Component-based thinking in designing interactive multiobjective evolutionary methods

Giomara Lárraga
giomara.g.larraga-maldonado@jyu.fi
University of Jyvaskyla
Faculty of Information Technology
Finland

Kaisa Miettinen
kaisa.miettinen@jyu.fi
University of Jyvaskyla
Faculty of Information Technology
Finland

ABSTRACT

Multiobjective optimization problems have multiple conflicting objective functions to be optimized simultaneously. They have many Pareto optimal solutions representing different trade-offs, and a decision-maker needs to find the most preferred one. Although most multiobjective evolutionary algorithms approximate the Pareto optimal set, their variants incorporate preference information to focus on a subset of solutions that interest the decision-maker. Interactive methods allow decision-makers to provide preference information iteratively during the solution process, enabling them to learn about available solutions and their preferences' feasibility. Nevertheless, most interactive evolutionary methods do not sufficiently support the decision-maker in finding the most preferred solution and may be cognitively too demanding.

We propose a framework for designing and implementing interactive evolutionary methods. It contains algorithmic components based on similarities in the structure of existing preference-based evolutionary algorithms and decision-makers' needs during interaction. The components can be combined in different ways to create new interactive methods or to instantiate the existing ones. We show an example of the implementation of the proposed framework composed of three elements: a graphical user interface, a database, and a set of algorithmic components. The resulting software can be utilized to develop new methods and increase their usability in real-world applications.

CCS CONCEPTS

• **Theory of computation** → **Continuous optimization**; **Evolutionary algorithms**.

KEYWORDS

multiobjective optimization, interactive methods, multiobjective evolutionary algorithms, decision-making.

ACM Reference Format:
Giomara Lárraga and Kaisa Miettinen. 2023. Component-based thinking in designing interactive multiobjective evolutionary methods. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583133.3596307

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3596307

1 INTRODUCTION

Multiobjective optimization problems (MOPs) involve multiple conflicting objective functions (or objectives, for short) that must be optimized simultaneously. Finding a single optimal solution that satisfies all objectives is not always possible for this type of problem. Instead, a set of solutions, forming a so-called Pareto optimal set, represents the best trade-offs between the different objectives. These solutions can be utilized for decision-making in various fields, such as engineering, economics, and finance. Various approaches have been devised for solving multiobjective optimization problems within the fields of multiple criteria decision-making (MCDM) and evolutionary multiobjective optimization (EMO). Examples of scalarization-based methods from MCDM can be found in [29] and in [10, 12] for EMO methods.

Multiobjective evolutionary algorithms (MOEAs) are optimization algorithms based on evolutionary computation, a subset of artificial intelligence inspired by the process of natural evolution. They can be classified into dominance-, indicator- and decomposition-based methods. They maintain a population of candidate solutions and iteratively apply genetic operations, such as crossover and mutation, to generate new solutions. The population is then evaluated according to the multiple objectives, and the solutions are selected based on their fitness. The process is repeated until an approximation of the Pareto optimal set is found (because of the heuristic nature, we cannot usually guarantee actual Pareto optimality).

One should note that only some Pareto optimal solutions are helpful for a decision-maker (DM), a domain expert interested in finding the most preferred solution according to their preferences. Usually, an analyst is also involved in the solution process, supporting the DM by performing analysis and providing recommendations.

According to the classification of multiobjective optimization methods given, e.g., in [21, 29], MOEAs can be regarded as *a posteriori* methods, as the preferences of the DM are considered after the optimization process (in selecting a final solution). However, a variety of MOEAs that incorporate the preferences of the DM have been proposed in the literature. These preference-based MOEAs focus on a region of the Pareto optimal set that is interesting for the DM called a region of interest.

A priori MOEAs ask for preference information at the beginning of the solution process. These methods are helpful when there is some prior knowledge about the problem and what kind of solutions are preferred, so the search is focused on promising regions of the Pareto optimal set. On the other hand, interactive MOEAs allow the DM to participate actively in the solution process by providing preferences iteration by iteration. In each iteration, the method shows a reduced set of solutions to the DM, who can then update

their preferences based on the obtained knowledge. The iterative solution process continues until the DM is satisfied with a solution. The benefits of these methods include increased computational efficiency when not all Pareto optimal solutions need to be represented and the ability of the DM to gain insight and learn, as well as revise preferences if needed.

A priori and interactive MOEAs are modifications of *a posteriori* MOEAs. Both types of methods share a similar structure, as they incorporate the preference information by modifying some of the evolutionary operators or adding new modules to the MOEA. However, most of the methods in the literature do not sufficiently consider the needs of real DMs. The main issues of the existing preference-based MOEAs regarding their usability by real DMs are the following:
High cognitive burden: Typically, preference-based MOEAs present many solutions to the DM to compare. The DM finds such a set difficult to digest and analyze, making it difficult to identify potentially good solutions [1].
Unrealistic assumptions on the knowledge of the DM: Some preference-based MOEAs ask the DM to provide technical information that is not related to their domain expertise but to the method's functionality, for example, the spread of the region of interest or parameters associated with the evolutionary operators. In addition, some methods ask for preference information in a form difficult to understand by the DM. Examples of these types of preference include 1) selecting reference vectors from a set [8, 9, 13, 42], 2) using a mathematical function to represent the preference of solutions [25, 31], 3) assigning values to each objective function to indicate their importance [3, 17, 28], and 4) establishing a fuzzy binary relation between objectives to reflect their preference [27, 44].
Difficult communication between the DM and the method: Most authors do not utilize a graphical user interface in their implementations to facilitate retrieving preference information and presenting the obtained solutions to DMs.

In this article, we identify multiple algorithmic components that can be utilized to construct new interactive MOEAs and to instantiate most existing ones. Then, we propose a framework for designing and implementing new interactive methods by combining the identified components in multiple ways. On the one hand, the proposed framework intends to increase awareness of the needs of real DMs when developing new interactive MOEAs. On the other hand, it enables the development of new methods conveniently. In addition, it facilitates the implementation of new methods and their further application. Our framework offers several key benefits:

- It enables the development of a new class of interactive MOEAs that not only focus on the technical aspects but also prioritize the user experience for real DMs.
- It takes advantage of the modularity of the methods, enabling the examination of novel mechanisms for incorporating preferences in MOEAs.
- It provides easy-to-use methods that assist DMs in finding a solution that aligns with their preferences.
- It allows for flexible implementation and is available as open-source software.

In summary, this article proposes a new way of designing interactive MOEAs that can better meet the needs of real DMs. The proposed framework builds on the structure of state-of-the-art *a priori* and interactive MOEAs and enhances it by adding new components that improve the communication between the DM and the methods. In addition, we provide a guideline for implementing a software that utilizes the proposed framework for facilitating the development and use of new interactive MOEAs. The software features a user-friendly interface and a database that can help to bridge the gap between theory and practice in the design and implementation of interactive MOEAs.

This article is structured as follows: Section 2 describes some algorithmic components for representing MOEAs. In the same section, we introduce frameworks for constructing interactive MOEAs proposed in the literature. We identify some algorithmic components for interactive MOEAs in Section 3 and utilize the identified components to instantiate some of the existing interactive MOEAs. We describe the structure of the proposed framework and exemplify its implementation in Section 4. Finally, we conclude the article and discuss the advantages and limitations of the proposed framework in Section 5.

2 RELATED WORK

Several preference-based MOEAs have been proposed in the literature. As a result, multiple authors have surveyed such methods and identified their advantages, drawbacks, and potential for improvement. A common finding from such survey articles is identifying algorithmic components common in most preference-based MOEAs. In addition, some attempts have been made to develop frameworks that facilitate the development of new interactive MOEAs. Although such frameworks also identify algorithmic components, a generalization is still missing in the literature. Such a generalization would be helpful to have a common guideline for allowing us to develop new methods that can be more usable for real DMs. This section analyzes the algorithmic components identified in both survey articles and frameworks for interactive MOEAs. Although the main focus of this research is to develop a framework for interactive MOEAs, analyzing the structure of *a priori* methods surveyed in the literature is highly beneficial for our purposes, as some of their components can also be useful for interactive methods.

2.1 Components of preference-based MOEAs identified in the literature

Coello [11] and Rachmawati et al. [32] classified preference-based MOEAs according to the type of preferences required by the methods, and did not identify structural similarities among the methods. Based on the authors' knowledge, Branke [6] presented the first survey of preference-based MOEAs, where algorithmic components were identified. He also noted that preference-based MOEAs are modified versions of general-purpose MOEAs. Some standard modules of these types of methods were identified: 1) information required from the DM, 2) modification of the MOEA, and 3) the type of output of the method (a delimited region or a biased distribution). Then, Bechikh et al. [36] and Wagner [39] utilized the same modular structure. However, the former extended the classification by considering various methods.

It is worth noting that although the components described in this section are useful to characterize *a priori* MOEAs, they are unsuitable for interactive ones. To adopt this structure for interactive

MOEAs, additional components should be considered to meet the actual needs of real DMs [1, 23, 37].

Xin et al. [40] proposed a taxonomy for interactive methods considering both scalarization-based methods and MOEAs, which include additional modules to the classification proposed in [6]. Such a taxonomy comprises the following components: 1) interacting pattern, 2) preference information, 3) preference model, and 4) search engine. The interacting pattern considers when to ask the DM to interact with the method, after the complete run of the search engine (when using an MOEA, after a certain number of generations), or during its run. The preference information is the way in which the DM expresses their preferences. Then, the preference model integrates the preferences of the DM into the method, guiding it to find interesting solutions for the DM. Finally, the search engine refers to the method that is utilized for optimization. The interaction pattern is the only component especially aimed at interactive methods. The rest of the components are similar to the classification proposed in [6]. They are intended to modify the method to direct the search toward a region of interest, so they can also be applied to generate *a priori* methods.

2.2 Frameworks for developing interactive MOEAs

A framework is a set of rules, guidelines, or principles that serve as a structure for achieving a particular goal or set of goals. In the case of designing interactive MOEAs, a framework refers to a set of modules that can be incorporated into any *a posteriori* MOEA to convert it into an interactive method. Analyzing some of these frameworks helps us better understand the existing algorithmic components identified and tested in the literature. Here we give an overview of some frameworks proposed in the literature for developing interactive MOEAs. Each of them works with a different type of MOEA.
Progressively Interactive EMO using Value Functions (PI-EMO-VF) [15]: The initial component of this framework utilizes an MOEA to approximate the Pareto optimal set. Then, a subset of solutions is shown to the DM. The DM is asked to provide a total or partial ranking according to their desirability. Such a ranking is then utilized to construct a value function, which is incorporated into the dominance principle of the MOEA. The procedure continues until the expected progress has been achieved according to the value function the method utilizes. This framework was tested using NSGA-II [14]. However, it can be utilized with any other dominance-based MOEA.
I-EMO-PLVF [24]: It is a framework for interactive decomposition-based MOEAs consisting of three modules: 1) consultation, 2) preference elicitation, and 3) optimization. During the consultation, the DM is asked to provide their preferences providing scalar scores for each solution generated by the method, representing their desirability. In the preference elicitation module, the scores are utilized to learn a value function progressively, which models the preferences of the DM. Then, the result of the consultation module is translated into the form that can be used in a decomposition-based MOEA, i.e., by redirecting the reference vectors toward a region of interest. The optimization module can be any decomposition-based MOEA. The authors utilized NSGA-III [13] and MOEA/D [43] during the experimentation.
Interactive Optimization using Preference Incorporated Space (IOPIS) [35]: It is a paradigm for interactive multiobjective optimization that reformulates the original problem utilizing multiple scalarization functions to map vectors in the objective space to a new space, called a preference incorporated space. It consists of the following steps: 1) preference elicitation, 2) problem creation, 3) problem solution, and 4) display solutions. First, the DM is asked to provide preference information as a reference point. Then, the problem is reformulated as a new multiobjective optimization problem. As objective functions, it has scalarization functions that incorporate the reference point. Any MOEA is then applied to solve the new problem, and the obtained solutions are filtered to show the DM only a subset of representative solutions. The procedure continues until the DM finds a satisfactory solution.
A general architecture for interactive decomposition-based MOEAs [23] consists of multiple modules for generating interactive MOEAs that accomplish the needs of real DMs (e.g., low cognitive burden and feeling of control over the interactive solution process). The main modules of this architecture are the following: 1) initialization, 2) preference elicitation, 3) component adaptation, 4) optimization, 5) spread adjustment, 6) selection of solutions, and 7) iteration. During the initialization, which information to show to the DM at the beginning of the solution process is decided. Then, the method asks for preference information from the DM in the preference elicitation module. The preference information is utilized to modify one of the components of a decomposition-based MOEA to focus on a region of interest. Such a modified component is utilized in a decomposition-based MOEA, which produces solutions in the region of interest. The solutions are filtered to show only a subset of representative solutions to the DM, who can decide to continue with another iteration or finish the solution process. The authors identified the potential of extending this architecture to other types of MOEAs, as some of the components can be generalized.

3 ALGORITHMIC COMPONENTS OF INTERACTIVE EVOLUTIONARY METHODS

As noted earlier, some work has been done to identify algorithmic components for preference-based MOEAs. In this article, we identify standard algorithmic components to build interactive MOEAs taking into account the ones available in the literature (see Section 2). The algorithmic components considered aim to generalize the existing ones and to include some additional functionalities related to the actual needs of a DM during an interactive solution process [1, 23, 37]. We aim to provide a guideline for designing new interactive MOEAs that can be applied to solve real-world problems and are easy to use by DMs. To identify the common algorithmic components of interactive MOEAs, we need first to describe what an interactive solution process should be like and the impact of each stage on the DM. Then, we relate the components that have been identified for each of the stages in the approaches described in Section 2. Finally, we generalize such components for further use in the framework proposed in this article.

3.1 Interactive solution process and related algorithmic components

In this section, we describe the general steps of an interactive solution process and the needs of a DM in each of them. We derive an algorithmic component from each of the steps of the process. It is worth noting that we are renaming the components in most of the cases because of the need for unification in the terminology. For simplicity, we assume minimization problems in what follows.

STEP 1: At the beginning of an interactive solution process, it is crucial to give the DM information to help them define their initial preferences and identify potential solutions. To do this, an *a posteriori* MOEA is usually run for several generations. After that, three outputs can be obtained:

(1) An approximation of an ideal point, which is found by taking the minimum values of each objective function among the solutions in the approximated Pareto optimal set.
(2) An approximation of a nadir point, which is found by taking the maximum values of each objective function in the same set.
(3) A subset of solutions, which requires a parameter specifying the number of solutions to be shown to the DM. This selection can be made, e.g., using a clustering mechanism to find a desired number of the most representative solutions.

This step is considered in the architecture proposed in [23]. In the rest of this article, we refer to this component as an *initializer*.

STEP 2: The DM is asked to provide preference information. This information can be expressed in various forms. Although it is desirable for the DM to choose the type of preference, MOEAs are often not flexible in this regard. Commonly used types of preference information in the EMO literature include:

- Reference point: A vector of desired values, also known as aspiration levels for each objective function.
- Preferred and non-preferred solutions: One or more preferred or non-preferred solutions selected from a set of solutions generated by the method.
- Ranking solutions: A partial or complete ordering of a subset of solutions generated by the method.
- Preferred ranges: Desired upper and lower bounds for each objective function.
- Trade-offs between objectives: When a unit is gained in one objective, how many units is it worth sacrificing in others?
- Ranking objectives: Objectives are compared and ranked using pairwise comparisons.

This step is a critical component of the solution process and is considered in all taxonomies and frameworks described in Section 2. In the rest of the article, we refer to this as *preference elicitation*.

STEP 3: The preference information is modeled to be incorporated in the optimization method. Usually, this modifies one of the evolutionary operators of the MOEA. Depending on the type of MOEA, mainly the following operators can be modified:

- Selection: the solutions that remain in the next generation are selected based on the preference information. To this aim, multiple variations of the dominance relation have been proposed [6, 36]. In decomposition-based MOEAs, it is possible to have a mechanism in the selection operator to update the reference vectors for directing the search toward the region of interest (e.g., [18, 20]).
- Quality indicator: For indicator-based MOEAs, it is possible to develop quality indicators that consider the preferences of the DM to guide the algorithm, as in [37].
- Additional operator: Some approaches reformulate the problem considering the preference information before starting the solution process [35]. In addition, multiple decomposition-based MOEAs rearrange the reference vectors at the beginning of the solution process (e.g., [18, 20]).

This component is also known as a modification [6], a preference model [40], preference elicitation [24], and component adaptation [23]. This article refers to it as a *preference handler*.

STEP 4: The preference handler is incorporated in the MOEA to guide the search toward a region of interest. In the literature, this component has been referred to as simply an MOEA [36], optimization [23, 24], and problem solution [35]. In the rest of this article, we call it an *optimizer*.

STEP 5: A subset of solutions from the region of interest generated by the MOEA is shown to the DM. In this step, it is important to let the DM decide the maximum number of solutions they can handle in each iteration. Different methods can be used to choose the most representative solutions from the region of interest depending on the type of preference. E.g., if the DM's preference information is a reference point, a scalarization function can be used to pick the solutions closest to it (e.g., [37]). A common approach is to use a clustering method and pick the closest solution to each cluster center. This component can also consider reducing the spread of the region of interest in the latter stages of the solution process. This is to help the DM to find the most preferred solution. This component has been referred to in the literature as display solutions [35] and selection of solutions [23]. We name this component as *filtering of solutions*.

STEP 6: The DM examines the solutions presented by the method. This could result in one of three cases: 1) the DM is satisfied with a solution and chooses it as the final one, ending the solution process; 2) the DM wants to see more solutions in the same area of interest and continues the solution process; or 3) the DM wants to explore a different region of interest and updates their preferences. It is worth noting that the stopping criterion must consider the satisfaction of the DM with one or multiple solutions. This component has been referred to as iteration [23]. However, most approaches do not consider this step despite its importance. We refer to this component as a *continuation procedure*.

3.2 General framework for interactive MOEAs

In this section, we describe the connections of the identified components to establish a framework for designing interactive evolutionary methods. The framework's structure is illustrated in Figure 1. According to such a structure, designing an interactive method involves the following steps:

(1) Determine the information shown to the DM at the beginning of the solution process. This step is optional, but it is highly desirable to show at least one of the options to the DM (ideal point, nadir point, or a subset of solutions).

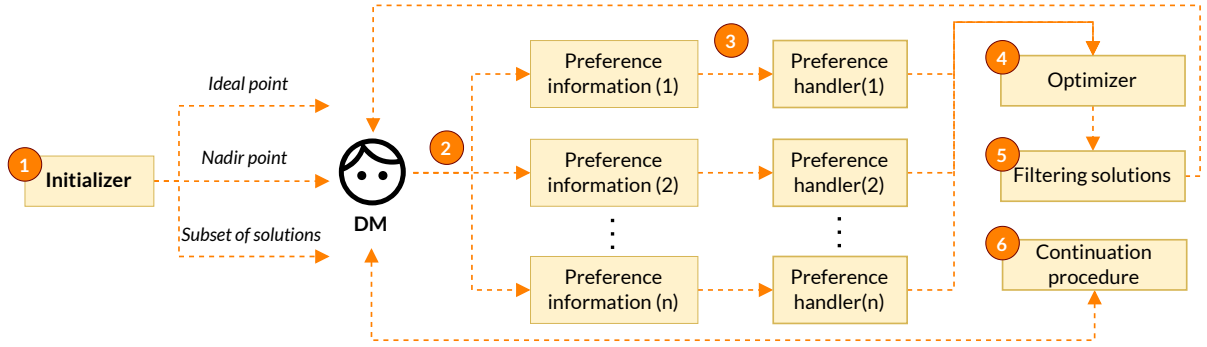

Figure 1: Framework for designing interactive MOEAs

(2) Select one or multiple types of preference. It is mandatory to select at least one type of preference and is desirable to consider more than one to give more flexibility to the DM.
(3) Select a preference handler for each type of preference selected in the previous step. Not all the handlers will be utilized simultaneously, but the one related to the utilized preference information in each iteration would be selected.
(4) Incorporate the preference handler into the optimizer to direct the search toward a region of interest.
(5) Utilize a filtering procedure to select a small set of representative solutions from the region of interest obtained from the previous iteration. These solutions are shown to the DM.
(6) Establish a continuation procedure. This involves requesting the DM to review the solutions obtained in the preceding step and determine whether to proceed or end the process.

The framework proposed in this article shares similarities with the architecture presented in [23], as both incorporate modules for designing interactive MOEAs that cater to the DM's needs. However, their main distinction lies in the scope of their structure. While [23] focuses on decomposition-based MOEAs, our framework is applicable to interactive dominance-based, decomposition-based, and indicator-based MOEAs. Additionally, the functionality of these approaches also differs in terms of how MOEAs are employed. The architecture of [23], proposes to transform existing *a priori* and *a posteriori* methods into interactive versions using external modules to handle preference information. In contrast, our proposal extracts preference-handling mechanisms from existing preference-based MOEAs and reutilizes them to design new interactive MOEAs.

3.3 Interactive MOEAs as instances of the proposed framework

In this section, we show the generality of the proposed framework by instantiating some existing interactive MOEAs from the literature. It is worth noting that not all interactive methods consider all the components identified in this article. As shown in [1], it is common that some interactive MOEAs do not consider the actual needs of real DMs. Because of this, components such as the initializer and filtering solutions are not present in the structure of some of the methods. Table 1 shows some relevant interactive MOEAs as instances of the proposed framework.

4 A SOFTWARE FOR DESIGNING INTERACTIVE METHODS

In this section, we present a design proposal for a software intended to facilitate the implementation of interactive MOEAs. The software aims to simplify the development and use of new interactive MOEAs by promoting good design practices learned from, e.g., the MCDM field. With an increasing interest in interactive MOEAs, it is crucial to consider the needs of real DMs, not just technical details. Otherwise, the methods will not be applicable to real problems by real DMs. The envisioned software is expected to drive a new era of interactive MOEAs that cater to the needs of DMs. The proposed software is composed of three components:

(1) A framework for designing interactive methods.
(2) A database for storing the metadata of the methods and the solutions generated by the solution process.
(3) A graphical user interface for easy communication between the DM and the methods.

The rest of this section will illustrate the software's implementation, highlighting each component.

4.1 Framework for the modular design of interactive MOEAs

Several open-source software frameworks are available for solving multiobjective optimization problems using evolutionary algorithms. The Platypus software, written in Python, provides several MOEAs and a performance evaluation tool. MOEA [19] is a Java-based framework that offers automatic process distribution across multiple cores and state-of-the-art MOEAs. The C++-based PaGMO [4] software provides both single and multiobjective optimization methods with parallelization capabilities, and there is a Python version of the software named PyGMO [22]. jMetal [16] is a Java-based, object-oriented framework with metaheuristics available in multiple languages, including C++ [26] and Python [2]. Pymoo [5] is a Python-based framework that includes visualization techniques. PlatEMO [38] is a MATLAB-based open-source software that provides multiple MOEAs, performance indicators, and a graphical user interface but requires a MATLAB license.

The Python-based open-source framework DESDEO [30] is the only framework that includes interactive methods, both scalarization-based methods and MOEAs. Because of our interest in interactive methods, we use DESDEO to demonstrate the implementation of the proposed framework, but it can also be implemented in

Table 1: Instantiation of state-of-the-art interactive MOEAs using the proposed framework.

Method	Initializer	Preference incorporation	Preference handler	Optimizer	Filtering solutions	Continuation procedure
Interactive PBEA [37]	Show a subset of solutions	Reference point	Quality indicator	PBEA [37]	Achievement function	DM's satisfaction
NEMO [7]	Show a pair of solutions	Pairwise comparisons	Modified selection (ranking and crowding distance)	NSGA-II [14]	Random	Not specified
iMOEA/D [18]	Show a subset of solutions	Preferred solution	Rearrangement of reference vectors	MOEA/D [43]	Random	Fixed number of iterations
Interactive WASF-GA [33]	Show ideal and nadir points	Reference point	Rearrangement of reference vectors	WASF-GA [34]	Clustering	DM's satisfaction
Interactive RVEA [20]	Show a subset of solutions	1) Reference points 2) (Non)preferred solutions 3) Preferred ranges	Different types of adaptation of reference vectors	RVEA [8]	Clustering	DM's satisfaction
ICB-MOEA/D [41]	Show a subset of solutions	Preferred solution	Problem reformulation	MOEA/D [43]	Show entire population	DM's satisfaction
1) I-MOEA/D-PLVF and 2) I-NSGA-III-PLVF [24]	Show a subset of solutions	Scores	Learn value function to adapt reference vectors	1)MOEA/D [43] 2)NSGA-III [13]	Solutions with best value function	Fixed number of iterations
1) IOPIS-RVEA 2) IOPIS-NSGA-III [35]	Show ideal and nadir points	Reference point	Problem reformulation	1) RVEA [8] 2) NSGA-III [13]	Clustering	DM's satisfaction

any other framework with a modular structure for MOEAs. For the implementation of the framework, we are interested in the DESDEO package named desdeo-emo. This package allows a modular implementation of MOEAs by considering the following modules: *population*, *recombination*, *selection*, *EAs*, *surrogatemodelling*, and *utilities*. More details on each module can be found in [30]. Taking advantage of the modularity of desdeo-emo, we can incorporate the algorithmic components described in Section 3. Figure 2 illustrates how to combine desdeo-emo with the algorithmic components identified in this research. The yellow boxes represent the main modules of desdeo-emo, and the blue boxes are the algorithmic components identified in this article that need to be incorporated in desdeo-emo. The implementation decisions are justified as follows:

The EA module in desdeo-emo integrates the interaction between the MOEA and the DM. This module can connect components such as the initializer, preference information types, solution filtering, and continuation procedure.

The initializer requires parameters that determine the type of information to present to the DM at the beginning of the solution process. To implement this, the EA module already includes a *pre-iteration* function that can be utilized to perform certain functions before each iteration. In this case, the first iteration needs to be identified within the *pre-iteration* function to ensure that the desired functionality only occurs at the beginning of the solution process.

The EA module contains the function *allowable_interaction_types* to specify the preference types considered by each MOEA. However, the validation and request of preferences take place in the desdeo-tools package. Therefore, incorporating new types of preference would require modifying the *Interaction* module on desdeo-tools.

The preference handlers can be managed in the *manage_preferences* function of the EA module. However, it requires coding multiple preference handlers (selection operators, recombination procedures, or other utilities as reference vector adaptation) and implementing a mechanism to select which one(s) to utilize with the optimizer. This is the biggest challenge in the implementation, as it involves restructuring some parts of the *EA* module. In addition, a decision tree can be helpful for the *manage_preferences* function to only allow preference handlers that can be utilized with the type of preferences and the optimizer selected.

The modular structure of desdeo-emo addresses the functionality of the optimizer component. However, the module is currently mainly oriented to decomposition-based and indicator-based approaches. To make the framework more general, it is important to implement more types of MOEAs, e.g., MOEAs based on dominance.

The *post-iteration* function of the EA module can be useful to incorporate the functionality of the filtering solutions component. To this aim, it is important to implement multiple filtering procedures and make connections with the type of preferences with which they can be utilized.

Finally, the continuation procedure is also considered in the EA module. However, such functionality must be improved to consider the three possible cases for this component: 1) end the solution process, 2) continue using the same preference information, or 3) explore a new region of interest. When the DM is interested in continuing the solution process using the same preferences, it is possible to reduce the spread of the region of interest to ease the selection of the most preferred solution of the DM.

It is worth noting that this is just an example of implementing the proposed framework to create a software for designing and utilizing interactive MOEAs. The specifics of the implementation, such as modifications to the desdeo-emo modules to integrate the new functionalities, may vary based on any issues encountered during implementation or to improve its performance. Furthermore, other frameworks for MOEAs (e.g., pymoo or jMetal) could be utilized to achieve the same implementation.

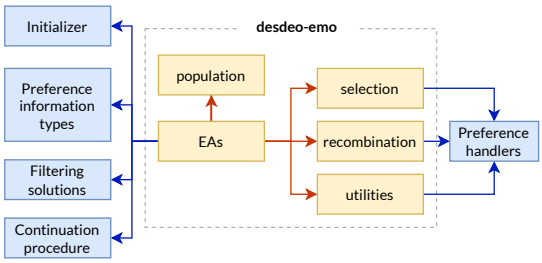

Figure 2: Integration of the proposed framework with desdeo-emo

4.2 Structure of the database

A database is needed in the proposed software for multiple purposes. Its main aim is storing the parameters needed to construct a method in the framework. In addition, a database can benefit DMs, as they can store potentially good solutions for each solution process. In the following, we briefly discuss the components of the suggested database. It is composed of reference tables and base tables.

Reference tables are used in database design to maintain consistency in the data and to reduce data redundancy. They typically contain a list of items along with their attributes or features. In the database model utilized for our software, we consider the following reference tables:

- PreferenceTypes: This table lists the possible types of preferences considered in the software. The attributes of this table include a name and a description.
- PreferenceHandlers: A list of preference handlers is included in this table. Their attributes include a name, the type of evolutionary operator, the type of MOEA in which it can be used, and the type of preference information accepted as input.
- Optimizer: It includes a list of MOEAs available in the software. As attributes, this table stores the name of the MOEA and its type (decomposition-based, indicator-based, or dominance-based).
- Problem: A list of MOPs is described in this table, indicating the number of variables, objectives, and the presence of constraints, among other attributes.

Base tables are the primary tables in a relational database that store data. They contain columns and rows and are used to store data. These tables are a key component of a relational database and play a critical role in supporting data storage, retrieval, and management. Such tables can contain primary keys and foreign keys. A primary key is a unique identifier for each record in a table. It is used to identify a specific record in the table and to enforce data integrity. Each table in a relational database should have a primary key, and the values in the primary key column should be unique for each record. On the other hand, a foreign key is a column or set of columns in one table that refers to the primary key of another table. A foreign key is used to establish a relationship between two tables and to enforce referential integrity, which ensures that the data in one table is consistent with the data in another table. The tables and attributes considered in the proposed database are the following:

- User: it includes a list of usernames and the type of user (analyst or DM). More attributes can be added to this table if additional user information is needed (email, password, etc.).
- Method: this table includes the parameters to construct an interactive method using the proposed framework. It contains an attribute for specifying the method's name and two foreign keys: *user_id* (the id of the user that created the method) and *optimizer_id* (the MOEA utilized in the interactive method). Additional parameters can also be included in this table, such as the parameters of the evolutionary operators.
- Initializer: the configuration of the initializer for each method is included in a record on this table. It has a foreign key (*method_id*) to be connected with the Method table.
- AllowedPreferences: this table records the types of preferences applicable for each method. It is connected to the Method, PreferenceTypes, and PreferenceHandler by foreign keys. Each method can be assigned to multiple types of preferences. And for each one, a preference handler is needed.
- SolutionProcess: this table represents an interactive solution process involving attributes such as *user_id*, *method_id*, and *problem_id*, which are also foreign keys.
- ArchivedSolutions: the solutions stored by the DM during a solution process are stored in this table. It is connected to the SolutionProcess table with a foreign key.
- Filtering: this table specifies the parameters for the filtering procedure utilized by each method. It is connected to the Method table with a foreign key.
- Continuation: it records the parameters of the continuation procedure of each method. It is connected to the Method table with a foreign key.

4.3 Graphical user interface

A graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices, such as computers, using visual elements such as windows, icons, and buttons. The goal of a GUI is to make the user experience more intuitive and user-friendly, allowing users to interact with a device using a familiar graphical interface rather than typing commands into a command line interface.

In a decision-making context, a GUI can provide a visual representation of information and allows DMs and analysts to interact with the data more intuitively and in a user-friendly manner. A GUI can help users make informed decisions by:

- Presenting information in a clear and organized manner: A GUI can present information in a way that is easy to understand and visually appealing. It can display data using graphs, charts, tables, and other visual elements that make it easier to analyze and compare the information.
- Enhancing collaboration: A GUI can provide a platform for collaboration between multiple users. For example, an analyst can design an interactive method with the proposed software, implement a problem, and involve a DM to solve it. Then, the analyst can access the solutions found by the DM or get feedback about the method. Also, the DM can explore the solutions stored during the solution process.
- Improving accessibility: A GUI can make it easier for users to access information and make decisions, even if they are not technically savvy. With its visual elements and intuitive

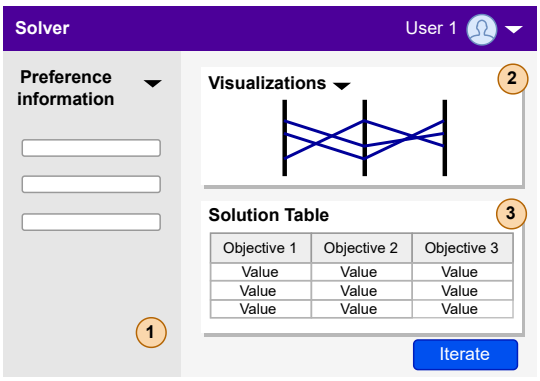

Figure 3: Sketch of the interface for solving a MOP

interface, a GUI can reduce the need for training and support, making it accessible to a wider range of users.

The proposed software is designed with multiple windows for each of its functionalities, catering to two types of users: analysts and DMs. Analysts can create new methods using an interactive GUI, with all parameters saved in the database. They can also view and use all previously created methods. In addition, they can assist DMs in choosing an appropriate method to solve optimization problems. DMs can then use the recommended method, store their preferred solutions, and choose their preferred visualization options.

The GUI for generating new methods involves multiple steps related to the algorithmic components of the framework. For this reason, it is difficult to illustrate this part of the GUI in the present article. However, we illustrate the interface to help the DM solve a problem. A sketch of the interface's structure is illustrated in Figure 3.

This interface is divided into three main sections:

(1) Preference information bar: the DM can provide their preferences in the vertical bar on the left of the window. If the method can handle multiple types of preferences, they can be selected by clicking the arrow at the top of the bar. It is worth noting that the form for retrieving the preferences differs depending on the utilized type. The text boxes in the preference information bar of Figure 3 are examples of the type of controls that can be utilized to retrieve information.
(2) Visualizations section: there is a box at the top right of the window that aims to display the obtained solutions utilizing a visualization. By clicking the arrow on the box's title, the DM can select the type of visualization to utilize (e.g., parallel coordinate plot, petal diagram, etc.).
(3) Solution table: there is a table on the bottom right of the window to show the solutions generated by the method at each iteration. In addition, this table can also be utilized to display the solutions stored during the solution process. The DM can decide whether one wants to view objective or decision variable values.

5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this article, we identified algorithmic components to design new interactive MOEAs. Such components were identified from the existing surveys and frameworks from the literature, but also considering the needs of DMs during an interactive solution process. The methods designed with the proposed framework aim to be user-friendly and reduce DMs' cognitive burden. This is accomplished by providing informative guidance at the start of the solution process and granting DMs the flexibility to select the type of preference information and the number of solutions to be shown in each iteration. With this framework, we want to increase awareness of the needs of real DMs during an interactive solution process. It is the first step in developing a new generation of interactive MOEAs that are not only good in terms of performance but also improve their usability in real-world applications.

We also proposed a way to implement the proposed framework to create a software that analysts and DMs can use. The main functionalities of this software are to allow the analysts to create new interactive MOEAs intuitively and to have the possibility of utilizing them with a GUI. For DMs, the proposed software can enhance the communication between them and the methods, providing an easy way of expressing their preferences and visualizing the obtained solutions. The main research directions derived from this research are the following:

Generalization for multiple types of interaction: Not all the existing interactive MOEAs can be instantiated utilizing the identified components. As it has been identified in the literature [40], there are multiple types of interaction: during the run of the MOEA and after running the MOEA for a certain number of generations. The current stage of the proposed framework only considers the interaction after running an MOEA for a given number of generations. Such generations are usually provided by the analyst together with the technical parameters of the method. However, extending our framework to generalize more interactive MOEAs is currently a research subject.

Identifying preference handlers: There exist multiple ways of handling different types of preference information. Identifying preference handlers requires an extensive literature review regarding *a priori* and interactive MOEAs.

Usability tests: Designing a GUI that is easy to use for a wide variety of users is not an easy task. The implementation of the GUI is in an early stage, where the involvement of experts on multiobjective optimization and decision-making is crucial. After prototyping a functional GUI that meets the requirements of a decision-making system, it is important to perform usability tests to prove that the GUI is accessible and user-friendly and helps guide the DM through the solution process.

Combination of multiple components: The framework's modularity allows us to generate new methods by combining the components in different ways. An interesting research direction is the creation and comparison of multiple interactive MOEAs utilizing the proposed framework to identify the advantages and drawbacks of each approach.

6 ACKNOWLEDGMENTS

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

REFERENCES

[1] B. Afsar, K. Miettinen, and F. Ruiz. Assessing the performance of interactive multiobjective optimization methods: A survey. *ACM Computing Surveys*, 54(4): article 85, 2021. doi: 10.1145/3448301.

[2] A. Benítez-Hidalgo, A.J. Nebro, J. García-Nieto, I Oregi, and J. Del Ser. jmetalpy: A python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, 51:article 100598, 2019. doi: 10.1016/j.swevo.2019.100598.

[3] X. Bi, . Yu, J. Liu, and Y. Hu. A preference-based multi-objective algorithm for optimal service composition selection in cloud manufacturing. *International Journal of Computer Integrated Manufacturing*, 33(8):751–768, 2020. doi: 10.1080/0951192X.2020.1775298.

[4] F. Biscani, D. Izzo, and C. H. Yam. A global optimisation toolbox for massively parallel engineering optimisation. *arXiv:1004.3824*, 2010. doi: 10.48550/arXiv.1004.3824.

[5] J. Blank and K. Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020. doi: 10.1109/ACCESS.2020.2990567.

[6] J. Branke. Consideration of partial user preferences in evolutionary multiobjective optimization. In J. Branke, K. Deb, K. Miettinen, and R. Słowiński, editors, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 157–178, Berlin, Heidelberg, 2008. Springer. doi: 10.1007/978-3-540-88908-3_6.

[7] J. Branke, S. Greco, R. Słowiński, and P. Zielniewicz. Interactive evolutionary multiobjective optimization using robust ordinal regression. In M. Ehrgott, C. M. Fonseca, X. Gandibleux, J. Hao, and M. Sevaux, editors, *Evolutionary Multi-Criterion Optimization*, pages 554–568, Berlin, Heidelberg, 2009. Springer. doi: 10.1007/978-3-642-01020-0_43.

[8] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791, 2016. doi: 10.1109/TEVC.2016.2519378.

[9] R. Cheng, T. Rodemann, M. Fischer, M. Olhofer, and Y. Jin. Evolutionary Many-Objective Optimization of Hybrid Electric Vehicle Control: From General Optimization to Preference Articulation. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1(2):97–111, 2017. doi: 10.1109/TETCI.2017.2669104.

[10] C. A. Coello Coeallo, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, 2nd edition, 2007. doi: 10.1007/978-0-387-36797-2.

[11] C.A. Coello Coello. Handling preferences in evolutionary multiobjective optimization: a survey. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 30–37, 2000. doi: 10.1109/CEC.2000.870272.

[12] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, 2001.

[13] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014. doi: 10.1109/TEVC.2013.2281535.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6 (2):182–197, 2002. doi: 10.1109/4235.996017.

[15] K. Deb, A. Sinha, P. J. Korhonen, and J. Wallenius. An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE Transactions on Evolutionary Computation*, 14(5):723–739, 2010. doi: 10.1109/TEVC.2010.2064323.

[16] J.J. Durillo, A.J. Nebro, and E. Alba. The jmetal framework for multi-objective optimization: Design and architecture. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010. doi: 10.1109/CEC.2010.5586354.

[17] W.M Ferreira, I.R Meneghini, D.I Brandao, and Guimarães.F.G. Preference cone based multi-objective evolutionary algorithm to optimal management of distribuited energy resources in microgrids. *Applied Energy*, 274:115326, 2020. doi: 10.1016/j.apenergy.2020.115326.

[18] M. Gong, F. Liu, W. Zhang, L. Jiao, and Q. Zhang. Interactive MOEA/D for Multi-objective Decision Making. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*, New York, New York, USA, 2011. ACM Press. doi: 10.1145/2001576.2001675.

[19] D. Hadka. MOEA framework: A free and open source java framework for multiobjective optimization., 2020. URL http://moeaframework.org/. accessed December 01, 2020.

[20] J. Hakanen, T. Chugh, K. Sindhya, Y. Jin, and K. Miettinen. Connections of reference vectors and different types of preference information in interactive multiobjective evolutionary algorithms. In *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*. IEEE, 2017. doi: 10.1109/SSCI.2016.7850220.

[21] C.-L Hwang and A.S.M. Masud. *Multiple Objective Decision Making – Methods and Applications: A State-of-the-Art Survey*. Springer, Berlin, Heidelberg, 1979. doi: 10.1007/978-3-642-45511-7.

[22] D. Izzo and F. Biscani. PyGMO: Python parallel global multiobjective optimizer., 2020. URL https://esa.github.io/pygmo. accessed November 19, 2020.

[23] G. Lárraga and K. Miettinen. A general architecture for generating interactive decomposition-based MOEAs. In G. Rudolph, A. V. Kononova, H. Aguirre, P. Kerschke, G. Ochoa, and T. Tušar, editors, *Parallel Problem Solving from Nature – PPSN XVII*, pages 81–95, Cham, 2022. Springer.

[24] K. Li, R. Chen, D. Savić, and X. Yao. Interactive decomposition multiobjective optimization via progressively learned value functions. *IEEE Transactions on Fuzzy Systems*, 27(5):849–860, 2019. doi: 10.1109/TFUZZ.2018.2880700.

[25] Z. Li and H.L. Liu. Integrating preferred weights with decomposition based multi-objective evolutionary algorithm. In *Proceedings of the 10th International Conference on Computational Intelligence and Security, CIS 2014*, pages 58–63. IEEE, 2015. doi: 10.1109/CIS.2014.117.

[26] E. Lopez-Camacho, M.J. Garcia Godoy, A.J. Nebro, and J.F. Aldana-Montes. jmetalcpp: optimizing molecular docking problems with a c++ metaheuristic framework. *Bioinformatics*, 30(3):437–438, 2014. doi: 10.1093/bioinformatics/btt679.

[27] B. Luo, L. Lin, and S. Zhong. PGA/MOEAD: a preference-guided evolutionary algorithm for multi-objective decision-making problems with interval-valued fuzzy preferences. *International Journal of Systems Science*, 49(3):595–616, 2018. doi: 10.1080/00207721.2017.1412537.

[28] I.R. Meneghini, F. Gadelha Guimarães, A. Gaspar-Cunha, and M. Weiss Cohen. Incorporation of region of interest in a decomposition-based multi-objective evolutionary algorithm. In A. Gaspar-Cunha, J. Periaux, K.C. Giannakoglou, N.R. Gauger, D. Quagliarella, and D. Greiner, editors, *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, pages 35–50. Springer, 2021. doi: 10.1007/978-3-030-57422-2_3.

[29] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1999.

[30] G. Misitano, B. S. Saini, B. Afsar, B. Shavazipour, and K. Miettinen. DESDEO: The modular and open source framework for interactive multiobjective optimization. *IEEE Access*, 9:148277–148295, 2021. doi: 10.1109/ACCESS.2021.3123825.

[31] M. Pilát and R. Neruda. Incorporating user preferences in MOEA/D through the coevolution of weights. In *Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, pages 727–734, New York, 2015. ACM. doi: 10.1145/2739480.2754801.

[32] L. Rachmawati and D. Srinivasan. Preference incorporation in multi-objective evolutionary algorithms: A survey. In *2006 IEEE International Conference on Evolutionary Computation*, pages 962–968, 2006. doi: 10.1109/CEC.2006.1688414.

[33] A. B. Ruiz, M. Luque, K. Miettinen, and R. Saborido. An interactive evolutionary multiobjective optimization method: Interactive WASF-GA. In A Gaspar-Cunha, C. Antunes, and C. Coello, editors, *Evolutionary Multi-Criterion Optimization: 8th International Conference, Proceedings, Part II*, pages 249–263, Berlin, Heidelberg, 2015. Springer. doi: 10.1007/978-3-319-15892-1_17.

[34] A.B. Ruiz, R. Saborido, and M. Luque. A preference-based evolutionary algorithm for multiobjective optimization: the weighting achievement scalarizing function genetic algorithm. *Journal of Global Optimization*, 62(1):101–129, 2015.

[35] B. S. Saini, J. Hakanen, and K. Miettinen. A new paradigm in interactive evolutionary multiobjective optimization. In T. Back, M. Preuss, A. Deutz, H. Wang, C. Doerr, H. Emmerich, and M. Trautmann, editors, *Parallel Problem Solving from Nature - PPSN XVI, 16th International Conference, Proceedings, Part II*, pages 243–256. Springer, 2020. doi: 10.1007/978-3-030-58115-2_17.

[36] B. Slim, K. Marouane, B.S. Lamjed, and G. Khaled. Chapter four - preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. *Advances in Computers*, 98:141–207, 2015. doi: 10.1016/bs.adcom.2015.03.001.

[37] L. Thiele, K. Miettinen, P. J. Korhonen, and J. Molina. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary Computation*, 17(3):411–436, 2009. doi: 10.1162/evco.2009.17.3.411.

[38] Y. Tian, R. Cheng, X. Zhang, and Y. Jin. Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]. *IEEE Computational Intelligence Magazine*, 12(4):73–87, 2017. doi: 10.1109/MCI.2017.2742868.

[39] T. Wagner and H. Trautmann. Integration of preferences in hypervolume-based multiobjective evolutionary algorithms by means of desirability functions. *IEEE Transactions on Evolutionary Computation*, 14(5):688–701, 2010. doi: 10.1109/TEVC.2010.2058119.

[40] B. Xin, L. Chen, J. Chen, H. Ishibuchi, K. Hirota, and B. Liu. Interactive multiobjective optimization: A review of the state-of-the-art. *IEEE Access*, 6:41256–41279, 2018. doi: 10.1109/ACCESS.2018.2856832.

[41] B. Xin, H. Li, and L. Wang. ICB-MOEA/D: An interactive classification-based multi-objective optimization algorithm. In *Chinese Control Conference, CCC*, pages 2500–2505. IEEE, 2018. doi: 10.23919/ChiCC.2018.8482688.

[42] J. Yan and H. Deng. Generation of large-bandwidth x-ray free electron laser with evolutionary many-objective optimization algorithm. *Physical Review Accelerators and Beams*, 22(2), 2019. doi: 10.1103/PhysRevAccelBeams.22.020703.

[43] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. doi: 10.1109/TEVC.2007.892759.

[44] X. Zhu, Z. Gao, and Z. Chen. Studies on the application of MOEA/D to aerodynamic optimization design. In *31st Congress of the International Council of the Aeronautical Sciences, ICAS 2018*. International Council of the Aeronautical

Sciences, 2018.