

Matias Nieminen

Navigation Methods in Multiobjective Optimization

Master's Thesis in Information Technology

June 5, 2024

University of Jyväskylä

Faculty of Information Technology

Author: Matias Nieminen

Contact information: mnieminen98@gmail.com

Supervisors: Kaisa Miettinen, and Giovanni Misitano

Title: Navigation Methods in Multiobjective Optimization

Työn nimi: Monitavoiteoptimoinnin navigointimenetelmät

Project: Master's Thesis

Study line: Mathematical Modelling in Science and Decision Analytics

Page count: 80+0

Abstract: Multiobjective optimization is the minimization or maximization of multiple conflicting objective functions. To make a distinction between multiple solutions to a multiobjective optimization problem, often a decision maker (DM), a domain expert, is needed to provide their preferences. Navigation methods for multiobjective optimization generate and visualize solutions in real-time, and allow a DM to control the solution process by changing their preferences during the process. In this thesis, a literature review was conducted to establish an understanding of the state of the art in navigation methods for multiobjective optimization problems. Based on the results of the review, a synthesis was formed to find out what kind of navigation methods exist, what features they have, and what, if anything, they have in common. The found navigation methods were analyzed and compared in terms of the type of solutions (i.e., Pareto optimal or approximated solutions) the DM sees during the solution process, whether and how they fulfill the desirable properties for navigation methods, defined in the literature. Some common features related to the implementations of the methods and their graphical user interfaces were found. Based on the synthesis, one can learn about the methods, and the synthesis may aid the DM in choosing a method when solving a multiobjective optimization problem. One may also develop a new optimization method by combining the different features of the methods found in the synthesis.

Keywords: Multiobjective optimization, Navigation methods, Interactive methods, Decision-making

Suomenkielinen tiivistelmä: Monitavoiteoptimointi on useiden ristiriitaisten tavoitefunktioiden minimointia tai maksimointia. Useiden monitavoiteoptimointiongelman ratkaisujen vertailemiseen tarvitaan usein päätöksentekijä, joka on usein sovellusalan asiantuntija, antamaan mieltymystietoja. Monitavoiteoptimointiin tarkoitettavat navigointimenetelmät tuottavat ja visualisoivat ratkaisuja reaaliajassa sekä mahdollistavat päätöksentekijälle ratkaisuprosessin kontrolloinnin muuttamalla mieltymystietojaan. Tässä tutkielmassa tehtiin kirjallisuuskatsaus, jonka tavoitteena oli selvittää navigointimenetelmien nykytilanne. Katsauksen tulosten perusteella muodostettiin synteesi, jonka tavoitteena oli selvittää, minkälaisia navigointimenetelmiä kirjallisuudesta löytyy, mitä ominaisuuksia niillä on, ja mitä yhteistä niillä on. Löydettyjä menetelmiä analysoitiin ja vertailtiin sen suhteen, millaisia ratkaisuja (Pareto-optimaalisia vai approksimoituja ratkaisuja) päätöksentekijä näkee ratkaisuprosessin aikana ja täyttävätkö ne kirjallisuudessa määritellyt navigointimenetelmien toivotut ominaisuudet, sekä miten ne täyttävät nämä ominaisuudet. Menetelmiltä löytyi yhdistäviä tekijöitä liittyen menetelmien sekä niiden graafisten käyttöliittymien toteutuksiin. Synteessin perusteella voi oppia olemassa olevista navigointimenetelmistä, ja synteesi voi auttaa päätöksentekijää valitsemaan ratkaisumenetelmän monitavoiteoptimointiongelman ratkaisemiseksi. Yhdistelemällä menetelmien eri ominaisuuksia, jotka tunnistettiin synteessissä, voidaan kehittää uusia navigointimenetelmiä.

Avainsanat: Monitavoiteoptimointi, Navigointimenetelmät, Interaktiiviset menetelmät, Päätöksenteko

List of Figures

Figure 1. The graphical user interface used in Pareto Race. Taken from Korhonen and Wallenius (1988).	17
Figure 2. The graphical user interface used in MIRA navigator. Taken from Thieke et al. (2007).	19
Figure 3. The graphical user interface used in multiple Pareto surface navigation. Taken from Craft and Monz (2010).	21
Figure 4. A graphical user interface for Pareto Navigator. Taken from Tarkkanen et al. (2013).	23
Figure 5. Illustration of the use of objective and restriction sliders in Bortz et al.'s (2014) method. Taken from Bortz et al. (2014).	24
Figure 6. The graphical user interface used in Lin and Ehrgott's (2018) method. Taken from Lin and Ehrgott (2018).	26
Figure 7. The graphical user interface for Nonconvex Pareto Navigator. Taken from Hartikainen, Miettinen, and Klamroth (2019).	28
Figure 8. The graphical user interface used in patch navigation. Taken from Collicott et al. (2021).	29
Figure 9. The iSOM visualization used in iSOM-Pareto Race. Taken from Yadav, Ramu, and Deb (2022).	31
Figure 10. The color codes used by the iSOM visualization. Taken from Yadav, Ramu, and Deb (2022).	31
Figure 11. An example of input navigation in RINADA. Taken from Baldan et al. (2023).	33
Figure 12. The graphical user interface of NAUTILUS Navigator. Taken from Ruiz et al. (2019).	35
Figure 13. The navigation view of O-NAUTILUS. Taken from Saini et al. (2022).	36

List of Tables

Table 1. Databases used in the literature review and the corresponding search queries used in the databases.	12
Table 2. How the methods fulfill the "navigation is complete" property.	39
Table 3. How the methods fulfill the "navigation is computationally efficient" property.	42
Table 4. How the methods fulfill the "construction of the navigation set is computationally efficient" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.	45
Table 5. How the methods fulfill the "accuracy of the navigation set can be measured" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.	48

Table 6. How the methods fulfill the "accuracy of the navigation set can be improved" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.	51
Table 7. How the methods fulfill the "the DM can control the navigation" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.	55
Table 8. How the methods fulfill the "low cognitive load is set on the DM" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.	58
Table 9. How the methods fulfill the "the DM is allowed to learn" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.	61
Table 10. How the methods fulfill the "the DM can get additional information of the navigation set" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.	64

Contents

1	INTRODUCTION	1
2	BACKGROUND	4
3	LITERATURE REVIEW	11
4	SYNTHESIS	15
4.1	Classification	15
4.1.1	Pareto front	15
4.1.2	Approximation of the Pareto front	17
4.1.3	NAUTILUS methods	33
4.2	Desirable properties	36
4.2.1	Technical properties	37
4.2.2	Properties related to user experience	52
5	DISCUSSION	65
6	CONCLUSIONS	68
	BIBLIOGRAPHY	70

1 Introduction

Many real-world problems have multiple, often conflicting, objective functions that are to be optimized simultaneously. These are called multiobjective optimization problems. Because the objective functions are conflicting, all of them cannot reach their optimal values at the same time. Therefore, for multiobjective optimization problems, there often does not exist one definite optimal solution. So, instead of just one optimal solution, a set of so-called Pareto optimal solutions exists. Moving from one Pareto optimal solution to another means that while at least one objective function's value improves, at least some other objective function value gets worse. Because of these trade-offs, a decision maker (DM) is needed to make the distinction between the solutions. The DM is a domain expert who has knowledge of the problem. The DM is able to give their preference information which is used to find the most preferred solution among the Pareto optimal ones.

Many methods have been developed to solve multiobjective optimization problems (Miettinen 1999). These methods can be classified according to the role of the DM in the solution process (Miettinen 1999). If there is no DM providing preferences, a no-preference method has to be used. In a posteriori methods, the DM provides their preferences after a representative set of Pareto optimal solutions is generated. A posteriori methods are especially useful when the DM cannot provide or does not know their preferences beforehand. In a priori methods, the DM provides their preferences before the solution process. A priori methods are useful for when there is not much time for the solution process. In interactive methods, the DM is actively involved in the solution process. During the solution process, the DM may iteratively change their preferences, which in turn allows them to explore different solutions to find the one that they prefer. The benefits of interactive methods include that the DM is able to learn about the problem and the feasible solutions. The DM can change their preferences according to what they learn about the solutions and the trade-offs between the objective functions.

This thesis focuses on navigation methods, a subclass of interactive methods. Navigation methods show the DM the changing objective function values in real-time, which allows the DM to learn about the problem and find areas that they find interesting (Hartikainen,

Miettinen, and Klamroth 2019). The benefit of navigation methods in comparison to other interactive methods comes from the fact that the DM sees new solutions in real-time as they explore a set of solutions, called a navigation set. The DM gives their preference information which is then used to compute solutions in real-time for the DM to see. This preference information could be, for example, values for the objective functions, that would be satisfactory to the DM, called aspiration levels. The DM is shown how the objective function values evolve during the navigation process, which helps the DM in learning about the problem and the possible solutions they can find. If the DM wants to look for solutions elsewhere, they can change their preferences based on what they have learned during the navigation process.

To be able to make the navigation process smooth and support the DM's learning, navigation methods need a graphical user interface. The graphical user interface shows the DM the changing objective function values and allows the DM to control the navigation. The objective function values should be displayed in an understandable way so that the DM has as low of a cognitive load as possible. To also lower the cognitive load set on the DM, the interface should make comparing different solutions easier so that the DM does not have to remember all the previously seen solutions.

Many different navigation methods for multiobjective optimization have been introduced in the literature (for example, Korhonen and Wallenius (1988), Monz et al. (2008) and Saini et al. (2022)). However, according to the best of our knowledge, this thesis presents the most extensive literature review conducted on navigation methods to date. Navigation methods are an interesting research subject because of their benefits compared to other interactive methods, which include that the DM sees solutions in real-time during the decision-making process. Therefore, the literature review aims to provide knowledge about the state of the art in navigation methods. The results of the literature review were gained by conducting a search in four different databases: Scopus, Web of Science Core Collection, ACM Digital Library's The ACM Guide to Computing Literature, and IEEE Xplore.

Based on the results of the literature review, a synthesis is formed. In the synthesis, the existing navigation methods are compared to analyze their similarities, their differences, and their features. Hartikainen, Miettinen, and Klamroth (2019) introduced a set of desirable properties, that navigation methods should fulfill. The methods are analyzed and compared

in terms of the type of a navigation set the navigation takes place in, whether the methods fulfill the desirable properties, and how they fulfill the properties.

This thesis has two main contributions. Firstly, the literature review gives an idea of what kind of navigation methods exist in the literature. Secondly, the synthesis formed based on the results of the literature review gives more details on the existing methods and how they compare to each other. Based on the synthesis, one can learn about the methods, and the synthesis may aid in choosing a method to solve a problem. One may also develop a new optimization method by combining the different features of the methods, that are identified in the synthesis.

The structure of the thesis is as follows. In Chapter 2, some background concepts are introduced and defined. The literature review process is described in detail in Chapter 3. The synthesis is formed based on the results of the literature review in Chapter 4. Chapter 5 consists of a discussion about navigation methods and ideas for future research. Finally, in Chapter 6, the thesis is concluded.

The contributions of this thesis may be used as a basis for future studies related to navigation methods. The literature review provides an overview of existing navigation methods to be considered in future reviews and mapping studies. The synthesis could be used as a foundation to prepare new syntheses with different perspectives.

2 Background

The problems in multiobjective optimization may be defined as follows:

$$\begin{aligned} \min \quad & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})\} \\ \text{subject to} \quad & \mathbf{x} \in S, \end{aligned} \tag{2.1}$$

where f_1, f_2, \dots, f_k are the *objective functions* that will henceforth be called *objectives* for short, k is the number of objectives and the dimension of the *objective space* \mathbb{R}^k , and $f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}))$ is an *objective vector* that consists of *objective values*. An n -dimensional *decision vector* $\mathbf{x} = (x_1, x_2, \dots, x_n)$ consists of *decision variables* x_1, x_2, \dots, x_n , and $S \subset \mathbb{R}^n$ is a *feasible set*, which is defined by *constraint functions*. Constraint functions set restrictions on the values of the decision variables in the *decision space* \mathbb{R}^n .

For multiobjective optimization problems with continuous decision variables, there can be an infinite number of feasible solutions to the problem, and because the objectives are usually conflicting, it is impossible to find one optimal solution to the problem so that all of the objectives reach their optimal value at the same time. Therefore, instead of just one optimal solution, there exist multiple *Pareto optimal solutions*. Pareto optimal solutions are feasible solutions where none of the objective values, corresponding to the solution, can be improved without worsening the value of at least one other objective. Pareto optimal solutions can also be defined as *nondominated solutions*. Assuming all the objectives are to be minimized, a solution $\mathbf{x}^1 \in S$ is said to *dominate* a solution $\mathbf{x}^2 \in S$, if $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2)$ for all $i = 1, \dots, k$ and $f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$ for at least one $j \in \{1, \dots, k\}$. For any objective that is to be maximized, the inequalities would be reversed. If there does not exist a solution $\mathbf{x} \in S$ that dominates \mathbf{x}^1 , then \mathbf{x}^1 is said to be Pareto optimal (PO). A *Pareto optimal set* is the set of all Pareto optimal solutions and the image of the Pareto optimal set in the objective space is called a *Pareto optimal front* or a *Pareto front* for short.

Two points that are often used in multiobjective optimization to describe the Pareto front are the so-called *ideal* and *nadir* points. An ideal point is a point that has as its components the best possible values each objective can have in the Pareto optimal set. A nadir point is the exact opposite, it has as its components the worst possible values of each objective

in the Pareto optimal set. The ideal point can be computed by optimizing each objective separately without the need to know the Pareto optimal set, whereas to find the exact nadir point, the Pareto optimal set is needed. The nadir point is, in practice, more difficult to compute because the Pareto optimal set is often not known. Therefore, the nadir point may have to be estimated by, for example, using a payoff table (Miettinen 1999). The ideal and nadir points are denoted as follows:

$$z^{ideal} = \left(\min_{\mathbf{x} \in S \text{ is PO}} f_1(\mathbf{x}), \min_{\mathbf{x} \in S \text{ is PO}} f_2(\mathbf{x}), \dots, \min_{\mathbf{x} \in S \text{ is PO}} f_k(\mathbf{x}) \right)$$

and

$$z^{nadir} = \left(\max_{\mathbf{x} \in S \text{ is PO}} f_1(\mathbf{x}), \max_{\mathbf{x} \in S \text{ is PO}} f_2(\mathbf{x}), \dots, \max_{\mathbf{x} \in S \text{ is PO}} f_k(\mathbf{x}) \right),$$

where all the objectives are assumed to be minimized. If an objective f_i , where $i \in \{1, \dots, k\}$, was to be maximized, the corresponding component of the ideal point would be $z_i^{ideal} = \max_{\mathbf{x} \in S \text{ is PO}} f_i(\mathbf{x})$, where $\mathbf{x} \in S$ is Pareto optimal, and the corresponding nadir component would be $z_i^{nadir} = \min_{\mathbf{x} \in S \text{ is PO}} f_i(\mathbf{x})$, where $\mathbf{x} \in S$ is Pareto optimal.

Mathematically it is not possible to determine the "best" solution among the Pareto optimal solutions, so to find that solution, a *decision maker's* (DM) preference information is needed because the "best" solution is subjective and depends on the DM's preferences. The DM is usually a domain expert who has knowledge of the problem and who can give preferences between different solutions. The solution that is reached according to the DM's preferences is called the *most preferred solution*. To reach this solution, there are many ways the DM can give their preference information. The DM may, for example, give aspiration levels to the objective values i.e., some values the DM would like to reach (Miettinen 1999). The DM may also give a classification of the objectives in which the DM chooses which objectives they would like to see improve, which they allow to impair, and which should stay the same (Miettinen 1999). Other types of preference information include choosing one among a set of Pareto optimal solutions, and marginal rates of substitutions in which the DM chooses how much they are willing to allow one objective value to impair in order to improve the value of one other while the other objective values stay the same (Luque, Ruiz, and Miettinen 2011).

Based on the DM's preference information, a *preference model* can be built (Figueira et al. 2008). Forming and usage of the preference model is beneficial because as said by

Figueira et al. (2008), "On one hand, the preference information provided by the DM contributes to the construction of a preference model and, on the other hand, the use of the preference model shapes the DM's preferences or, at least, makes the DM's conviction evolve."

As mentioned in the introduction, methods for solving multiobjective optimization problems can be divided into four classes based on the DM's role in the solution process (Miettinen 1999). There is assumed to be no preference information available from a DM in *no-preference methods*, therefore some method is used to reach some solution which is then shown to the DM who can either accept the solution or reject it. In *a posteriori methods*, a representative set of Pareto optimal solutions is generated and shown to the DM who then chooses among them the most preferred one. In *a priori methods*, the DM is asked to give preference information before the solution process and then a solution is found based on the information. *Interactive methods* have an iterative process to help the DM learn about the problem, explore the solutions, and ultimately find the most preferred solution. One of the benefits of interactive methods compared to others is the possibility of learning for the DM. After each iteration, the DM is shown, for example, the ideal and nadir points and the objective values corresponding to the *current solution(s)* i.e., the objective values for the most recent solution(s) computed, to help them learn about the problem. To find more satisfactory solutions, the DM can then change their preferences for the next iteration which are then used to produce more solutions. This allows the DM to learn about the problem and the effects their preference information and changing it has on the solutions found.

As mentioned, one subclass of interactive methods is *navigation methods*. The idea of navigation methods is to dynamically generate solutions and allow the DM to see the changing objective values by visualizing them, which lets the DM see the effects of their preferences in real-time so that they can modify them if necessary. In contrast to other interactive methods where the DM gets to see new solutions after each iteration, in navigation methods the DM sees new solutions continuously during the solution process i.e., each iteration. In interactive solution processes, two phases can often be identified: a *learning phase* and a *decision phase* (Miettinen, Ruiz, and Wierzbicki 2008). In the learning phase, the DM learns about the problem, the trade-offs between the objectives and the feasible solutions to the problem, and eventually identifies a region of interest. In the decision phase, the DM finds the most

preferred solution based on what they learned in the learning phase, that is, by fine-tuning the search in the region of interest. Navigation methods support especially the learning phase (Ruiz et al. 2019).

Allmendinger et al. (2017) define *navigation* as “the interactive procedure of traversing through a set of points (the navigation set) in the objective space guided by a decision maker (DM). The ultimate goal of this procedure is to identify the single most preferred Pareto optimal solution.” To reach this ultimate goal via navigation, many different navigation methods have been developed. Though the concepts of navigation and navigation methods had not yet been defined, navigation methods have existed since the work by Korhonen and Wallenius (1988).

Hartikainen, Miettinen, and Klamroth (2019) defined a modular structure for navigation methods. They named these modules as

- *navigation set*: the set in the objective space in which the navigation takes place, that can either be the Pareto front, or a representation or an approximation of it,
- *navigation control*: the DM gets to control the navigation by changing the navigation direction and speed to find better solutions, and
- *projection*: if the navigation set is an approximation of the Pareto front and the DM has found an interesting solution, it can be projected to the closest Pareto optimal solution.

These modules were defined for navigation methods for computationally expensive problems but as the authors note, excluding projection which is not needed when the navigation set is the actual Pareto front, they are also valid in the case of navigation methods for computationally inexpensive problems (Hartikainen, Miettinen, and Klamroth 2019). For computationally expensive problems, the navigation set is an approximation of the Pareto front, generated by using some a posteriori method. Therefore, to ensure that the final solution at the end of the navigation is Pareto optimal, some *achievement scalarizing function* (Miettinen 1999) may be used to *project* the approximated solution onto the Pareto front. If we have an approximated solution, it can be set as a *reference point* $\mathbf{q} = (q_1, q_2, \dots, q_k) \in \mathbb{R}^k$, and it can be

projected onto the Pareto front by solving the following achievement scalarizing problem

$$\begin{aligned} \min_{\mathbf{x}} \max_{i=1,\dots,k} \left[\frac{f_i(\mathbf{x}) - q_i}{z_i^{nadir} - z_i^{utopia}} \right] + \rho \sum_{i=1}^k (f_i(\mathbf{x}) - q_i) \\ \text{subject to } \mathbf{x} \in S, \end{aligned} \quad (2.2)$$

where $z_i^{utopia} = z_i^{ideal} - \varepsilon$ are components of a *utopian point* $z^{utopian} \in \mathbb{R}^k$, $\varepsilon > 0$ is a small constant and $\rho > 0$ is a small augmentation coefficient. It can be proven, that when an appropriate solver is used, the solution to problem (2.2) is Pareto optimal (Miettinen 1999; Saini et al. 2022).

Hartikainen, Miettinen, and Klamroth (2019) also introduced a set of desirable properties for navigation methods which will be used as a basis for the synthesis in Chapter 4. These properties were divided into two categories, technical properties and properties related to user experience. The technical properties are, in a sense, easier to measure since a method either fulfills them or does not. The technical properties are as follows:

- *Navigation is complete*: Any feasible Pareto optimal solution is reachable for the DM with some preference information from any current solution. If the navigation set is an approximation of the Pareto front, the Pareto optimal solution is a projection of a point in the navigation set.
- *Navigation is computationally efficient*: If the original problem is computationally expensive, using an approximation of the Pareto front as the navigation set should make the computation of solutions faster. The DM should get results in real-time.
- *Construction of the navigation set is computationally efficient*: Although this can be, and mostly is, done offline before the involvement of the DM, it should not take long.
- *Accuracy of the navigation set can be measured*: In many cases, the navigation set is an approximation of the Pareto front, so its accuracy should be measurable. This accuracy information should tell when the navigation set is accurate enough for the navigation process and can also be shown to the DM if needed.
- *Accuracy of the navigation set can be improved*: If the navigation set is not accurate enough, the accuracy can be improved, for example, by adding more Pareto optimal solutions to construct the approximation.

The properties related to user experience, on the other hand, and whether a method fulfills them or not is more difficult to measure and may depend on who you ask, and ultimately on the user and the graphical user interface implemented for the method. These properties are as follows:

- *The DM can control the navigation:* Can the DM reach some reachable values in the navigation if they want, and if they want to restrict a certain area from being navigated to, are they able to accomplish this?
- *Low cognitive load is set on the DM:* The idea is to set as little cognitive load on the DM as possible. This means that controlling the navigation should be intuitive and the process should be visualized in an understandable way.
- *The DM is allowed to learn:* The DM should be able to learn about the problem and the feasible solutions and change their mind. They should also be able to take steps backwards or go back to a solution that has already been passed in the navigation process.
- *The DM can get additional information of the navigation set:* Additional information here means that the DM should receive more information in addition to the found solutions. This information could be, for example, the reachable ranges of the objective values in the navigation set or the set of feasible navigation directions from the current solution.

Navigation methods have been utilized in many different areas. One area in which many different navigation methods and decision support systems have been developed is radiotherapy treatment planning. There, the multiobjective optimization problem comes from trying to control local tumors while minimizing the side effects in the surrounding normal tissue and organs by determining the appropriate dosage of radiation (Thieke et al. 2007). Navigation methods in radiotherapy treatment have also been developed by Ehrgott and Winz (2008), Craft and Monz (2010), Fredriksson and Bokrantz (2013), Lin and Ehrgott (2018) and Collicott et al. (2021).

Some other application areas are chemical process design (Bortz et al. 2014; Baldan et al. 2023), lot sizing (Kania et al. 2021), water distribution system design (Moazeni and Khazaei 2021) and, for example, stockpiling critical materials for a national emergency (Korhonen

and Wallenius 1988). To further explore the existing navigation methods, a literature review has been conducted, and the review process and results are described in the next chapter.

3 Literature review

In this chapter, the steps of the literature review process are explained in detail. The aim of the review is to find the navigation methods introduced in the literature. First, an overview of the review process is given and then the different steps are explained in more detail. Finally, the results of the review are briefly presented.

The review was conducted by compiling search queries and inserting them into four different databases' search engines. The databases used in the search, that are also listed in Table 1, were Scopus, Web of Science Core Collection, ACM Digital Library's The ACM Guide to Computing Literature, and IEEE Xplore. These databases were chosen because of their citation indexing and the reproducibility of their search results. After performing the searches, the search results were surveyed and each result was either included or excluded from the review based on their title, abstract and, in some cases, introduction or conclusion chapters using the inclusion and exclusion criteria that are introduced later in the chapter. Then, to find more potentially relevant papers, the references of the included papers and also papers citing them were analyzed utilizing the same inclusion and exclusion criteria.

The search process began by iteratively formulating a search query, that would result in as many relevant results as possible, without having to go through too many irrelevant results. From the beginning, it was clear that this was going to be difficult since navigation methods do not have a common and universally used terminology. Therefore, trying to find a comprehensive set of keywords — that form the search query — was done iteratively by using one search query to first find some relevant articles, and then going through the keywords used in the articles and adding them to the query and doing the search again. The keywords used in the search queries were exclusive, which probably caused some relevant papers to have been left outside the search results. On the other hand, the exclusiveness of the keywords also seemed to exclude the vast majority of the irrelevant papers from the search results, which made analyzing the results much more efficient. Furthermore, many different combinations of keywords were tried and the combination which resulted in the seemingly best and most relevant results was chosen as the final one. The final search queries used in different databases are listed in Table 1. The queries are equivalents and modified to match the format

required by each database.

Database	Search query
Scopus	multicriteria OR multiobjective OR "multiple criteria" OR "multiple objective" OR "vector optimization" AND "navigation method" OR "navigation based method" OR navigator OR "pareto race" AND optimization AND interactive OR decision OR preference*
Web of Science	((ALL=(multicriteria)) OR ALL=(multiobjective) OR ALL=("multiple criteria") OR ALL=("multiple objective") OR ALL=("vector optimization")) AND (ALL=("navigation method") OR ALL=("navigation based method") OR ALL=(navigator) OR ALL=("pareto race")) AND (ALL=(interactive) OR ALL=(decision) OR ALL=(preference*)) AND ALL=(optimization)
ACM Digital Library	[[All: multiobjective] OR [All: multicriteria] OR [All: "multiple criteria"] OR [All: "multiple objective"] OR [All: "vector optimization"]] AND [[All: "navigation method"] OR [All: "navigation based method"] OR [All: "navigator"] OR [All: "pareto race"]] AND [All: "optimization"] AND [[All: interactive] OR [All: "decision"] OR [All: "preference*"]]
IEEE Xplore	("All Metadata":multiobjective OR "All Metadata":multicriteria OR "All Metadata": "multiple criteria" OR "All Metadata": "multiple objective" OR "All Metadata": "vector optimization") AND ("All Metadata": "navigation method" OR "All Metadata": "navigation based method" OR "All Metadata": navigator OR "All Metadata": "pareto race") AND ("All Metadata": interactive OR "All Metadata": decision OR "All Metadata": preference*) AND ("All Metadata": optimization)

Table 1. Databases used in the literature review and the corresponding search queries used in the databases.

Some inclusion and exclusion criteria were formulated to determine which papers were relevant to the review and which were not. The purpose of the criteria was to rule out some of the irrelevant search results that were still found with the final search queries. The criteria were updated as different search queries were experimented with to make the criteria more comprehensive and the final inclusion and exclusion of the found papers more efficient. The inclusion and exclusion criteria were also formed in a way so that they would not contain any unnecessary criteria that would exclude relevant papers. For a paper to be included, one of the inclusion criteria had to be fulfilled. Also, for a paper to be excluded, one of the exclusion criteria had to be fulfilled. The two exclusion criteria were that the paper was not written in English and that a full-text version of the paper was not available, in spite of attempts to get access. No papers that fulfilled either of the exclusion criteria were found.

The inclusion criteria were:

- The paper introduces a navigation method or an implementation or an application of a navigation method in multiobjective optimization. This distinction was necessary because some of the search results use such terms as "navigation" or "navigator" in a different context to the definitions of navigation and navigation methods in Chapter 2.
- The paper gives insight into different types of preference information or graphical user interface implementation ideas related to navigation methods.

After the search queries were finalized, the final searches were conducted and the results from the four databases were gathered for further investigation. The final searches were conducted on October 25, 2023. Then a preliminary qualification process was conducted with the search results. In the preliminary qualification, it was checked whether the found papers fulfilled the exclusion criteria. If a paper was not written in English, it would be excluded from the review and put on a list with other excluded papers. All the papers that were found were written in English, so this criterion did not exclude any otherwise possibly relevant papers. If the full-text version of a paper was not available through the publisher, it was added to a list with the other search results with markings that indicated that a full text of the paper had not yet been attained.

After the preliminary qualification, the full-text versions that had not yet been attained were

looked for. First, they were searched for via the Finna service to see whether the paper was available in some university in Finland or The National Repository Library in a digital or physical form. If the paper was still not found, then the authors were contacted for a copy of the paper. All the full-texts of the papers to be included in the review were found through this process so eventually no paper was left out of the review due to inaccessibility. The rest of the papers found in the searches were all added to Microsoft Excel spreadsheets, with each database having its own corresponding Excel file. Then, as a second qualification, the inclusion criteria were used to determine which of the found papers were to be included in the review and which to be excluded from it. After this qualification process, the included papers were then added to another Excel file to be able to see all the relevant papers at once. At this point, all the duplicates from different databases were also removed.

As mentioned before, forming a set of keywords to find relevant papers was not trivial. Therefore, only some of the papers used in the review were actually found using the search queries. The rest of the papers for the review were found by going through the already included papers' citations and the papers that had cited them. The same process that was used on the search results was also used to find out which of these cited papers were relevant and to be included in the review. Altogether from the four databases, 187 results were found using the search queries, including some duplicates across the databases. Out of the papers found, 11 were considered relevant according to the criteria. From the citations, 12 more relevant papers were found, so in total 23 papers were included in the literature review.

Reading the papers was the next step in the review process. It could be started as soon as the first papers to be included were determined. All the papers that were qualified for the review were read, and while they were being read, notes were taken that could then be used in the synthesis. In the next chapter, a synthesis is formed based on the results of the literature review described in this chapter.

4 Synthesis

In this chapter, a synthesis is formed of the navigation methods found in the literature review described in Chapter 3. First, a classification of the methods, according to the type of navigation set the DM navigates in, is introduced. Then the methods are compared in terms of the desirable properties for navigation methods introduced by Hartikainen, Miettinen, and Klamroth (2019), and listed in Chapter 2, and how the methods fulfill the properties.

4.1 Classification

In this section, a classification of different navigation methods found in the literature review is introduced. The classification was done according to the type of navigation set the DM navigates in. Different types of navigation sets here are named Pareto front, approximation of the Pareto front, and NAUTILUS-type. In what follows, these three types are described in more detail in the order they were listed. A graphical user interface implementation is presented for each method. The interfaces are explained shortly, for more information about them, the reader is referred to the references in each figure's caption.

4.1.1 Pareto front

Navigating on the actual Pareto front in real-time is only possible when the multiobjective optimization problem is linear because solving many nonlinear and possibly nonconvex problems would take too long. With linear problems, during the navigation process, each successive Pareto optimal solution may be computed using parametric linear optimization with no necessity for any approximations.

Pareto Race (Korhonen and Wallenius 1988): Pareto Race is applicable to linear multiobjective optimization problems. Pareto Race does not need any information on the solutions to the problem before the navigation process. This is because no representation of the Pareto front is computed beforehand, but each solution to be shown is rather calculated during the navigation. Because the problems are linear, the Pareto optimal solutions can be computed in real-time. This makes the navigation set used in Pareto Race the actual Pareto front.

Pareto Race involves a graphical user interface to help the DM's navigation process, which is shown in Figure 1. The interface consists of bars representing each objective and their current value. As the DM navigates the Pareto front, the lengths of the bars change according to the changing objective values.

The navigation begins from a starting point that is a Pareto optimal solution. To compute the starting point, the DM gives aspiration levels for the objectives and the corresponding reference point is used in an achievement scalarizing function (2.2). Optimizing the achievement scalarizing function then gives the starting point. Similarly, during the navigation, Pareto Race utilizes the achievement scalarizing function to find Pareto optimal solutions by using the DM's aspiration levels. The DM controls the navigation using keyboard keys that are shown on the graphical user interface. With these controls, the DM may move in the current direction at a constant speed, increase the speed forward or backward, reduce the speed, turn, fix the aspiration level of an objective or relax the aspiration level of a fixed objective.

To turn, the DM chooses one objective to be further improved, which changes the reference direction used to compute new solutions. When an aspiration level of an objective is fixed, that objective becomes a constraint function. The objective's current value is set as an absolute lower or upper bound for the constraint function, depending on whether the objective is to be maximized or minimized. When a fixed aspiration level is relaxed, the function becomes an objective function again. The DM's actions to update the configurations of the navigation dynamically change the formulation of the problem being optimized. Therefore, the DM sees the effects that their actions have on the solutions found in real-time as they see the changing objective values in the graphical user interface.

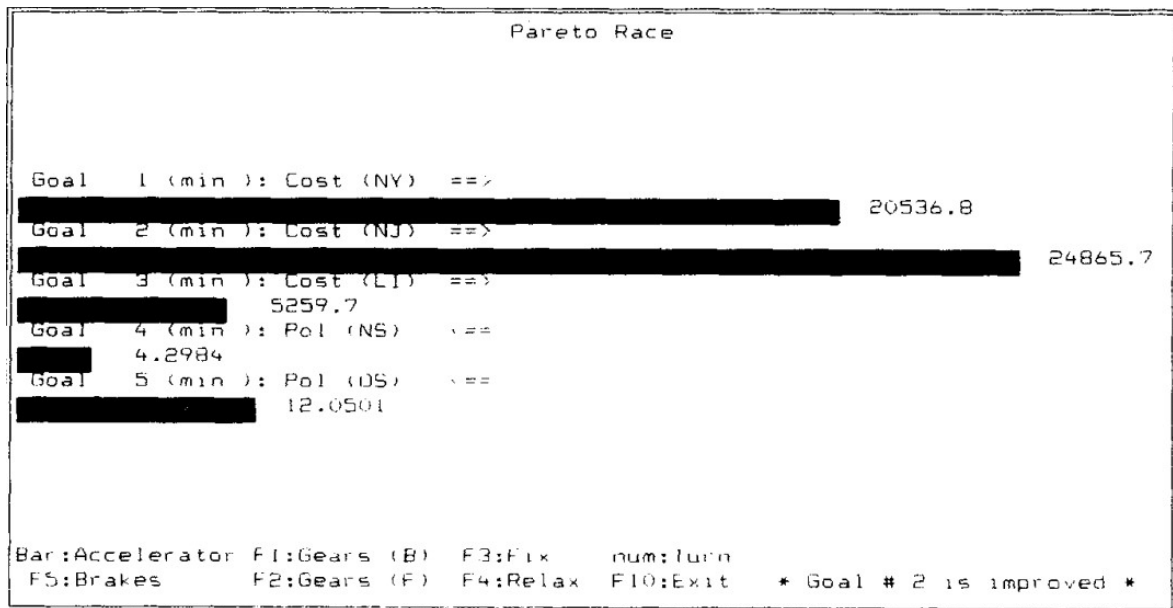


Figure 1. The graphical user interface used in Pareto Race. Taken from Korhonen and Wallenius (1988).

4.1.2 Approximation of the Pareto front

For nonlinear problems, computing many new solutions during the navigation would take too much time for the navigation to be real-time. Therefore, a representation of the Pareto front is generated using some a posteriori method, for example, an evolutionary algorithm¹, to make the navigation smoother. Depending on the a posteriori method used to generate the representation, the solutions that form the representation may be approximated Pareto optimal solutions. For example, evolutionary algorithms cannot guarantee that the solutions are globally nondominated i.e., Pareto optimal, which makes them approximated solutions. The approximated solutions should still be close enough to the Pareto optimal solutions so that the DM gets as good of an idea of the reachable Pareto optimal solutions as possible. The representation is generated offline before the involvement of the DM so that the DM does not have to wait. Also, as mentioned in Chapter 2 about the desirable properties for

1. Evolutionary algorithms are a posteriori methods that generate a set of solutions by using mechanisms inspired by evolution, for example, reproduction, mutation and selection. A fitness function is used to determine which solutions are the "strongest" among a set of solutions, a population. The strongest solutions are then used in the next evolution which generates more solutions.

navigation methods, if the approximation is not accurate enough, it should be possible to improve the accuracy.

The methods in this category assume a representative set of precomputed Pareto optimal solutions as a basis to generate the navigation set. The generated navigation set approximates the Pareto front. Therefore, the most preferred solution chosen by the DM at the end of the navigation process may not be Pareto optimal. If the most preferred solution is not Pareto optimal, it can be projected onto the Pareto front. The projection can be done by solving the achievement scalarizing problem (2.2) with the approximated solution as the reference point. Many of the methods also assume that the multiobjective optimization problems that are solved are convex, though some are also applicable to nonconvex problems.

Pareto navigation (Monz et al. 2008): The Pareto navigation method is applicable to convex problems and was originally developed for radiotherapy treatment planning. Pareto navigation assumes a set of precomputed Pareto optimal solutions. The precomputed solutions and all the convex combinations of these solutions form the navigation set. The accuracy with which the set of precomputed solutions represents the set of Pareto optimal solutions affects the quality and variety of the solutions found during the navigation. The MIRA navigator software (Thieke et al. 2007) is used for user interaction. The graphical user interface for MIRA navigator, which is designed for radiotherapy treatment planning, is shown in Figure 2. In the MIRA navigator, the DM can see the current objective values and the set of solutions, which shows the reachable ranges of values for each objective. The objectives are visualized as a radar chart on the left side of the graphical user interface with each objective as an axis of the radar. On the right side, there is some more information about the current solution related to radiotherapy treatment planning.

Pareto navigation has two navigation controls: selection and restriction. Selection means finding a new solution by changing the value of some objective. In the MIRA navigator (Figure 2), each axis of the radar chart has a selector at the current objective value that the DM can slide to change the corresponding objective value. Then a new solution is computed, by solving a linear problem, and the sliders presenting the objective values are moved accordingly for the other objectives. Restriction means excluding some solutions from the navigation by changing the obtainable range of values of an objective. Each axis of the

radar chart has a restrictor with which some solutions can be excluded. Moving a restrictor changes the shown obtainable ranges of values of the objectives so the DM sees the effects of their actions.

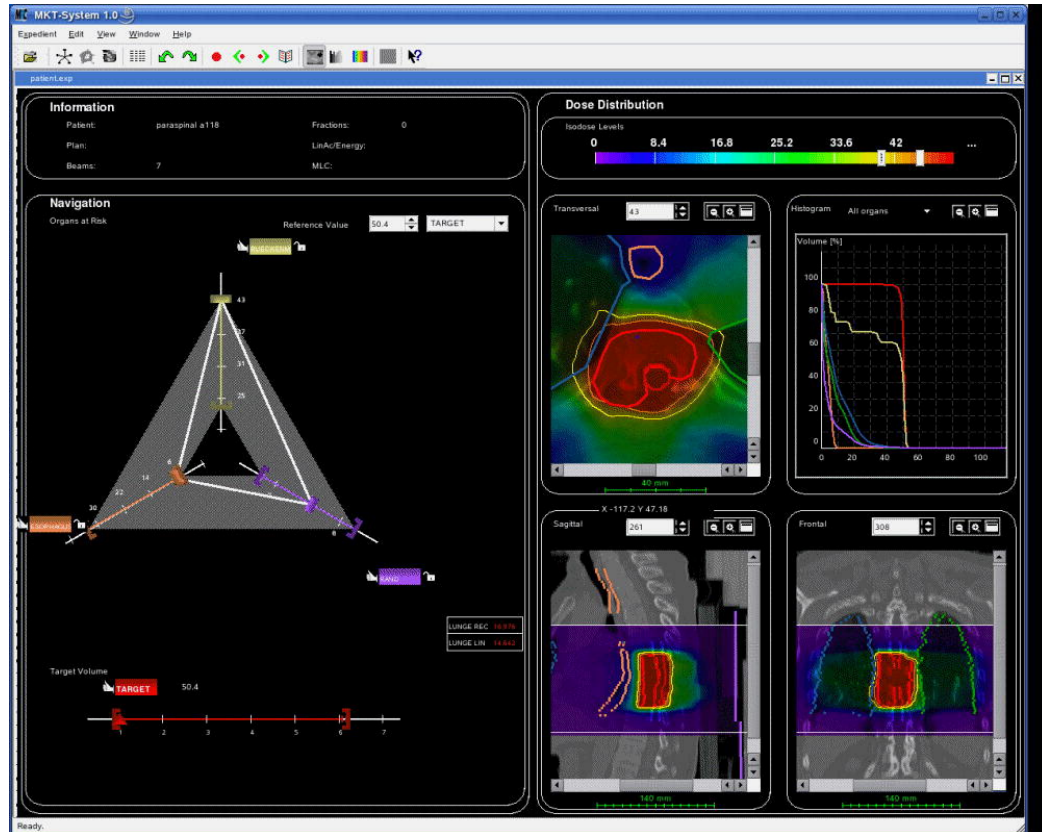


Figure 2. The graphical user interface used in MIRA navigator. Taken from Thieke et al. (2007).

Multiple Pareto surface navigation (Craft and Monz 2010): The method called multiple Pareto surface navigation is applicable to convex problems and was developed for radiotherapy treatment planning. Though the original problem may be nonconvex caused by defining the beam angles² as decision variables, the problem that is to be optimized with the method is made convex by removing the beam angles as decision variables. Instead, the different beam angle configurations³ are used to define and compute multiple Pareto fronts. The method

2. Beam angle optimization means determining how many beams are to be fired at the patient and from what angle (Bertsimas et al. 2013).

3. Beam angle configuration is a choice of how many beams, and from what angle, are fired at the patient.

assumes a set of precomputed Pareto optimal solutions and multiple Pareto fronts (and beam angle configurations) that consist of the precomputed solutions and linear combinations of them. The multiple Pareto fronts that form the navigation set are therefore approximations of the Pareto fronts.

A graphical user interface has been developed to assist the DM in the navigation process, which is shown in Figure 3. The interface and the visualization used are designed for radiotherapy treatment planning and are not meant for a general problem. To navigate in a single Pareto front, the DM adjusts the significance of the objectives by using a checkbox and buttons related to each objective: lock, switch, up, and down. Each objective is visualized as its own graph and next to each graph are the buttons corresponding to that objective. By pressing up or down the DM increases or decreases the significance of the corresponding objective, and the next solution is computed by solving a linear problem. Locking an objective by checking a checkbox, means that the value of the corresponding objective will not change. For navigation on multiple Pareto fronts, the DM can press one of the switch buttons. This causes the navigated front to change. The next front to be navigated on will be the closest front that has a chance to improve the value of the corresponding objective. The DM has two options to handle the front switches: manual and automatic. When the DM chooses to switch fronts manually, they have more control in the navigation process as the front is switched only when the DM wants. If the DM chooses automatic switching, when buttons up or down are pressed, optimization is done on all fronts and the front with the best value for the corresponding objective becomes active.

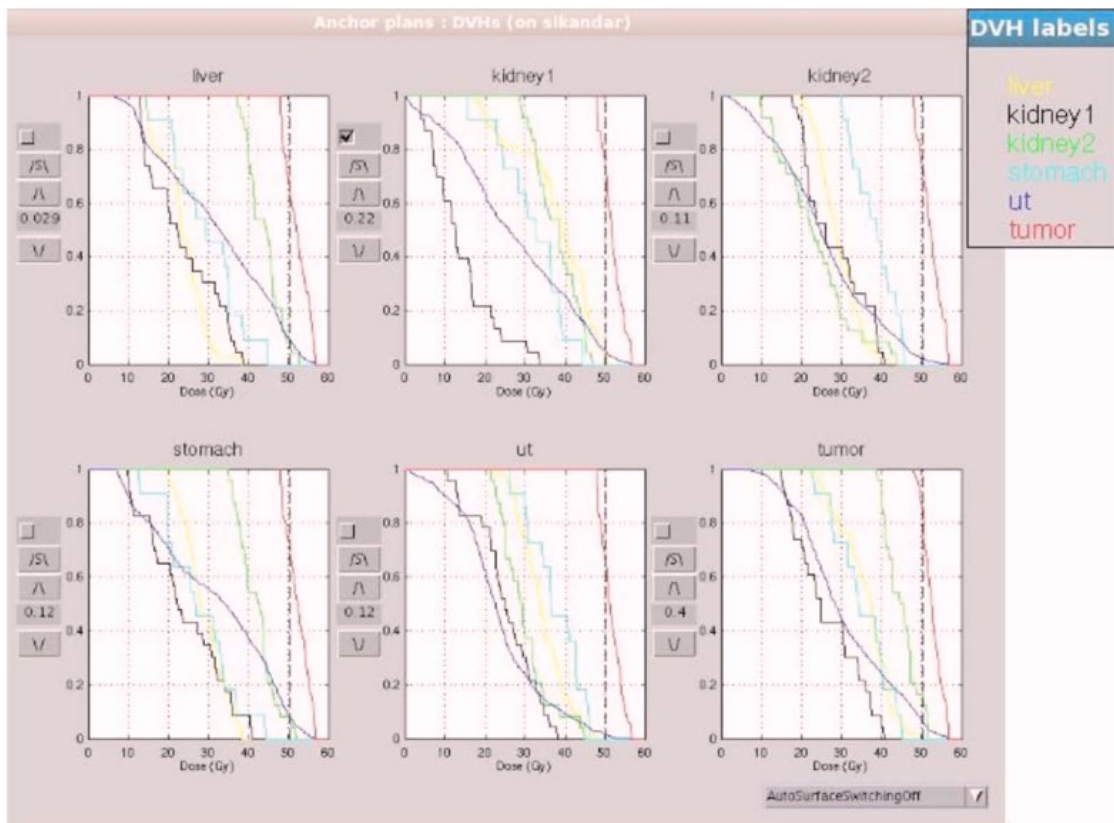


Figure 3. The graphical user interface used in multiple Pareto surface navigation. Taken from Craft and Monz (2010).

Pareto Navigator (Eskelinen et al. 2010): The Pareto Navigator method is applicable to convex problems. In the beginning, a small set of Pareto optimal solutions is assumed to have been generated. This set of solutions is then used to generate a polyhedral approximation of the Pareto front, which is used as the navigation set. A graphical user interface is shown in Figure 4, which shows on the left the current objective values and aspiration levels, in the middle panel a set of Pareto optimal solutions, and on the right the objective values seen during the navigation as a continuous line. The DM may change the speed, in practice the step size, by moving the slider at the bottom of the graphical user interface, and the direction of the navigation, by changing the preference information, at any point. The DM may also go back to any previous solution, by moving the red vertical line in Figure 4, and continue the navigation from there. In this way, the DM is able to learn about the changes in solutions that their choices of preferences cause.

At the start of the navigation, the DM is asked to choose a starting point among the available Pareto optimal solutions. The DM navigates in the approximated Pareto front by giving their preference information, for example, as a reference point. The preference information and the current solution, or the starting point at the beginning, are used to compute a navigation direction. The DM is then shown new solutions, that are generated with small step sizes, in that direction dynamically using the graphical user interface. New solutions are computed by solving parametric linear optimization problems. The DM may ask for the approximation to be regenerated based on some interesting area they have found to get a more accurate approximation of that area. When the DM has found a preferable solution and that solution has been projected onto the Pareto front, the DM may decide whether they would like to continue the navigation. If the projected Pareto optimal solution is satisfactory, they can end the navigation process. If, on the other hand, they would like to keep looking for an even more desirable solution, they can continue the navigation.

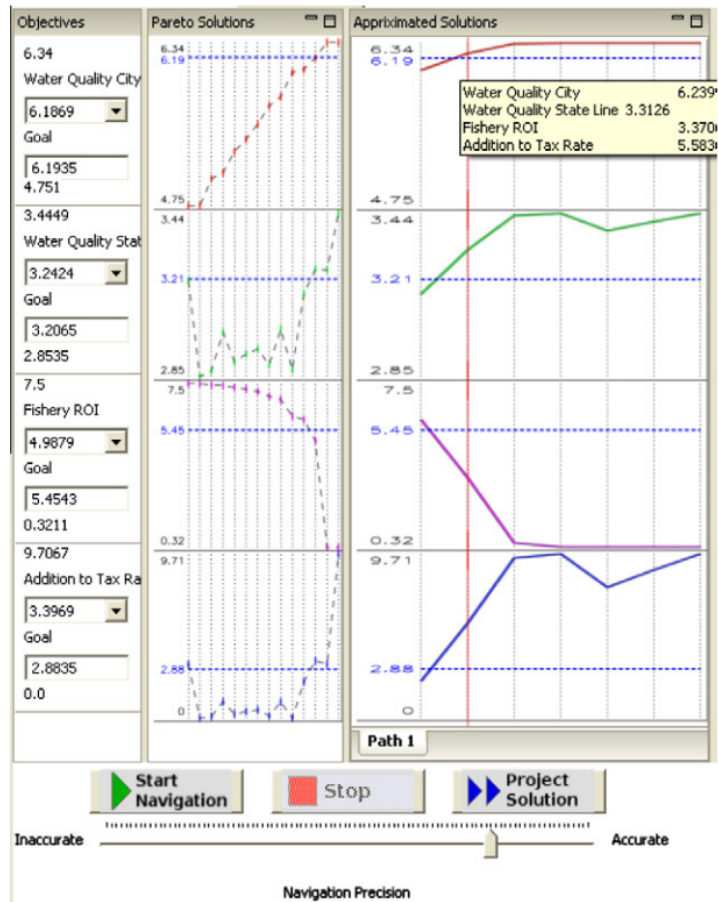


Figure 4. A graphical user interface for Pareto Navigator. Taken from Tarkkanen et al. (2013).

Bortz et al.'s (2014) method: The method proposed by Bortz et al. is applicable to non-convex problems and was originally developed for chemical process design optimization problems. To generate an approximation of the Pareto front, some Pareto optimal solutions are needed. The approximation is generated using a method that combines sandwiching⁴ and hyperboxing⁵ algorithms to be able to approximate nonconvex Pareto fronts. The DM navigates on the generated approximation using a graphical user interface with sliders for

4. Sandwiching algorithm computes upper and lower approximations for a convex function to approximate the Pareto front. Weighted sum scalarization is used to compute new Pareto optimal solutions to create the approximations until the upper and lower approximations are close enough to each other (Bortz et al. 2014).

5. Hyperboxing algorithms divide the objective space into boxes with Pareto optimal points on some corners. These boxes are then tested for convexity as described in Bortz et al. (2014). Hyperboxing algorithms are, in this case, used to detect nonconvexities on the Pareto front.

each corresponding objective. Figure 5 shows how the sliders work in two dimensions. To navigate, the DM can move one slider at a time to change the value of that corresponding objective. While the DM moves a slider, the method finds a new solution to match that objective value. The DM can then see the changes that are caused in the other objectives. The DM may also set ranges for the objective values as lower and upper bounds.

Some variables are used as input variables in the process simulation and are constant during the optimization. The rest of the variables are called free variables, and they define the decision space. The user interface also has sliders for the free variables for the DM to be able to compare solutions in the design space as well as the objective space. The sliders for the free variables, however, cannot be moved by the DM, but are moved as the objective and restriction sliders are moved. When the DM moves an objective slider, a linear problem is solved to determine the other objective values and the other objective sliders are moved accordingly. Because the navigation set is an approximation, when the DM finds a desirable solution, it may not be Pareto optimal. Therefore, a final simulation run is conducted using the Pascoletti-Serafini scalarization (Bortz et al. 2014; Pascoletti and Serafini 1984) with the found solution as a reference point.

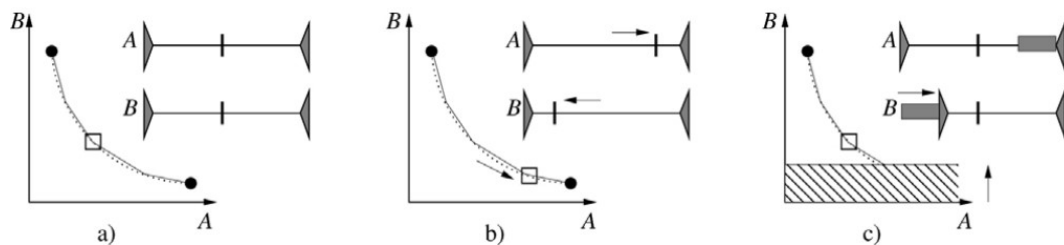


Figure 5. Illustration of the use of objective and restriction sliders in Bortz et al.'s (2014) method. Taken from Bortz et al. (2014).

Lin and Ehrgott's (2018) method: The method proposed by Lin and Ehrgott is applicable to nonconvex problems. It was originally developed for radiotherapy treatment planning. To generate a representation of the Pareto front for nonconvex problems, a set of Pareto optimal solutions is computed with a procedure explained in Lin and Ehrgott (2018). This representation then acts as the navigation set. To make the navigation process easier for the DM, a graphical user interface has been developed and is shown in Figure 6. The graphical

user interface has three main components for each objective: an objective slider, an aspiration slider and a constraint check box. Each slider is limited by the highest and lowest values that the objective has in the navigation set.

The DM gives their preference information as aspiration levels for the objectives which happens by moving an aspiration slider. When an aspiration slider is moved, a new solution is computed by solving a linear problem based on the new aspiration levels and the objective sliders are updated. The DM can also move the objective sliders which sets the new value of the objective as a constraint to the linear problem and the problem is solved to get a new solution and the other objective sliders are updated. When the DM activates a constraint check box, a hard constraint is added to the linear problem based on an upper bound the DM sets for the objective. The user interface also informs the DM whether an aspiration value is satisfied by using background colors for the objective value text boxes. If the background color is green, it means the aspiration level is reached. If, on the other hand, the background color is red, the aspiration level has not been reached for that objective.

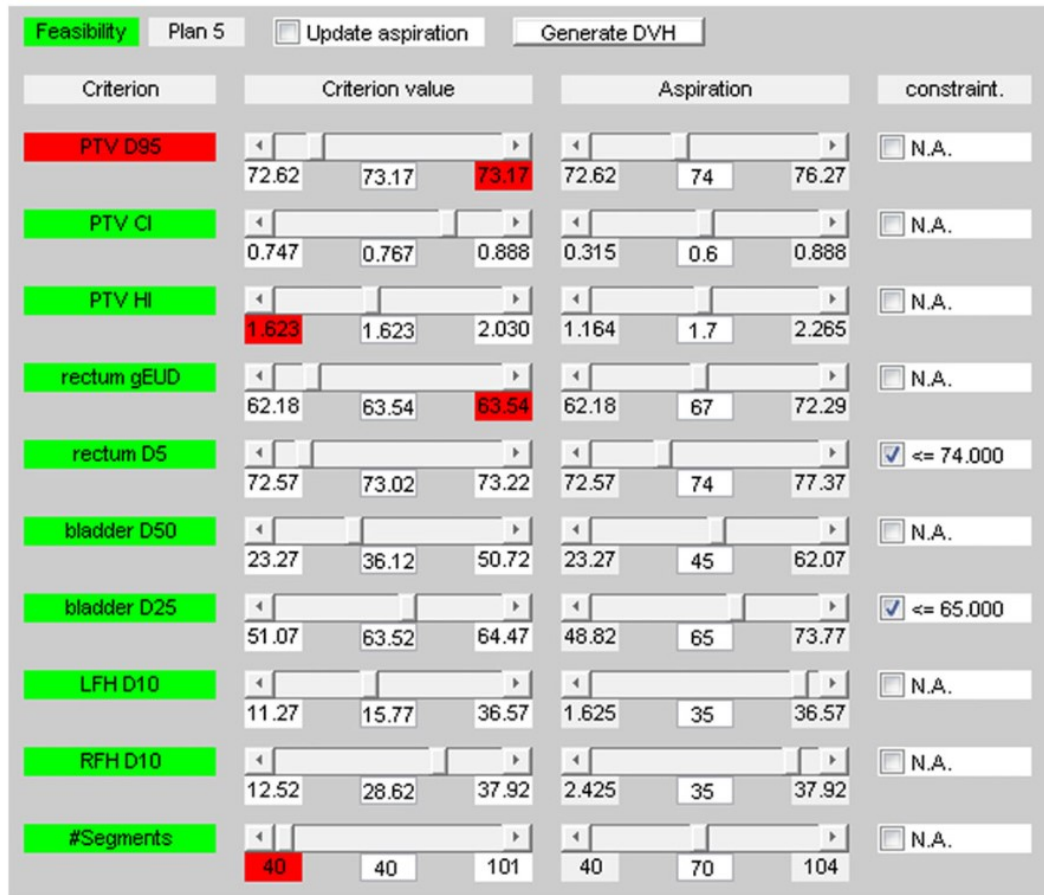


Figure 6. The graphical user interface used in Lin and Ehrgott’s (2018) method. Taken from Lin and Ehrgott (2018).

Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019): The Nonconvex Pareto Navigator method extends the Pareto Navigator method introduced above to nonconvex multiobjective optimization problems. Like in Pareto Navigator, a set of Pareto optimal solutions is needed to create the navigation set. The navigation set in Nonconvex Pareto Navigator is an approximation of the Pareto front generated with the PAINT method (Hartikainen, Miettinen, and Wiecek 2012) and so-called e-cones. The e-cones are meant to join the disconnected parts of the PAINT approximation to make navigation possible since the Pareto front of nonconvex problems can be disconnected.

The DM can see the solutions and control the navigation in a graphical user interface shown in Figure 7. In the left panel, for each objective are, in order top to bottom, current objective

value, aspiration level and upper bound (lower for objectives that are maximized), checkboxes to disable the corresponding aspiration level and bound. The middle panel shows the set of computed Pareto optimal solutions as continuous lines and bounds and aspiration levels as green and blue dotted lines. The right panel shows the navigated solutions as a continuous line and bounds and aspiration levels as in the middle panel. The red vertical line represents the starting point and the slider in the bottom allows the DM to change the speed (in practice the step size) of the navigation.

As in Pareto Navigator, the DM navigates in the approximation by giving their preference information in the form of aspiration levels, that they can change at any point during the navigation process. The DM may also set upper bounds for the objectives not to be exceeded. New solutions are computed by solving mixer integer linear optimization problems. At any time, the DM can choose a previous solution and continue the navigation from there by updating the preference information. Like in Pareto Navigator, if the final solution, that is projected onto the Pareto front, is not satisfactory to the DM, the accuracy of the approximation can be improved if the DM chooses to continue the navigation.

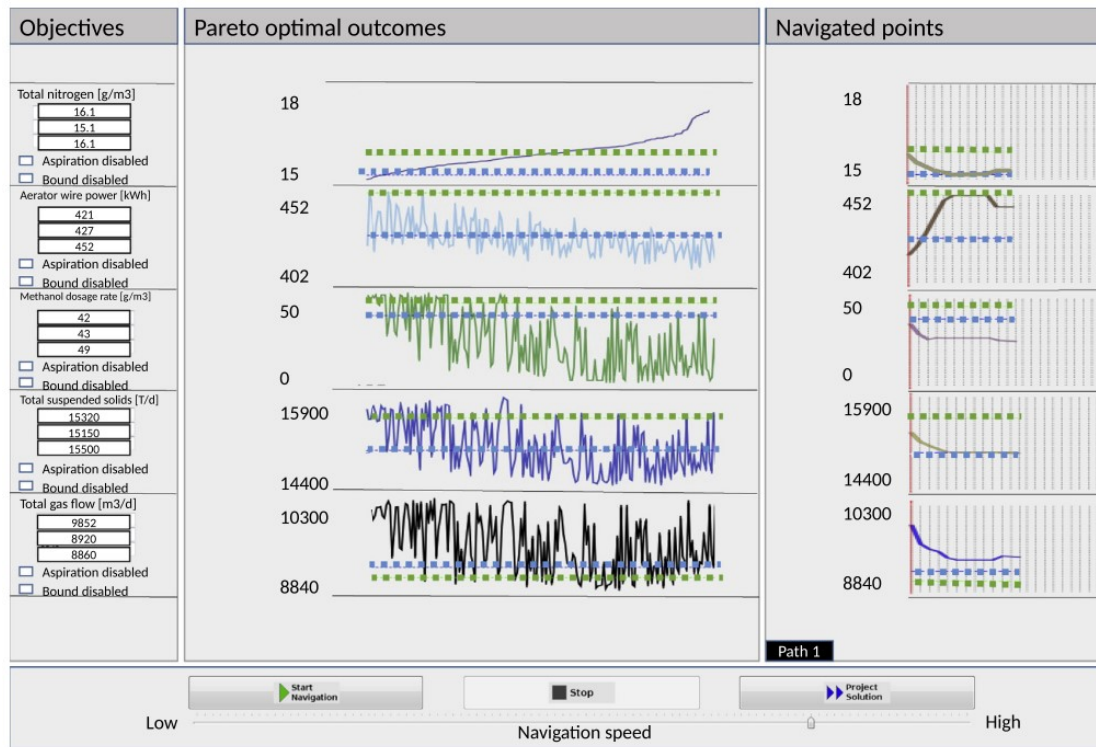


Figure 7. The graphical user interface for Nonconvex Pareto Navigator. Taken from Hartikainen, Miettinen, and Klamroth (2019).

Patch navigation (Collicott et al. 2021): The patch navigation method is applicable to non-convex problems with both continuous and binary variables. The number of binary variables should be relatively low. The binary variables represent on/off type choices the DM may have to make. Making a choice on all the binary variables forms a multiobjective optimization problem with the continuous variables as decision variables, and the chosen binary variable values as constants for the problem. The Pareto optimal solutions for such a problem form one Pareto front. The combination of different choices for binary variable values results in multiple Pareto fronts. The idea of patch navigation is to use Pareto navigation, that is introduced above, to navigate across multiple convex Pareto fronts that compose the original nonconvex problem's Pareto front. A patched Pareto front is a union of finitely many individual convex Pareto fronts, that are called patches. The patches are required to be approximated by a finite set of Pareto optimal solutions. These approximated patches form the navigation set.

Patch navigation has a graphical user interface, which is shown in Figure 8. The graphical user interface lets the DM see the changes in the reachable values of the objectives as they include or exclude a patch. Patch solutions are the "best" solutions of each separate patch, and the so-called global solution is the "best" solution among the patch solutions. A so-called merit function, defined in Collicott et al. (2021), is used to determine which solutions are the "best" in comparison. The graphical user interface displays the current global solution and patch solutions. The method also shows the DM the distance of the current solution to the closest solutions on the other patches. This lets the DM know whether their preferences are met on some other patches, or if a solution like the current chosen solution can be found on a different patch. The solutions on different patches are shown in different colors which lets the DM compare solutions from different patches. The method has three actions the DM may perform: selection, restriction and patch activation or deactivation. The selection and restriction actions are similar to the ones mentioned above related to Pareto navigation. In patch activation or deactivation, the DM may include or exclude a patch to be able to navigate to the more interesting areas of the navigation set. When the DM performs any of the actions, the method solves linear problems to update the objective values.

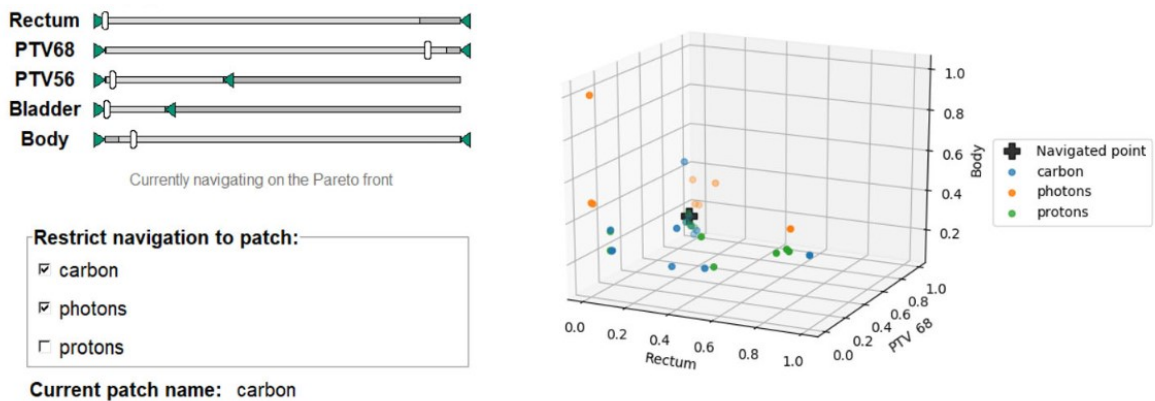


Figure 8. The graphical user interface used in patch navigation. Taken from Collicott et al. (2021).

iSOM-Pareto Race (Yadav, Ramu, and Deb 2022): The iSOM-Pareto Race method aims at expanding the Pareto Race method to be effectively applicable to both linear and non-linear multiobjective optimization problems, though no assumption on the problems' convexity is mentioned. This is done by utilizing a reference direction-based evolutionary algorithm, RD-NSGA-II by Deb and Kumar (2007) with an achievement scalarizing function to compute each solution during the navigation process. Self-organizing maps (SOM) map high-dimensional data, in multiobjective optimization the multiple objective vectors, to a 2-dimensional representation. This is meant to help with reading high-dimensional data and, in the context of navigation methods, to visualize the solutions to the DM in an understandable way. Interpretable self-organizing maps (iSOM) is a modified SOM which is said to be inherently understandable (Thole and Ramu 2020).

The preference information given by the DM is the same as in Pareto Race: aspiration levels as a reference direction to start the navigation and then using the Pareto Race controls to move around in the navigation set. As in Pareto Race, an achievement scalarizing function is used to find the next solution during the navigation. Figure 9 shows the iSOM visualization used in the method. The method's visualization has a graph for each objective (named f_1 , f_2 and f_3 in Figure 9) that uses a heat map to show where the optimal values for that objective can be found. The U-Matrix, G and T graphs offer the DM some additional information about the quality of the solutions in different regions as heat maps. More information about the interpretation of the visualization can be found in Yadav, Ramu, and Deb (2022).

The graphs also use arrows to show the navigation steps that have been taken. Each solution displayed—the pregenerated solutions and each navigated solution—is color-coded as in Figure 10, which gives the DM more information about the solutions. In the beginning, RD-NSGA-II is used to create a well-distributed set of solutions that approximate Pareto optimal ones, that are then visualized. The DM then provides the preference information to start the navigation that is controlled as in Pareto Race. New solutions are computed by repeatedly solving an achievement scalarizing function, more information about the method in Yadav, Ramu, and Deb (2022).

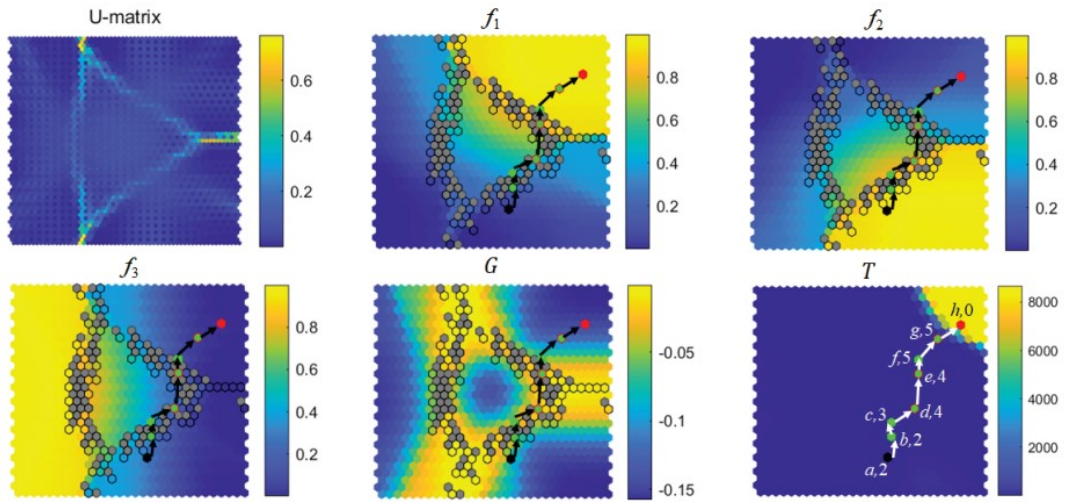


Figure 9. The iSOM visualization used in iSOM-Pareto Race. Taken from Yadav, Ramu, and Deb (2022).









S.No.	Points	Edge	Face	Color	Node
1	Boundary points	Yes	No	Black	
2	Near-constraint points	No	Yes	Grey	
3	Start point/current solution	No	Yes	Black	
4	Turn	No	Yes	Magenta	
5	Accelerator	No	Yes	Green	
6	Brakes, Gears	Yes	No	Red	
7	Final solution	No	Yes	Red	
8	Progress of Pareto Race	—	—	—	

Figure 10. The color codes used by the iSOM visualization. Taken from Yadav, Ramu, and Deb (2022).

Real-time Interactive Navigation within a Data set (Baldan et al. 2023): The method called Real-time Interactive Navigation within a Data set (RINADA) is applicable to convex problems and was originally developed for chemical process engineering. The method is

meant for data (solutions) obtained from a flowsheet simulation and assumes a discrete set of solutions. Surrogate models are then used to approximate the original set of solutions to shorten the distance between solutions during navigation. Based on the discrete set of solutions, an approximated convex hull is generated as the navigation set. The method makes no assumptions on the type of surrogate model used, but they are assumed to make the approximation relatively accurate. RINADA has a graphical user interface which is shown in Figure 11.

In RINADA, the DM controls the navigation by moving sliders to change the decision variable and objective values, and by setting bounds to the decision variables and objectives. Each decision variable and objective has its own slider. There are two types of navigation here: input and output navigation. In input navigation, the DM can see how changing the decision variable values effects the objective values. Input navigation happens when the DM drags a decision variable's slider. This makes the other decision variable sliders move, which in turn causes the objective sliders to move. Figure 11 shows an example of input navigation. In output navigation, the DM can see what changes in decision variable values are needed to reach certain objective values. Output navigation happens when the DM drags an objective's slider. This makes the decision variable sliders move so that the changed objective value will be achieved. The other objective sliders will then move according to the new decision variable values. In both input and output navigation, a nonlinear problem is solved to navigate to the next solution.

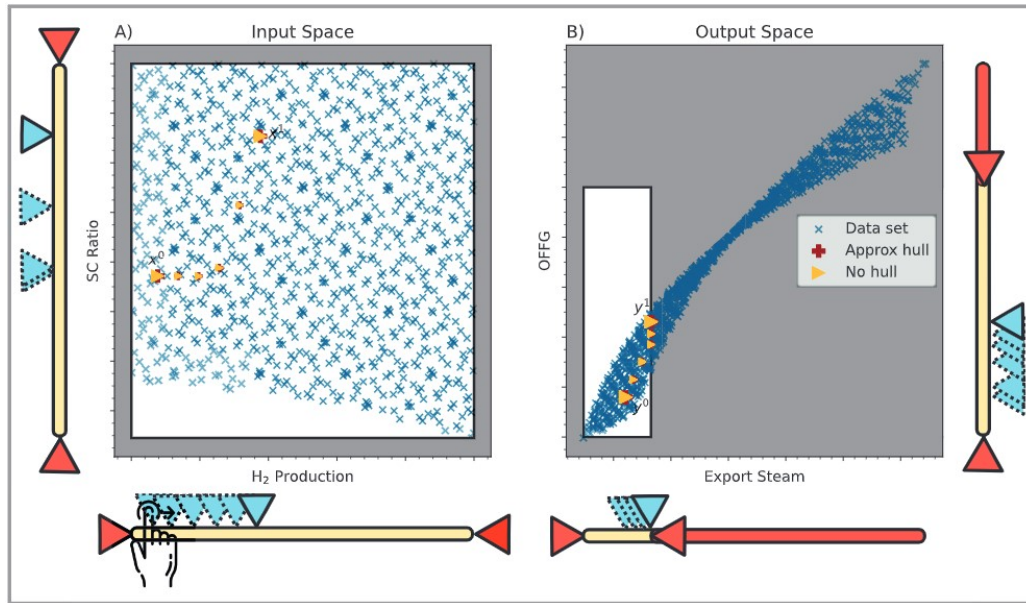


Figure 11. An example of input navigation in RINADA. Taken from Baldan et al. (2023).

4.1.3 NAUTILUS methods

Anchoring happens when a DM's thinking gets "stuck" on some possibly irrelevant information (Miettinen 1999). This may occur when the DM chooses a starting point and has difficulties changing their preferences to move away from the starting point as if they were tied to an anchor. The starting point of an interactive method matters which can be explained by anchoring (Miettinen and Ruiz 2016). If the starting point is not close to the most preferred solution, the DM may not be able to navigate to the most preferred solution. Because of the trade-offs between the objectives involved with Pareto optimal solutions, not having the necessity to "lose" in some objectives to "gain" in others may help the DM in reaching a more satisfactory solution (Miettinen et al. 2010). According to the prospect theory by Kahneman and Tversky (1979), people react asymmetrically to losses and gains as the reactions to losses have been observed to be stronger than gains.

NAUTILUS methods are interactive methods that aim to avoid anchoring by starting from the nadir point or some other point from which there can be seen improvements in all of the objective values (Miettinen and Ruiz 2016). In NAUTILUS methods, the DM moves iteratively towards the Pareto front, and at each iteration the DM sees improvement in all objective val-

ues. The solutions the DM sees during the solution process are not Pareto optimal, as only the final solution at the end is Pareto optimal. Not having to deal with trade-offs between the objectives helps the DM in avoiding anchoring and finding their most preferred solution. In what follows, two navigation methods, that follow the idea of NAUTILUS methods, are introduced.

NAUTILUS Navigator (Ruiz et al. 2019): The NAUTILUS Navigator method is applicable to any type of problem. The method assumes a precomputed set of approximated Pareto optimal solutions. NAUTILUS Navigator applies the idea of NAUTILUS methods by starting the navigation process from an estimated nadir point. Instead of seeing approximated Pareto optimal solutions during the navigation process, the DM sees the reachable ranges of the objective values from the current solution. These reachable ranges are updated in real-time during the navigation process and visualized in a graphical user interface, shown in Figure 12. The DM can also see the previous reachable ranges which helps the DM learn about the problem and the changing objective values.

The DM controls the navigation, using the graphical user interface, by defining aspiration levels for the objectives. The DM may provide the aspiration levels by typing them in the corresponding text boxes on the left side of the reachable ranges in Figure 12. The DM can also specify the speed, in practice the step size, in which the navigation process approaches the approximated Pareto front. Each successive solution is computed with a simple formula defined in Ruiz et al. (2019), and at each solution, the reachable ranges are computed by solving ϵ -constraint problems (Ruiz et al. 2019). The DM can see the reachable ranges shrink during the navigation process and can change their preference information at any point. The DM can also go back to a previous solution by clicking on the corresponding spot on the previous reachable ranges, and then continue the navigation from there by providing new preference information. At the end of the navigation process, assuming an approximation of the Pareto front is used, the final solution of the navigation is on the approximated Pareto front. The final solution may then be projected onto the Pareto front using an achievement scalarizing function.

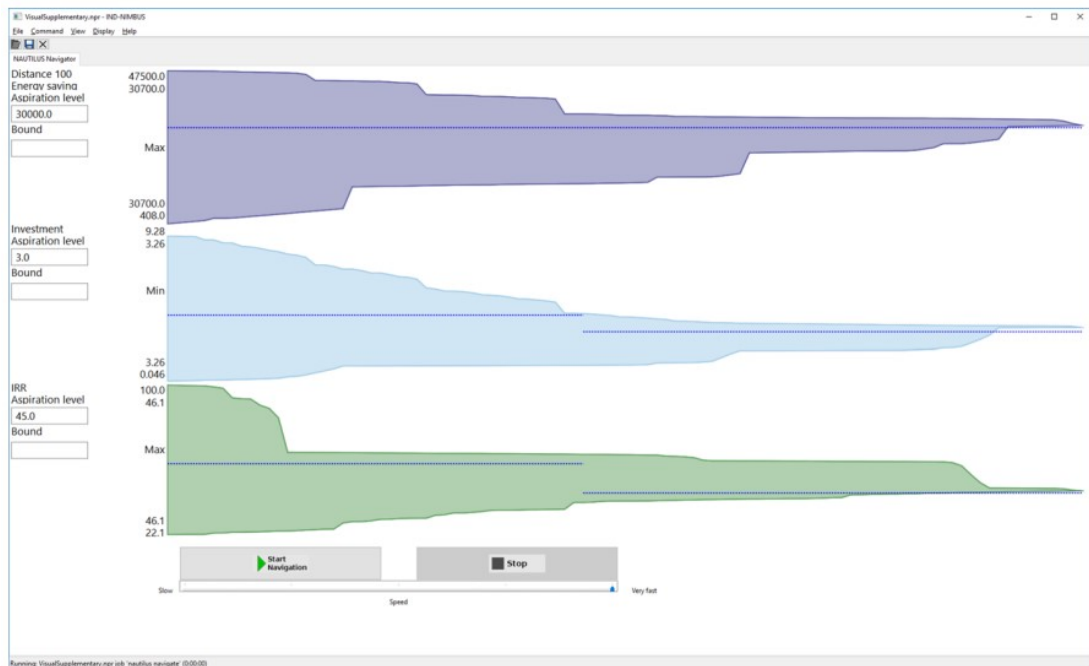


Figure 12. The graphical user interface of NAUTILUS Navigator. Taken from Ruiz et al. (2019).

Optimistic NAUTILUS Navigator (Saini et al. 2022): The Optimistic NAUTILUS Navigator (O-NAUTILUS) method is applicable to any type of problem and is especially meant to be able to handle computationally expensive problems. The method assumes a precomputed set of solutions, though no further assumptions are made of the set. O-NAUTILUS applies the idea of NAUTILUS methods by starting the navigation process from an estimated nadir point. The method utilizes surrogate models, that are used to replace and approximate the original computationally expensive objective functions. O-NAUTILUS has two sets involved in the navigation: a set of known solutions and an optimistic estimation of the Pareto front. The set of known solutions is the same as in NAUTILUS Navigator, representing the approximated Pareto front. The optimistic estimation of the Pareto front is calculated using the surrogate models.

The navigation view of O-NAUTILUS is shown in Figure 13. O-NAUTILUS shows the DM some additional information, besides the reachable ranges, during the navigation process. This additional information is the optimistic ranges that have solutions that are predicted to

have good objective values but are not included in the reachable ranges. During the navigation, both the reachable and optimistic ranges are updated in real-time and the DM can see the ranges evolve and shrink. As mentioned, the navigation starts from an estimated nadir point. The DM gives their preference information as aspiration levels for the objective values. To compute each successive solution, a formula defined in Saini et al. (2022), is used. At each solution, the reachable and optimistic ranges are computed, as explained in Saini et al. (2022). The DM can also, at any time, go back to a previous solution and continue the navigation from there by providing new preference information.

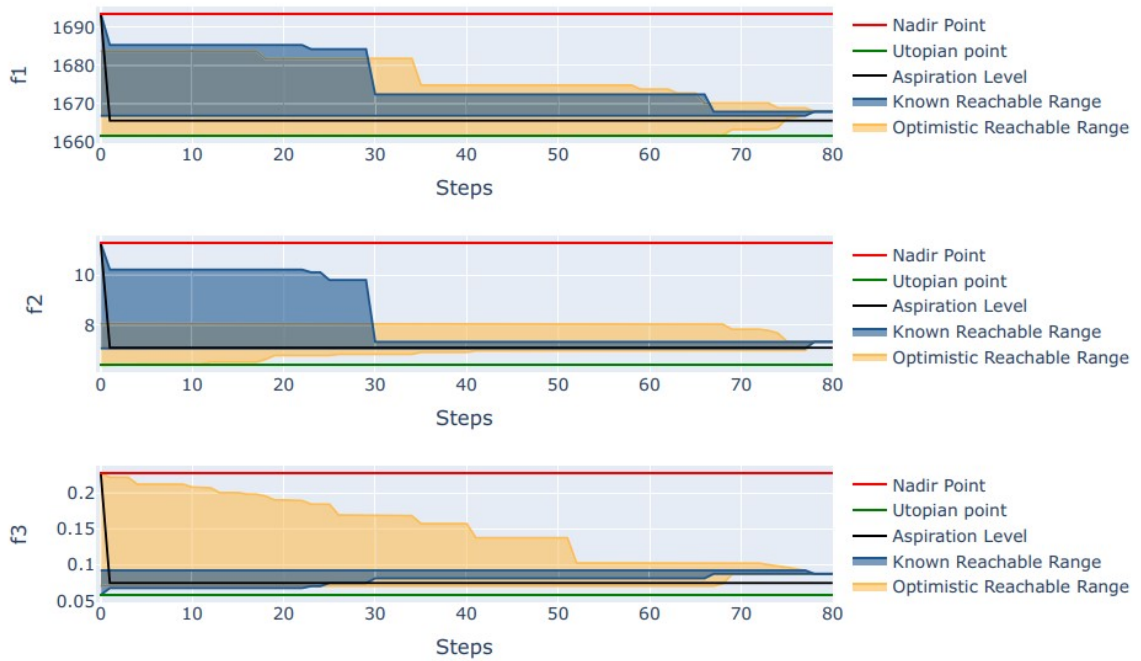


Figure 13. The navigation view of O-NAUTILUS. Taken from Saini et al. (2022).

4.2 Desirable properties

In this section, the methods introduced in Section 4.1 are analyzed according to the desirable properties for navigation methods from Hartikainen, Miettinen, and Klamroth (2019) that are listed in Chapter 2. The technical properties are covered first, followed by the properties related to user experience. As mentioned in Hartikainen, Miettinen, and Klamroth (2019), the properties are introduced for navigation methods for computationally expensive problems. The properties are relevant to all navigation methods, although properties concerning compu-

tational efficiency are more relevant when it comes to computationally expensive problems. The methods are analyzed based on whether they fulfill these desirable properties and how.

4.2.1 Technical properties

1. Navigation is complete. This property means that the DM can reach any feasible Pareto optimal solution from any current solution by giving appropriate preference information (Hartikainen, Miettinen, and Klamroth 2019). Table 2 lists the methods and how they fulfill the property. The developers of Pareto navigation (Monz et al. 2008), Bortz et al.'s (2014) method, Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019), and NAUTILUS Navigator (Ruiz et al. 2019), have stated in their respective papers that their methods fulfill the property. The DM can reach any feasible Pareto optimal solution by controlling the navigation using the graphical user interfaces developed for the methods.

In Pareto navigation (Monz et al. 2008), any Pareto optimal solution can be found by performing certain actions of restriction and selection, which is proven in (Monz 2006). Bortz et al. (2014) state, that in their method, the entire Pareto front is navigable to the DM by moving the sliders in the graphical user interface, which is said to be made possible by using linear interpolation, though not specified how. Hartikainen, Miettinen, and Klamroth (2019) introduce e-cones for Nonconvex Pareto Navigator that help make the navigation complete even if there are disconnected parts in the approximation of the Pareto front. The completeness of navigation when using Nonconvex Pareto Navigator is proven by utilizing the e-cones and assuming the approximation is a closed set in Hartikainen, Miettinen, and Klamroth (2019). As stated in Ruiz et al. (2019), any feasible Pareto optimal solution is reachable for the DM by directing the navigation process using NAUTILUS Navigator.

Pareto Race (Korhonen and Wallenius 1988), Pareto Navigator (Eskelinen et al. 2010), iSOM-Pareto Race (Yadav, Ramu, and Deb 2022), and RINADA (Baldan et al. 2023) can also be seen to fulfill the property. In Pareto Race, the navigation set is the actual Pareto front, which means that any feasible Pareto optimal solution is reachable during the navigation. This is possible as Pareto Race is only applicable to linear multiobjective optimization problems. For iSOM-Pareto Race, Yadav, Ramu, and Deb (2022) state, that the idea is to

use the original Pareto Race idea and navigate on the Pareto front. In Pareto Navigator and RINADA, the navigation set is a convex hull approximation constructed based on a set of precomputed Pareto optimal solutions. Any solution in a convex hull approximation may be reached.

In methods for which it is not clear whether any feasible Pareto optimal solution may be reached, the completeness of the navigation may depend on, for example, the accuracy and representativeness of the approximation of the Pareto front. Regarding multiple Pareto surface navigation (Craft and Monz 2010) and patch navigation (Collicott et al. 2021) it is not specified whether any feasible Pareto optimal solution can be found through the navigation process. Though, in these methods, navigation happens on convex fronts which could make any feasible solution reachable during navigation. The DM navigates on multiple convex fronts which could restrict some areas from being navigated to.

The methods considered in this thesis, that can be seen to not fulfill this property, are Lin and Ehrgott's (2018) method and O-NAUTILUS (Saini et al. 2022). In the method by Lin and Ehrgott (2018), the DM navigates in a set of precomputed Pareto optimal solutions, which means that any Pareto optimal solution that is not included in the set of precomputed solutions is not reachable during the navigation process. O-NAUTILUS uses surrogate models to make the method applicable to problems that may be too difficult for other methods to handle. As mentioned in Saini et al. (2022), using surrogates may lead to some Pareto optimal solutions not being reachable by the DM in some cases.

Method	How the navigation is made complete
Pareto Race (Korhonen and Wallenius 1988)	Navigation happens on the Pareto front and the whole Pareto front is navigable.
Pareto navigation (Monz et al. 2008)	With an algorithm described and proven in Monz (2006).
Multiple Pareto surface navigation (Craft and Monz 2010)	Not specified whether navigation is complete.
Pareto Navigator (Eskelinen et al. 2010)	Navigation set is a convex hull approximation in which any solution can be reached.
Bortz et al.'s (2014) method	Linear interpolation is used which is said to make it possible to reach any Pareto optimal solution (Bortz et al. 2014), though it is not specified how.
Lin and Ehrgott's (2018) method	Does not fulfill the property.
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	Proof provided in Hartikainen, Miettinen, and Klamroth (2019).
NAUTILUS Navigator (Ruiz et al. 2019)	As the DM does not move on the approximation, any solution is reachable with some preference information or by moving backwards to have more options.
Patch navigation (Collicott et al. 2021)	Not specified whether navigation is complete.
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	Navigation happens on the Pareto front by using an evolutionary algorithm RD-NSGA-II by Deb and Kumar (2007) with an achievement scalarizing function, which makes it possible to reach any feasible Pareto optimal solution.
O-NAUTILUS (Saini et al. 2022)	Does not fulfill the property.
RINADA (Baldan et al. 2023)	Navigation set is a convex hull approximation in which any solution can be reached.

Table 2. How the methods fulfill the "navigation is complete" property.

2. Navigation is computationally efficient. This means that, if the original problem is computationally expensive, using an approximation of the Pareto front as the navigation set should make the computation of solutions computationally less expensive i.e., faster (Hartikainen, Miettinen, and Klamroth 2019). While making the navigation more efficient to support real-time navigation, navigation in an approximated set makes the navigation less accurate. Table 3 lists the methods and how they fulfill the property. All of the methods fulfill the property, as shown in Table 3. However, for some methods, restrictions on the number of objective functions and decision variables are mentioned.

As shown in Table 3, in many methods simple linear problems are solved to compute each successive solution during the navigation process. This makes the navigation computationally more efficient than what solving the original multiobjective optimization problems would be. In Pareto Race (Korhonen and Wallenius 1988), the problems that are being optimized are linear themselves, so computing each solution is already computationally efficient. In Pareto navigation, linear problems are solved for each action of selection and restriction (Monz et al. 2008). Similarly to Pareto navigation, in patch navigation, linear problems are also solved for each action (selection, restriction and patch (de)activation) the DM makes (Collicott et al. 2021). Regarding patch navigation, it is mentioned, that for 15 or fewer objectives and 50 or fewer patches (convex Pareto fronts), the navigation should be at least close to real-time (Collicott et al. 2021). In multiple Pareto surface navigation, linear problems are solved to navigate on a single Pareto front and between multiple Pareto fronts (Craft and Monz 2010). In the method by Bortz et al. (2014), a linear problem is solved to compute each solution during the navigation.

In other methods, some other simpler problems are solved during the navigation to make the navigation computationally more efficient. In Pareto Navigator, a parametric linear optimization problem is solved to compute the next approximated solution (Eskelinen et al. 2010). Similarly, in Lin and Ehrgott's (2018) method, linear optimization problems are solved during the navigation (Lin and Ehrgott 2018). In Nonconvex Pareto Navigator, the navigation happens by solving mixed integer linear optimization problems to compute each successive solution (Hartikainen, Miettinen, and Klamroth 2019). In RINADA (Baldan et al. 2023), a nonlinear problem is solved to compute the next solution in both input and output naviga-

tion. Baldan et al. (2023) state, that for RINADA, seven decision variables and five objective functions are the highest numbers for the navigation to be real-time. Beyond those numbers, the method would not be computationally efficient enough. NAUTILUS Navigator (Ruiz et al. 2019) and O-NAUTILUS (Saini et al. 2022) do not use the original problem to compute the next solution during the navigation process. Therefore, the computational efficiency of the navigation is not affected by the complexity of the original problem.

Although for iSOM-Pareto Race (Yadav, Ramu, and Deb 2022), it is not clear how the method supports real-time navigation, it is categorized here to fulfill the property. In iSOM-Pareto Race, an evolutionary algorithm, RD-NSGA-II (Deb and Kumar 2007), is used with an achievement scalarizing function to compute each successive solution (Yadav, Ramu, and Deb 2022). Although it is not specified how the evolutionary algorithm makes the navigation real-time, it is stated in Deb and Kumar (2007) that RD-NSGA-II is computationally efficient, though no computation times are provided.

Method	How the navigation is made computationally efficient
Pareto Race (Korhonen and Wallenius 1988)	Applicable to linear problems only, so only linear problems are solved during navigation.
Pareto navigation (Monz et al. 2008)	Linear problems are solved for each action of selection and restriction.
Multiple Pareto surface navigation (Craft and Monz 2010)	Linear problems are solved to navigate on a single Pareto front, as well as between multiple Pareto fronts.
Pareto Navigator (Eskelinen et al. 2010)	A parametric linear optimization problem is solved to compute each successive solution.
Bortz et al.'s (2014) method	A linear problem is solved to compute each successive solution during the navigation.
Lin and Ehrgott's (2018) method	Linear optimization problems are solved during the navigation.
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	Mixed integer linear optimization problems are solved to compute each successive solution.
NAUTILUS Navigator (Ruiz et al. 2019)	A simple formula is solved to compute each successive solution.
Patch navigation (Collicott et al. 2021)	A linear problem is solved for each action of selection, restriction and patch (de)activation.
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	RD-NSGA-II (Deb and Kumar 2007) is used with an achievement scalarizing function to compute each successive solution.
O-NAUTILUS (Saini et al. 2022)	A simple formula is solved to compute each successive solution.
RINADA (Baldan et al. 2023)	A nonlinear problem is solved to compute each successive solution in both input and output navigation, which makes the navigation happen in real-time with up to seven decision variables.

Table 3. How the methods fulfill the "navigation is computationally efficient" property.

3. Construction of the navigation set is computationally efficient. This means, that although the construction of the navigation set can be, and often is, done offline before the involvement of the DM, the construction should not take long (Hartikainen, Miettinen, and Klamroth 2019). Table 4 lists the methods and how they fulfill the property. Some methods have similar features that have been implemented to fulfill the property. These similar features have been numbered in Table 4 to emphasize the similarities and avoid repetition. As shown in Table 4, almost all of the methods can fulfill this property. For some methods, the computational efficiency may depend on some a posteriori method used to generate the approximation. There are two exceptions: Pareto Race (Korhonen and Wallenius 1988) and iSOM-Pareto Race (Yadav, Ramu, and Deb 2022). As mentioned, in Pareto Race, the navigation set is the actual Pareto front and each solution in the set is calculated during the navigation. For iSOM-Pareto Race it is not clear what the actual navigation set is and how it is generated.

As can be seen in Table 4, some methods assume an approximation of the Pareto front as the navigation set, so the computational efficiency of constructing them depends on the method used to generate the approximation. Choosing a method that makes the construction computationally more efficient means sacrificing the accuracy of the approximation. The navigation set in Pareto navigation is a set of precomputed Pareto optimal solutions and their convex combinations (Monz et al. 2008). Similar to Pareto navigation, in multiple Pareto surface navigation, the navigation set consists of precomputed solutions and linear combinations of them (Craft and Monz 2010). Computing the linear combinations should not take long so the construction of the navigation sets is computationally efficient.

In Pareto Navigator, the navigation set is a polyhedral approximation of the Pareto front, which is generated utilizing a small set of Pareto optimal solutions (Eskelinen et al. 2010). Eskelinen et al. (2010) do not make a distinction on what method to use to generate the approximation for Pareto Navigator, so the computational efficiency may vary according to the method used. This is also the case in NAUTILUS Navigator (Ruiz et al. 2019): a set of precomputed Pareto optimal solutions that approximate the Pareto front is assumed, so the computational efficiency depends on the a posteriori method used to generate the approximation. O-NAUTILUS considers two sets during the navigation (Saini et al. 2022). The set of

known solutions is the same as in NAUTILUS Navigator, and the computational efficiency of constructing it depends on the method used. The other set, an optimistic estimation of the Pareto front, is formed using trained surrogate models. The computational efficiency of the construction of the optimistic estimation depends on the surrogate model technique and the multiobjective optimization method used to compute it. But, by default, surrogate models are assumed to be computationally less expensive than the original functions.

In patch navigation (Collicott et al. 2021) and RINADA (Baldan et al. 2023), the navigation set is an approximated convex hull. In patch navigation, for each Pareto front, a convex hull approximation is generated using some method, for example, a sandwiching algorithm (Collicott et al. 2021). The computational efficiency depends on the method used to approximate the different Pareto fronts. In RINADA, the generation of the approximation has two steps that are described in Baldan et al. (2023). In Baldan et al. (2023), there is a table, that shows the time it takes to generate a convex hull in different dimensions. With up to seven decision variables, the generation can still happen in seconds (Baldan et al. 2023).

In Bortz et al.'s (2014) method, to generate an approximation as the navigation set, a method introduced in Bortz et al. (2014) is used. The method uses sandwiching and hyperboxing algorithms to approximate the Pareto front, which is said to be efficient (Bortz et al. 2014), though no computation times are provided. In Lin and Ehrgott's (2018) method, the navigation set is a discrete set of solutions constructed using column generation, as described in Lin, Ehrgott, and Raith (2017). In Lin, Ehrgott, and Raith (2017), a test is conducted on the method and some computation times are shown in a table. Using a discrete set (a database) of solutions is also said to reduce computational expenses (Lin and Ehrgott 2018), though it is not mentioned how. For Nonconvex Pareto Navigator, the navigation set is constructed using the PAINT method and e-cones (Hartikainen, Miettinen, and Klamroth 2019). The computational cost comes from the construction of the approximation using the PAINT method, which is shown to take a relatively short time in Hartikainen, Miettinen, and Wiecek (2012).

Method	How the construction of the navigation set is made computationally efficient
Pareto Race (Korhonen and Wallenius 1988)	The property is not relevant for the method.
Pareto navigation (Monz et al. 2008)	Feature 3.1: Construction consists of computing linear combinations of the precomputed Pareto optimal solutions.
Multiple Pareto surface navigation (Craft and Monz 2010)	Feature 3.1.
Pareto Navigator (Eskelinen et al. 2010)	Depends on the method used to generate the convex hull approximation, but this should not take long.
Bortz et al.'s (2014) method	A method that uses sandwiching and hyperboxing algorithms is used.
Lin and Ehrgott's (2018) method	A discrete set of solutions is constructed using column generation, as explained in Lin, Ehrgott, and Raith (2017).
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	The computational cost comes from the construction of the approximation using the PAINT method, which is shown to take a relatively short time in Hartikainen, Miettinen, and Wiecek (2012).
NAUTILUS Navigator (Ruiz et al. 2019)	Feature 3.2: Depends on the method used to generate the approximation used as the navigation set.
Patch navigation (Collicott et al. 2021)	Feature 3.2.
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	The property is not relevant for the method.
O-NAUTILUS (Saini et al. 2022)	Feature 3.2.
RINADA (Baldan et al. 2023)	A two-step method is used to generate the convex hull approximation, that is described in Baldan et al. (2023)

Table 4. How the methods fulfill the "construction of the navigation set is computationally efficient" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.

4. Accuracy of the navigation set can be measured. In many cases, the navigation set is an approximation of the Pareto front so the accuracy of it should be measurable (Hartikainen, Miettinen, and Klamroth 2019). This accuracy information can then be used to decide when the navigation set is accurate enough for the navigation process and can also be shown to the DM if needed. Table 5 lists the methods and how they fulfill the property. Some methods have similar features that have been implemented to fulfill the property. These similar features have been numbered in Table 5 to emphasize the similarities and avoid repetition. In Pareto Navigator (Eskelinen et al. 2010), the DM may observe how accurate the navigation set is when asking for an approximated solution to be projected onto the Pareto front.

The methods by Bortz et al. (2014) and Lin and Ehrgott (2018), Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019), and RINADA (Baldan et al. 2023) have a way of measuring the accuracy of the navigation set. In the method by Bortz et al. (2014), the algorithm used to generate the approximation allows the DM to set a desired accuracy for the approximation of the Pareto set. The accuracy of the approximation is therefore checked while generating the approximation. The quality of the discrete navigation set used in Lin and Ehrgott's (2018) method can be measured, as described in Lin, Ehrgott, and Raith (2017), for example, by calculating the distance of the representation to the Pareto front. The navigation set in Nonconvex Pareto Navigator is a combination of a PAINT approximation and e-cones. The accuracy of the PAINT approximation can be measured with an error vector (Hartikainen, Miettinen, and Klamroth 2019). For RINADA, there is a test to be conducted on the approximated convex hull, that is described in Baldan et al. (2023), which checks the error of the approximation in comparison to the exact convex hull. Baldan et al. (2023) also introduce a test to see whether the solutions navigated to are too far from the actual data set.

For some methods, whether or not the accuracy of the navigation set can be measured, depends on the method used to construct the navigation set. This is the case for NAUTILUS Navigator (Ruiz et al. 2019) and O-NAUTILUS (Saini et al. 2022). Whether or not the approximation is accurate enough may still be observed during the solution process as the final solution in the navigation may be projected onto the Pareto front. From the DM's perspective, if the projected Pareto optimal solution is far from the found approximated solution, then the accuracy of the approximation should be improved.

A way of measuring the accuracy of the navigation set either does not exist or it just is not mentioned regarding Pareto navigation (Monz et al. 2008), multiple Pareto surface navigation (Craft and Monz 2010), patch navigation (Collicott et al. 2021), and iSOM-Pareto Race (Yadav, Ramu, and Deb 2022). For Pareto navigation and multiple Pareto surface navigation, no way of measuring the accuracy of the navigation set is presented. However, in both cases, it is mentioned that the more accurately the precomputed Pareto optimal solutions approximate the Pareto front, the more accurate and representative the navigation set will be. For patch navigation, the accuracy of the navigation set and whether it can be measured depends on the method used to generate the approximation. There is no mention of the possibility of measuring the accuracy of the representation of the Pareto front in iSOM-Pareto Race.

Method	How the accuracy of the navigation set can be measured
Pareto Race (Korhonen and Wallenius 1988)	The property is not relevant for the method.
Pareto navigation (Monz et al. 2008)	Does not fulfill the property.
Multiple Pareto surface navigation (Craft and Monz 2010)	Does not fulfill the property.
Pareto Navigator (Eskelinen et al. 2010)	Feature 4.1: The DM may observe how accurate the navigation set is when asking for an approximated solution to be projected onto the Pareto front.
Bortz et al.'s (2014) method	The algorithm used to generate the approximation checks the accuracy during the generation.
Lin and Ehrgott's (2018) method	The quality of the navigation set can be measured, as described in Lin, Ehrgott, and Raith (2017), for example, by calculating the distance of the representation to the Pareto front.
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	The accuracy of the PAINT approximation used in the construction of the navigation set can be measured with an error vector (Hartikainen, Miettinen, and Wiecek 2012).
NAUTILUS Navigator (Ruiz et al. 2019)	Feature 4.1.
Patch navigation (Collicott et al. 2021)	Does not fulfill the property
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	The property is not relevant for the method.
O-NAUTILUS (Saini et al. 2022)	Feature 4.1.
RINADA (Baldan et al. 2023)	A test that checks the error of the approximated convex hull in comparison to the exact convex hull.

Table 5. How the methods fulfill the "accuracy of the navigation set can be measured" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.

5. Accuracy of the navigation set can be improved. This means that in the event that the navigation set is not accurate enough, the accuracy can be improved (Hartikainen, Miettinen, and Klamroth 2019). When the DM finds an interesting approximated solution and wants to project it onto the Pareto front, and if that projected Pareto optimal solution is not satisfactory as the final solution, that projected solution should be added to the navigation set which improves the accuracy. Table 6 lists the methods and how they fulfill the property. Some methods have similar features that have been implemented to fulfill the property. These similar features have been numbered in Table 6 to emphasize the similarities and avoid repetition. For patch navigation (Collicott et al. 2021), the accuracy of the navigation set and whether it can be improved depends on the method used to generate the approximation. There is no mention of a possibility to improve the accuracy of the representation of the Pareto front regarding iSOM-Pareto Race (Yadav, Ramu, and Deb 2022) and RINADA (Baldan et al. 2023).

As can be seen in Table 6, in many methods, the accuracy of the navigation set may be improved by generating more Pareto optimal solutions and using them to regenerate the approximation of the Pareto front. In Pareto navigation (Monz et al. 2008) and multiple Pareto surface navigation (Craft and Monz 2010), a set of precomputed Pareto optimal solutions is assumed, from which the navigation set is constructed. The accuracy of the navigation set may be improved by generating more Pareto optimal solutions that better represent the Pareto front. This is also the case in Pareto Navigator: the accuracy of the polyhedral approximation may be improved by adding more Pareto optimal solutions to generate the approximation (Eskelinen et al. 2010). For Nonconvex Pareto Navigator, the accuracy of the navigation set can be improved by adding more Pareto optimal solutions to generate the PAINT approximation (Hartikainen, Miettinen, and Klamroth 2019). Adding more Pareto optimal solutions to improve the accuracy of the navigation set in Pareto Navigator and Nonconvex Pareto Navigator means the approximations need to be regenerated.

In NAUTILUS Navigator (Ruiz et al. 2019), new Pareto optimal solutions may be generated, and the accuracy of the approximation may be improved within the current reachable ranges (Ruiz et al. 2019). To avoid regenerating the whole approximation, the Pareto fill module, introduced in Ruiz et al. (2015), can be used to generate more solutions within the reachable

ranges. In O-NAUTILUS (Saini et al. 2022), the DM can conduct exact function evaluations in interesting regions (Saini et al. 2022). The results of the evaluations can then be added to the set of known solutions. The surrogate models used to compute the optimistic ranges can then be retrained which makes the optimistic ranges more accurate in that interesting region.

The methods by Bortz et al. (2014) and Lin and Ehrgott (2018) can improve the accuracy of the navigation set without needing more Pareto optimal solutions to be computed. In both methods, if the DM wants to improve the accuracy during the navigation process, the navigation set has to be regenerated. In the method by Bortz et al. (2014), the accuracy of the approximated navigation set is checked during the generation of the approximation. The accuracy of the approximation can be improved while generating (or regenerating) the approximation by setting a desired accuracy for the approximation, which can be set by the DM. The quality of the discrete navigation set used in Lin and Ehrgott's (2018) method may also be improved by changing the parameters used to construct the navigation set, as described in Lin, Ehrgott, and Raith (2017).

Method	How the accuracy of the navigation set can be improved
Pareto Race (Korhonen and Wallenius 1988)	The property is not relevant for the method.
Pareto navigation (Monz et al. 2008)	Feature 5.1: Generating more Pareto optimal solutions to add to the set of precomputed solutions from which the navigation set is constructed.
Multiple Pareto surface navigation (Craft and Monz 2010)	Feature 5.1.
Pareto Navigator (Eskelinen et al. 2010)	Feature 5.1.
Bortz et al.'s (2014) method	The DM can set a desired accuracy a priori.
Lin and Ehrgott's (2018) method	Changing the parameters used to construct the navigation set.
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	Feature 5.1.
NAUTILUS Navigator (Ruiz et al. 2019)	The Pareto fill module, introduced in Ruiz et al. (2015), can be used to generate more solutions within the reachable ranges.
Patch navigation (Collicott et al. 2021)	Does not fulfill the property
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	The property is not relevant for the method.
O-NAUTILUS (Saini et al. 2022)	The DM can conduct exact function evaluations in interesting regions, and the surrogate models used to compute the optimistic ranges can then be retrained.
RINADA (Baldan et al. 2023)	Does not fulfill the property.

Table 6. How the methods fulfill the "accuracy of the navigation set can be improved" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.

4.2.2 Properties related to user experience

Hartikainen, Miettinen, and Klamroth (2019) mention, that the properties related to user experience may be more difficult to measure, and whether a method fulfills them may depend on the DM that uses the method. The analysis done in this subsection relies completely on the information given in the respective papers as implementations are not readily available to be tested from a DM's perspective, and the functionality and usability of the user interfaces play an important role in assessing the user experience.

6. The DM can control the navigation. This property means that the method should allow the DM to control the navigation (Hartikainen, Miettinen, and Klamroth 2019). To see if a method fulfills the property, it can be checked whether the DM can reach some specific values by navigating or restrict some values they do not want to navigate to (Hartikainen, Miettinen, and Klamroth 2019). The implementations for the methods that are introduced in the considered papers all offer the DM ways to control the navigation. This property is therefore fulfilled by the methods. Table 7 lists the methods and how they fulfill the property. Some methods have similar features that have been implemented to fulfill the property. These similar features have been numbered in Table 7 to emphasize the similarities and avoid repetition.

In Pareto Race (Korhonen and Wallenius 1988) and iSOM-Pareto Race (Yadav, Ramu, and Deb 2022), the DM controls the navigation by using the navigation controls mentioned in Section 4.1 regarding Pareto Race. The DM can use the controls by pressing the corresponding keyboard keys. If the DM has in mind some specific values, they can reach them by moving around the Pareto front using the provided controls. The DM can also restrict certain areas by fixing some objective values which adds those values as constraints to the problem that is solved during the navigation. However, the DM can only set an aspiration level for one objective function at a time.

In multiple Pareto surface navigation (Craft and Monz 2010), the DM navigates by pressing buttons in the graphical user interface shown in Figure 3. The buttons are up, down, switch and lock. The up and down buttons increase and decrease the value of the corresponding objective, and the lock button locks the corresponding objective value so that it does not

change while moving to other solutions. The DM can press the switch button to switch the Pareto front, that is being navigated on, for a Pareto front that has a solution that improves the corresponding objective value. There is no other way of restricting objective values besides the restrictions coming from setting a locked objective value as a constraint and switching between the Pareto fronts.

In Pareto Navigator (Eskelinen et al. 2010), Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019), NAUTILUS Navigator (Ruiz et al. 2019), and O-NAUTILUS (Saini et al. 2022), the DM can control the navigation by giving preference information by moving lines corresponding to their aspiration levels and bounds for the objectives or by typing in the corresponding text boxes. The DM can therefore set the specific values they want to reach as the aspiration levels and navigate to those values if they are feasible. These methods have similar implementations in regard to providing and visualizing the preference information. In all of these methods, the DM may provide their preference information by typing them into the text boxes corresponding to the objectives. A graphical user interface for Pareto Navigator was suggested by Tarkkanen et al. (2013) (Figure 4), in which the DM can also give aspiration levels (or a reference point) for the objectives by clicking on the panel in between two Pareto optimal solutions or vertical lines displayed. The example graphical user interface for Nonconvex Pareto Navigator, and the graphical user interfaces for NAUTILUS Navigator and O-NAUTILUS (Figures 7, 12 and 13), on the other hand, suggest changing preference information by moving the horizontal lines representing the aspiration levels and bounds.

The DM can control the navigation by moving sliders corresponding to the objective functions in the graphical user interface in Pareto navigation (Monz et al. 2008), Bortz et al.'s (2014) method, Lin and Ehrgott's (2018) method, Patch navigation (Collicott et al. 2021), and RINADA (Baldan et al. 2023). By moving sliders corresponding to the objectives, the DM can set the aspiration level of one objective at a time. The graphical user interfaces (Figures 2, 5, 6, 8, and 11) for these methods also allow the DM to set bounds for the objectives. Pareto navigation has two actions the DM can perform to control the navigation by moving sliders in the graphical user interface. The first action is restriction which allows the DM to set constraints to the objectives, the other is selection which allows the DM to set an aspi-

ration level for one objective at a time. Patch navigation utilizes the two actions of Pareto navigation while adding a third action called patch (de)activation. In patch (de)activation, the DM includes or excludes a certain patch (part of the Pareto front) from being navigated to. Restriction and selection can be done by moving the sliders while patch (de)activation happens by checking and unchecking checkboxes. In RINADA, it is possible to control the navigation by moving sliders corresponding to the decision variables (input navigation) or the objectives (output navigation).

Method	How the DM can control the navigation
Pareto Race (Korhonen and Wallenius 1988)	Feature 6.1: Navigation can be controlled by pressing keyboard keys.
Pareto navigation (Monz et al. 2008)	Feature 6.2: Moving sliders corresponding to the objectives and possibly decision variables.
Multiple Pareto surface navigation (Craft and Monz 2010)	Pressing up, down, switch and lock buttons.
Pareto Navigator (Eskelinen et al. 2010)	Feature 6.3: Moving lines corresponding to the aspiration levels and bounds or by typing into the corresponding text boxes.
Bortz et al.'s (2014) method	Feature 6.2.
Lin and Ehrgott's (2018) method	Feature 6.2.
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	Feature 6.3.
NAUTILUS Navigator (Ruiz et al. 2019)	Feature 6.3.
Patch navigation (Collicott et al. 2021)	Feature 6.2.
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	Feature 6.1.
O-NAUTILUS (Saini et al. 2022)	Feature 6.3.
RINADA (Baldan et al. 2023)	Feature 6.2.

Table 7. How the methods fulfill the "the DM can control the navigation" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.

7. Low cognitive load is set on the DM. This property means that using a method should set as little cognitive load on the DM as possible (Hartikainen, Miettinen, and Klamroth 2019). To achieve this, the navigation process should be made intuitive for the DM and the possible visualization of the method should be understandable to the DM (Hartikainen, Miettinen, and Klamroth 2019). Table 8 lists the methods and how they fulfill the property. Some methods have similar features that have been implemented to fulfill the property. These similar features have been numbered in Table 8 to emphasize the similarities and avoid repetition. There is no mention of the visualization technique regarding multiple objectives for Bortz et al.'s (2014) method. However, the navigation itself happens by moving sliders which can be seen to be intuitive.

Pareto Race (Korhonen and Wallenius 1988), patch navigation (Collicott et al. 2021) and iSOM-Pareto Race (Yadav, Ramu, and Deb 2022) can be seen to not fulfill this property. The graphical user interface used to visualize the navigation in Pareto Race consists of bars representing the objective values, as seen in Figure 1. It is stated in Tarkkanen et al. (2013), that having to remember previous solutions makes comparing the solutions a cognitively demanding task. Therefore, even though the visualization that consists of bars is easy to understand, it makes comparing solutions more burdensome for the DM. In patch navigation, the graphical user interface (Figure 8) has a slider for each objective and navigation happens by moving these sliders. It is mentioned, that there may be some sudden jumps in the sliders during navigation, as the navigated patch changes, which could cause the DM to feel less in control (Collicott et al. 2021). The visualization for iSOM-Pareto Race, as shown in Figure 9, has color-coded solutions to inform the DM about the different solutions. The color-coded solutions, combined with a heat map displaying the values of the corresponding objective, form the display for one objective. These displays are formed for all of the objectives along with some other plots to give more information. This type of visualization may become more useful when a DM gets more familiar with the concept but at first may be taxing.

In Pareto navigation (Monz et al. 2008), multiple Pareto surface navigation (Craft and Monz 2010) and Lin and Ehrgott's (2018) method, the navigation itself can be seen to be intuitive and the visualizations understandable. As mentioned, comparing each new solution found during the navigation, while having to remember previous solutions, makes the task cogni-

tively more demanding. These methods are seen to still fulfill the property as their graphical user interfaces (Figures 2, 3 and 6) are understandable, and navigation using them is made intuitive. Regarding multiple Pareto surface navigation, it is mentioned, that switching Pareto fronts can cause sudden changes in objective values, which could cause the DM to feel less in control (Craft and Monz 2010). Therefore, the DM is given the option to switch Pareto fronts manually which lowers the cognitive load as the DM can decide if they want to switch fronts.

Pareto Navigator (Eskelinen et al. 2010), Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019), NAUTILUS Navigator (Ruiz et al. 2019), O-NAUTILUS (Saini et al. 2022) and RINADA (Baldan et al. 2023) show the DM the solutions previously navigated to. This helps the DM in comparing found solutions as they do not have to remember the previous solutions which lowers the cognitive load set on the DM. In addition, in Nonconvex Pareto Navigator, the DM may choose to let some objective values change freely during parts of the navigation which can lower the cognitive load on the DM (Hakanen, Sahlstedt, and Miettinen 2013). NAUTILUS Navigator and O-NAUTILUS show the DM all the previous reachable ranges and the DM can see the ranges shrinking during the navigation process. Showing the DM the previous reachable ranges as well is said to lower the cognitive load for the DM (Ruiz et al. 2019). In addition to the reachable ranges of NAUTILUS Navigator, O-NAUTILUS also displays the optimistic ranges to the DM which does add some cognitive load for the DM. In RINADA, previous solutions navigated to are displayed as different colors to other solutions which makes it easier to compare already navigated solutions which lowers the cognitive load set on the DM. However, in RINADA, all the solutions in the navigation set are displayed at once, which makes it more difficult to compare them.

Method	How a low cognitive load is set on the DM
Pareto Race (Korhonen and Wallenius 1988)	Does not fulfill the property.
Pareto navigation (Monz et al. 2008)	Feature 7.1: Moving sliders to navigate can be seen as intuitive, and the visualization is understandable.
Multiple Pareto surface navigation (Craft and Monz 2010)	Pressing up, down, switch and lock buttons can be seen as intuitive, and the visualization is understandable.
Pareto Navigator (Eskelinen et al. 2010)	Feature 7.2: Displays solutions previously navigated to, making comparing solutions a less demanding task.
Bortz et al.'s (2014) method	Does not fulfill the property
Lin and Ehrgott's (2018) method	Feature 7.1.
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	Feature 7.2.
NAUTILUS Navigator (Ruiz et al. 2019)	Feature 7.2.
Patch navigation (Collicott et al. 2021)	Does not fulfill the property
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	Does not fulfill the property.
O-NAUTILUS (Saini et al. 2022)	Feature 7.2.
RINADA (Baldan et al. 2023)	Feature 7.2.

Table 8. How the methods fulfill the "low cognitive load is set on the DM" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.

8. The DM is allowed to learn. The methods should support the DM in learning about the problem and the reachable solutions. The DM should also be able to change their mind and take steps backwards or go to a solution that has already been passed in the navigation process (Hartikainen, Miettinen, and Klamroth 2019). Table 9 lists the methods and how they fulfill the property. Some methods have similar features that have been implemented to fulfill the property. These similar features have been numbered in Table 9 to emphasize the similarities and avoid repetition.

Pareto navigation (Monz et al. 2008), Bortz et al.'s (2014) method, Lin and Ehrgott's (2018) method, patch navigation (Collicott et al. 2021), and RINADA (Baldan et al. 2023) have sliders that the DM uses to control the navigation. These methods, that feature sliders for the DM to control the navigation, support the DM's learning as the DM may always move the sliders back to a previous position. Once they have taken the steps backwards, the DM can then try moving some other slider to see if they can get different results, which allows the DM to change their mind and learn about the problem. However, as mentioned regarding patch navigation, there may be some sudden jumps in the sliders during navigation which could make it more difficult for the DM to compare solutions and learn from them as they may be far away from each other.

Pareto Race (Korhonen and Wallenius 1988), iSOM-Pareto Race (Yadav, Ramu, and Deb 2022) and multiple Pareto surface navigation (Craft and Monz 2010) feature controls to support the DM's learning. The DM can use the "gear" controls, featured in Pareto Race and iSOM-Pareto Race, and change the movement direction from forwards to backwards and vice versa. By using the gears, the DM can go back to a previously passed solution and then use the "turn" control to change the movement direction. In multiple Pareto surface navigation, the DM can learn about the problem and reachable solutions by pressing buttons up and down, and reverse each action if they feel the solutions are getting less desirable. The DM can also change their mind by switching the Pareto front that is being navigated on at any point.

Pareto Navigator (Eskelinen et al. 2010), Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019), NAUTILUS Navigator (Ruiz et al. 2019), and O-NAUTILUS (Saini et al. 2022) show the DM the passed solutions and allow the DM to move backwards

to any previously passed solution. As the DM is shown the solutions that have been navigated to earlier, it is easier for the DM to see if the solutions are developing towards more desirable values. If not, then the DM can always pause the navigation and go back to some passed solution and then change their preferences to move in some other direction. This allows the DM to learn and change their mind. The DM can go backwards to a passed solution by clicking on the solution on the graphical user interface. The graphical user interfaces for Pareto Navigator and Nonconvex Pareto Navigator, shown in Figures 4 and 7, also feature different tabs for different directions, so the DM can also compare solutions found from different areas.

Method	How the DM is allowed to learn
Pareto Race (Korhonen and Wallenius 1988)	Feature 8.1: Controls that allow to take steps backwards and turn.
Pareto navigation (Monz et al. 2008)	Feature 8.2: Sliders that control the navigation can be moved back to a previous position.
Multiple Pareto surface navigation (Craft and Monz 2010)	Pressing up, down, switch and lock buttons can be reversed at any point to go back to a previous solution.
Pareto Navigator (Eskelinen et al. 2010)	Feature 8.3: Shows the DM the passed solutions and allows the DM to move backwards to a previously passed solution.
Bortz et al.'s (2014) method	Feature 8.2.
Lin and Ehrgott's (2018) method	Feature 8.2.
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	Feature 8.3.
NAUTILUS Navigator (Ruiz et al. 2019)	Feature 8.3.
Patch navigation (Collicott et al. 2021)	Feature 8.2.
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	Feature 8.1.
O-NAUTILUS (Saini et al. 2022)	Feature 8.3.
RINADA (Baldan et al. 2023)	Feature 8.2.

Table 9. How the methods fulfill the "the DM is allowed to learn" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.

9. The DM can get additional information of the navigation set. This property means that the DM should receive some additional information during the navigation process in addition to each solution found (Hartikainen, Miettinen, and Klamroth 2019). The additional information could be, for example, the reachable objective values in the navigation set from the current solution. Table 10 lists the methods and how they fulfill the property. Some methods have similar features that have been implemented to fulfill the property. These similar features have been numbered in Table 10 to emphasize the similarities and avoid repetition. There is no mention of additional information about the navigation set shown to the DM regarding Pareto Race (Korhonen and Wallenius 1988) and multiple Pareto surface navigation (Craft and Monz 2010).

The graphical user interfaces for Pareto navigation (Monz et al. 2008), Lin and Ehrgott's (2018) method, NAUTILUS Navigator (Ruiz et al. 2019), patch navigation (Collicott et al. 2021), O-NAUTILUS (Saini et al. 2022), and RINADA (Baldan et al. 2023) show the DM the current reachable objective values in the navigation set. Lin and Ehrgott's method (Figure 6) also informs the DM whether a certain navigation step is feasible or not. Patch navigation (Figure 8) also tells the DM which patch they are currently navigating on and whether they are navigating on the Pareto front. NAUTILUS Navigator and O-NAUTILUS (Figures 12 and 13) also show the reachable ranges from the already passed solutions. In addition, O-NAUTILUS displays the optimistic reachable ranges for the objectives. In RINADA, the visualization (Figure 11) shows the data set to the DM which could make it harder to compare solutions as the whole set is shown at once.

Pareto Navigator (Eskelinen et al. 2010), Bortz et al.'s (Bortz et al. 2014) method, Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019), and iSOM-Pareto Race (Yadav, Ramu, and Deb 2022) give some other additional information on the problem and its solutions that may not be about the navigation set. Pareto Navigator and Nonconvex Pareto Navigator, show the DM the set of precomputed Pareto optimal solutions at all times, which the DM can then use to restart the navigation from at any time. In addition, the DM may ask to project the current solution onto the Pareto front and see how the approximated solution compares to the Pareto optimal one. Bortz et al.'s method has sliders for both the decision variables and objectives that display the values, which may help the DM's decision-making.

The iSOM-Pareto Race visualization provides a lot of additional information for the DM, for example, the color-coded solutions displayed show the DM whether a solution is near a constraint.

In the next chapter, some of the features identified in this subsection are discussed. Some other perspectives that can be taken into account when preparing syntheses in the future, and other future research topics are suggested as well as other ways to facilitate conducting literature reviews and preparing syntheses.

Method	What additional information can the DM get of the navigation set?
Pareto Race (Korhonen and Wallenius 1988)	Does not fulfill the property.
Pareto navigation (Monz et al. 2008)	Feature 9.1: Shows the DM the current reachable values for the objectives and possibly some additional information.
Multiple Pareto surface navigation (Craft and Monz 2010)	Does not fulfill the property.
Pareto Navigator (Eskelinen et al. 2010)	Feature 9.2: Shows the set of precomputed Pareto optimal solutions.
Bortz et al.'s (2014) method	Shows both the current objective and decision variable values.
Lin and Ehrgott's (2018) method	Feature 9.1.
Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019)	Feature 9.2.
NAUTILUS Navigator (Ruiz et al. 2019)	Feature 9.1.
Patch navigation (Collicott et al. 2021)	Feature 9.1.
iSOM-Pareto Race (Yadav, Ramu, and Deb 2022)	A lot of additional information for the DM, for example, the color-coded solutions displayed show the DM whether a solution is near a constraint.
O-NAUTILUS (Saini et al. 2022)	Feature 9.1.
RINADA (Baldan et al. 2023)	Feature 9.1.

Table 10. How the methods fulfill the "the DM can get additional information of the navigation set" property. Some similar features to fulfill the property can be identified in the methods. These features have been numbered to emphasize the similarities and avoid repetition.

5 Discussion

Some similarities in terms of the implementations of the methods were identified in the synthesis. As can be seen in Table 3, in many methods, a simple linear optimization problem or some other simple problem is solved to compute each successive solution during the navigation process. Therefore, some general solvers to compute the solutions can be implemented and used in the implementation of multiple navigation methods. Also, as seen in Table 4, multiple methods assume a precomputed approximation or representation of the Pareto front as the navigation set or as a basis of creating the navigation set. Therefore, a general method to construct the navigation set can be implemented and then utilized in the implementations of different navigation methods.

Graphical user interfaces are crucial for navigation methods as they support the DM in decision-making. Graphical user interfaces have been a crucial part of the implementations of the methods since the first navigation method considered in this thesis, Pareto Race by Korhonen and Wallenius (1988). They allow the DM to control the navigation and, learn about the problem and the solutions that can be found by navigating. That is why it is important to understand which features make a graphical user interface appropriate for a certain DM. There is probably not one set of features that caters to every DM's needs, which is why it is important to conduct studies, as in the work by Tarkkanen et al. (2013), to learn about different possibilities for the implementations.

In the synthesis, some common graphical user interface features could be identified, that were utilized in different methods. Some of the similarities were related to how the DM controls the navigation. The common features, that can also be seen in Table 7, that were used in multiple methods' implementations were movable sliders and lines corresponding to the objective values and lower and upper bounds for the objectives, and also the option to provide aspiration levels for the objectives by typing them into the corresponding text boxes. One common feature that was used to lower the cognitive load set on the DM was displaying all the solutions the DM had navigated to previously. To allow the DM to learn about the problem and change their mind, some of the methods had an option for the DM to choose one of the previous solutions and to continue the navigation from there.

As mentioned in Chapter 3, the terminology used for navigation methods is not universal. For example, the term "navigation method" or, as sometimes used, "navigation-based method" is not always used when describing the methods. Not having universal terminology makes it more difficult to do literature reviews and mapping studies for two reasons. First, constructing search queries and conducting searches in databases are more tedious tasks when one does not necessarily know a set of keywords to find all the relevant papers. Second, in the literature, some terms may be used in different contexts, and they may have different meanings. For example, the term "navigation" is not solely used regarding navigation methods but also in regard to other interactive methods, which is why it would be beneficial to separate navigation methods from other interactive methods by standardizing the terminology.

Another way to facilitate reviewing and mapping navigation methods, as well as other optimization methods, would be to make implementations of the methods openly available for testing and reviewing. If the methods have already been implemented as they are presented in the papers, the implementation used to demonstrate the methods in the papers should be available. Having the implementations available for testing would make conducting a synthesis and comparing different methods in terms of, for example, the desirable properties by Hartikainen, Miettinen, and Klamroth (2019) easier and more comprehensive. As the implementations typically are not (openly) available, the tests conducted in the papers cannot be repeated and reproducing results can become very challenging, if not impossible. Many of the papers have used real-world problems to demonstrate the methods, which makes the results more interesting, but the results of those tests, and the steps the DM has taken, cannot be reproduced without the implementation used in the tests. Having implementations available would help in repeating and reproducing results reported in the respective papers to verify the methods' usefulness, for example.

Many of the methods considered in this thesis assume that the problems that are being solved are convex. Having to make an assumption about the types of solvable problems hinders the broader applicability of such methods. Therefore, in the future, it would be beneficial to develop navigation methods that can also handle nonconvex problems. Some newer navigation methods, e.g., Nonconvex Pareto Navigator by Hartikainen, Miettinen, and Klamroth (2019), and O-NAUTILUS by Saini et al. (2022), have been developed to be applicable to

nonconvex problems as well. When choosing the method for solving a problem, it would be beneficial to have methods that can handle all types of problems.

To cater to two of the phases of the solution process mentioned in Chapter 2, the learning and decision phase, it is also possible to use both a navigation method and another interactive method to find the most preferred solution (Eskelinen et al. 2010). A navigation method can be used to learn about the problem and the trade-offs involved, and to find an interesting area in the navigation set (learning phase). Some other interactive method can then be used to find the most preferred solution in that area. For example, Heikkinen et al. (2023) first used Nonconvex Pareto Navigator (Hartikainen, Miettinen, and Klamroth 2019) to find an interesting region, and a solution that was then projected onto the Pareto front. The projected solution was then used as a starting point for an interactive method, NIMBUS (Miettinen and Mäkelä 2006). NIMBUS was then used to find the most preferred solution.

The way and the form in which the DM gives their preference information is an important aspect to consider when developing and implementing navigation methods. Giving preference information should always be made as intuitive and seamless as possible for the DM. Different ways for giving preference information that have been implemented in the graphical user interfaces were discussed in Section 4.2. The methods considered in this thesis also have different forms the DM can give their preference information in. One option is to give an aspiration level for one objective while allowing the other objective values to change freely, or aspiration levels to all objectives. In some methods, the DM may give a classification that tells if the DM wants, for example, an objective value to improve or is willing to let an objective value get worse in order to improve some others. The methods should, if possible, offer the DM the option to choose in which form they would prefer to give their preference information.

6 Conclusions

A literature review was conducted on navigation methods for multiobjective optimization problems, and the search process and the results of the review were described in this thesis. The methods introduced in the papers found in the review were analyzed and compared to form a synthesis. The analysis and comparison were done according to the type of navigation set where the navigation takes place, and according to the desirable properties for navigation methods introduced by Hartikainen, Miettinen, and Klamroth (2019). It was determined whether each method can be seen to fulfill the properties or not, and if they were seen to fulfill the properties, it was elaborated how.

The aim of the literature review was to establish an understanding of the state of the art in navigation methods. 12 different methods that were considered navigation methods were found and analyzed. The synthesis formed based on the literature review aimed to find out what kind of navigation methods exist and what similarities they have. A set of features, that were used in the methods, was identified. In the set of features, some similarities were found between different methods. These similarities were considered in terms of implementing the methods as well as implementing and designing graphical user interfaces for the methods.

Due to time constraints, the synthesis is not as diverse and comprehensive as it could have been. Therefore, an interesting topic for future research could be to expand the synthesis formed in this thesis by analyzing and comparing the methods from different perspectives. Some of these different perspectives that could be further examined are discussed in Chapter 5. Also, the search queries could be improved which could lead to better results from the literature review. Therefore, further research into finding a set of keywords to form the search queries for the literature review could result in finding more methods that could be considered in future literature reviews.

This thesis has provided an overview of what kind of navigation methods exist. The identified features and similarities between the methods offer some insight into some of the components that are used to implement the methods. These pieces can be used to develop new methods in the future. The contributions of this thesis can be used to choose the appropriate

navigation method to support the DM's decision-making, and also as a foundation for future literature reviews and syntheses on navigation methods.

Bibliography

- Allmendinger, Richard, Matthias Ehrgott, Xavier Gandibleux, Martin Josef Geiger, Kathrin Klamroth, and Mariano Luque. 2017. “Navigation in Multiobjective Optimization Methods”. *Journal of Multi-Criteria Decision Analysis* 24 (1–2): 57–70.
- Baldan, Marco, Patrick Otto Ludl, Philipp Süß, Dominik Schack, Robin Schmidt, and Michael Bortz. 2023. “Real-Time Interactive Navigation on Input-Output Data Sets in Chemical Processes”. *Chemie-Ingenieur-Technik* 95 (7): 1028–1040.
- Bertsimas, Dimitris, Valentina Cacchiani, David Craft, and Omid Nohadani. 2013. “A Hybrid Approach to Beam Angle Optimization in Intensity-Modulated Radiation Therapy”. *Computers & Operations Research* 40 (9): 2187–2197.
- Bortz, Michael, Jakob Burger, Norbert Asprion, Sergej Blagov, Roger Böttcher, Uwe Nowak, Andreas Scheithauer, Richard Welke, Karl-Heinz Küfer, and Hans Hasse. 2014. “Multi-Criteria Optimization in Chemical Process Design and Decision Support by Navigation on Pareto Sets”. *Computers and Chemical Engineering* 60:354–363.
- Collicott, Cristina, Esther Bonacker, Ina Lammel, Katrin Teichert, Michal Walczak, and Philipp Süß. 2021. “Interactive Navigation of Multiple Convex Patches”. *Journal of Multi-Criteria Decision Analysis* 28 (5–6): 311–321.
- Craft, David, and Michael Monz. 2010. “Simultaneous Navigation of Multiple Pareto Surfaces, with an Application to Multicriteria IMRT Planning with Multiple Beam Angle Configurations”. *Medical Physics* 37 (2): 736–741.
- Deb, Kalyanmoy, and Abhishek Kumar. 2007. “Interactive Evolutionary Multi-Objective Optimization and Decision-Making Using Reference Direction Method”. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, 781–788. Association for Computing Machinery.
- Ehrgott, Matthias, and Ines Winz. 2008. “Interactive Decision Support in Radiation Therapy Treatment Planning”. *OR Spectrum* 30 (2): 311–329.

- Eskelinen, Petri, Kaisa Miettinen, Kathrin Klamroth, and Jussi Hakanen. 2010. “Pareto Navigator for Interactive Nonlinear Multiobjective Optimization”. *OR Spectrum* 32:211–227.
- Figueira, José Rui, Salvatore Greco, Vincent Mousseau, and Roman Słowiński. 2008. “Interactive Multiobjective Optimization Using a Set of Additive Value Functions”. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, edited by Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Słowiński, 97–119. Springer.
- Fredriksson, Albin, and Rasmus Bokrantz. 2013. “Deliverable Navigation for Multicriteria IMRT Treatment Planning by Combining Shared and Individual Apertures”. *Physics in Medicine and Biology* 58 (21): 7683–7697.
- Hakanen, Jussi, Kristian Sahlstedt, and Kaisa Miettinen. 2013. “Wastewater Treatment Plant Design and Operation under Multiple Conflicting Objective Functions”. *Environmental Modelling and Software* 46:240–249.
- Hartikainen, Markus, Kaisa Miettinen, and Kathrin Klamroth. 2019. “Interactive Nonconvex Pareto Navigator for Multiobjective Optimization”. *European Journal of Operational Research* 275 (1): 238–251.
- Hartikainen, Markus, Kaisa Miettinen, and Margaret M. Wiecek. 2012. “PAINT: Pareto Front Interpolation for Nonlinear Multiobjective Optimization”. *Computational Optimization and Applications* 52:845–867.
- Heikkinen, Risto, Juha Sipilä, Vesa Ojalehto, and Kaisa Miettinen. 2023. “Flexible Data Driven Inventory Management with Interactive Multi-Objective Lot Size Optimisation”. *International Journal of Logistics Systems and Management* 46 (2): 206–235.
- Kahneman, Daniel, and Amos Tversky. 1979. “Prospect Theory: an Analysis of Decision Under Risk”. *Econometrica* 47 (2): 263–292.
- Kania, Adhe, Juha Sipilä, Bekir Afsar, and Kaisa Miettinen. 2021. “Interactive Multiobjective Optimization in Lot Sizing with Safety Stock and Safety Lead Time”. In *International Conference on Computational Logistics*, 208–221. Springer.
- Korhonen, Pekka, and Jyrki Wallenius. 1988. “A Pareto Race”. *Naval Research Logistics* 35 (6): 615–623.

- Lin, Kuan-Min, and Matthias Ehrgott. 2018. “Multiobjective Navigation of External Radiotherapy Plans Based on Clinical Criteria”. *Journal of Multi-Criteria Decision Analysis* 25 (1–2): 31–41.
- Lin, Kuan-Min, Matthias Ehrgott, and Andrea Raith. 2017. “Integrating Column Generation in a Method to Compute a Discrete Representation of the Non-Dominated Set of Multi-Objective Linear Programmes”. *4OR* 15:331–357.
- Luque, Mariano, Francisco Ruiz, and Kaisa Miettinen. 2011. “Global Formulation for Interactive Multiobjective Optimization”. *OR Spectrum* 33 (1): 27–48.
- Miettinen, Kaisa. 1999. *Nonlinear Multiobjective Optimization*. Boston: 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 (2): 426–434.
- Miettinen, Kaisa, and Marko M. Mäkelä. 2006. “Synchronous Approach in Interactive Multiobjective Optimization”. *European Journal of Operational Research* 170 (3): 909–922.
- 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.
- Moazeni, Faegheh, and Javad Khazaei. 2021. “Interactive Nonlinear Multiobjective Optimal Design of Water Distribution Systems Using Pareto Navigator Technique”. *Sustainable Cities and Society* 73:103110.
- Monz, Michael. 2006. “Pareto Navigation – Interactive Multiobjective Optimisation and Its Application in Radiotherapy Planning”. PhD thesis, Technische Universität Kaiserslautern.

- Monz, Michael, Karl-Heinz Küfer, Thomas R Bortfeld, and Christian Thieke. 2008. “Pareto Navigation – Algorithmic Foundation of Interactive Multi-Criteria IMRT Planning”. *Physics in Medicine and Biology* 53 (4): 985–998.
- Pascoletti, Adriano, and Paolo Serafini. 1984. “Scalarizing Vector Optimization Problems”. *Journal of Optimization Theory and Applications* 42:499–524.
- Ruiz, Ana B., 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 (2): 213–231.
- Ruiz, Ana B., 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 (1): 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 (4): 865–889.
- Tarkkanen, Suvi, Kaisa Miettinen, Jussi Hakanen, and Hannakaisa Isomäki. 2013. “Incremental User-Interface Development for Interactive Multiobjective Optimization”. *Expert Systems with Applications* 40 (8): 3220–3232.
- Thieke, Christian, Karl-Heinz Küfer, Michael Monz, Alexander Scherrer, Fernando Alonso, Uwe Oelfke, Peter E. Huber, Jürgen Debus, and Thomas Bortfeld. 2007. “A New Concept for Interactive Radiotherapy Planning with Multicriteria Optimization: First Clinical Evaluation”. *Radiotherapy and Oncology* 85 (2): 292–298.
- Thole, Sidhant Pravinkumar, and Palaniappan Ramu. 2020. “Design Space Exploration and Optimization Using Self-Organizing Maps”. *Structural and Multidisciplinary Optimization* 62 (3): 1071–1088.

Yadav, Deepanshu, Palaniappan Ramu, and Kalyanmoy Deb. 2022. “Visualization-Aided Multi-Criterion Decision-Making Using Reference Direction Based Pareto Race”. In *Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, 125–132. IEEE.