shouhong, and Hai Wang. 2010. "Towards Innovative Design Research in Information Systems". *Journal of Computer Information Systems* 51:11–18.

Xin, Bin, 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* 6:41256–41279.

Appendices

A Use of AI

In this thesis, AI tools are used to enhance the academic language by finding synonyms and refining sentence structures, as well as to aid in software development by understanding syntax and creating basic components.