A general architecture for generating interactive decomposition-based MOEAs

Giomara Lárraga ✉[1 [0000−0001−8280−7040]] and Kaisa Miettinen[1 [0000−0003−1013−4689]]

University of Jyvaskyla, Faculty of Information Technology, FI-40014 University of Jyvaskyla, Finland
{giomara.g.larraga-maldonado,kaisa.miettinen}@jyu.fi

Abstract. Evolutionary algorithms have been widely applied for solving multiobjective optimization problems. Such methods can approximate many Pareto optimal solutions in a population. However, when solving real-world problems, a decision maker is usually involved, who may only be interested in a subset of solutions that meet their preferences. Several methods have been proposed to consider preference information during the solution process. Among them, interactive methods support the decision maker in learning about the trade-offs among objectives and the feasibility of solutions. Also, such methods allow the decision maker to provide preference information iteratively. Typically, interactive multiobjective evolutionary algorithms are modifications of existing *a priori* or *a posteriori* algorithms. However, they mainly focus on finding a region of interest and do not support the decision maker finding the most preferred solution. In addition, the cognitive load imposed on the decision maker is usually not considered. This article proposes an architecture for developing interactive decomposition-based evolutionary algorithms that can support the decision maker during the solution process. The proposed architecture aims to improve the applicability of interactive methods in solving real-world problems by considering the needs of a decision maker. We apply our proposal to generate an interactive decomposition-based algorithm utilizing a reference vector re-arrangement procedure and MOEA/D. We demonstrate the performance of our proposal with a real-world problem and multiple benchmark problems.

Keywords: multiobjective optimization · evolutionary algorithms · preference information · decision making · interactive methods · interactive preference incorporation

1 Introduction

Multiobjective optimization problems involve multiple conflicting objective functions that must be optimized simultaneously. Because of the conflict among the objective functions, these problems do not have a single optimal solution, but a set of trade-off solutions named a Pareto optimal set. The goal of solving a multiobjective optimization problem is to help a decision maker (DM) find the

most preferred trade-offs among objectives. A DM is a person with expertise about the problem and is usually interested in a subset of solutions that meets their preferences, known as a region of interest.

Methods for solving multiobjective optimization problems can be classified according to the role of the DM in the solution process into no preference, *a priori*, interactive, and *a posteriori* methods [22]. No preference methods are utilized when no DM is available and the problem is solved without considering any preference information. *A priori* methods ask for preference information once at the beginning of the solution process. On the other hand, *a posteriori* methods generate multiple solutions representing Pareto optimal ones and consider the preference information afterward. In interactive methods, the DM can provide preference information iteratively, allowing them to direct the solution process progressively. When studying interactive solution processes of DMs, one can often observe two phases: learning and decision phases, as stated in [21]. The DM explores different solutions during the learning phase until they find a region of interest. Then, in the decision phase, the DM fine-tunes the search to find the most preferred solution in that region.

Several scalarization-based methods [22] and evolutionary algorithms [7] have been proposed to solve multiobjective optimization problems. Multiobjective evolutionary algorithms (MOEAs) are population-based metaheuristics capable of representing the Pareto optimal set with approximated solutions. MOEAs can be divided into three main classes [29]: dominance-based, indicator-based, and decomposition-based algorithms. Decomposition-based MOEAs [14] have recently gained researchers' attention because of their scalability in terms of the number of objectives. These MOEAs decompose the original multiobjective optimization problem into multiple single-objective optimization problems or simpler multiobjective optimization problems to be solved collaboratively with the use of a scalarizing function and a set of so-called reference vectors. Decomposition-based MOEAs are suitable for preference incorporation as they can easily focus on certain parts of the Pareto optimal set by modifying the decomposition. MOEAs have been typically utilized as *a posteriori* methods. Although some interactive decomposition-based MOEAs are available in the literature (e.g. [12, 11, 3]), most of them focus only on the learning phase and identifying the region of interest. In other words, they do not consider a decision phase to help the DM find the most preferred solution.

In this article, we propose a general architecture for developing interactive decomposition-based MOEAs that address the needs of a decision maker. Our proposal consists of multiple modules that can be utilized to convert *a priori* and *a posteriori* methods into interactive ones. Each module contains different procedures that some interactive MOEAs have employed in the literature. In addition, new procedures can be incorporated into each one of the modules. The rest of the article is structured as follows. Section 2 presents background information on the main concepts used in the article. Then, a brief review of the existing interactive decomposition-based MOEAs is presented in Section 3. Section 4 describes some desirable properties of an interactive solution process.

Then, we present the proposed architecture to meet the desirable properties of interactive decomposition-based MOEAs in Section 5. As a proof of concept, we present some results and an algorithmic comparison in Section 6. We conclude the article in Section 7.

2 Background

A multiobjective optimization problem minimizing k (with $k \geq 2$) conflicting objective functions f_i $(i = 1, \ldots, k)$ can be mathematically formulated as follows:

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

where $S \subset \mathbb{R}^n$ is the feasible set of decision vectors $\mathbf{x} = (x_1, ..., x_n)^T$ with n decision variables. For every feasible decision vector $\mathbf{x}$, there is a corresponding objective vector $\mathbf{F}(\mathbf{x})$. In some problems, feasible decision vectors have to satisfy equality constraints and inequality constraints. Because of the conflict among the objective functions in (1), not all of them can achieve their optimal values simultaneously. A solution $\mathbf{x}^1 \in S$ dominates a solution $\mathbf{x}^2 \in S$ if and only if $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2)$ for all $i = 1, \ldots k$, and $f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$ for at least one index $j = 1, \ldots, k$. Then, a solution $\mathbf{x}^* \in S$ is Pareto optimal if and only if there is no solution $\mathbf{x} \in S$ that dominates it. A Pareto optimal set is then formed by all Pareto optimal solutions, and the corresponding objective vectors compose a Pareto front.

An ideal $\mathbf{z}^*$ and a nadir $\mathbf{z}^{nad}$ point represent the best and worst objective function values in the Pareto front, respectively. The ideal point can be calculated by minimizing each objective function separately. Calculating the nadir point is usually difficult since it requires computing the entire Pareto optimal set. However, it can be approximated using a pay-off table [22] or other means [9].

Decomposition-based MOEAs [14] utilize a set of reference vectors (which are also known as reference points or weight vectors) [1] to decompose the original multiobjective optimization problem into a set of single-objective optimization problems or simpler multiobjective optimization problems to be solved collaboratively. Usually, in the initialization of decomposition-based MOEAs, a set of reference vectors uniformly distributed in the objective space is generated utilizing e.g. a simplex lattice design [6]. This method requires a parameter p to control the density of the reference vectors. Then, the total amount of reference vectors is given by $\binom{p+k-1}{k-1}$. A scalarizing function is utilized to evaluate the solutions belonging to a part of the objective space. The solutions then evolve in the direction of the reference vector associated with such a part. Scalarizing functions map an objective vector to a real-valued scalar. Examples of decomposition-based MOEAs are MOEA/D [30], RVEA [5], and NSGA-III [8] which utilize dominance in combination with decomposition.

[1] For simplicity, we will utilize the term reference vectors throughout this article

In interactive methods, the DM provides preference information iteratively. Iterations are intervals during which MOEAs ask for preference information from the DM. They typically occur every GEN generations, where GEN is a parameter set before the method start. It is worth noting that a DM can provide preference information in multiple ways [4, 20]. Reference points are a common way of representing preference information in MOEAs [4]. A reference point $\mathbf{z}^{ref}$ is a k-dimensional vector consisting of a desirable value for each objective function.

3 Related Works

Some interactive decomposition-based MOEAs have been proposed in the literature. As stated in [4], most of the preference-based MOEAs are modifications of an existing *a posteriori* MOEA. We can classify interactive decomposition-based MOEAs according to how they accomplish interactivity. Although different types of preference information have been utilized in these methods, we consider here only the ones employing reference points.

The simplest way of imitating interactivity in MOEAs is by performing a series of *a priori* steps. However, the applicability of such methods in real-world problems is often not considered in the papers where they have been proposed, as some of their properties would significantly increase the DM's workload. For example, they usually do not let the DM decide when to interact with the method. In addition, they typically display an extensive set of solutions to be compared at each iteration. The interactive version of R-MOEA/D [25] is an example of an algorithm utilizing this structure.

Some methods modify the decomposition without altering the structure of the decomposition-based MOEA. Each iteration uses the preferences to update the decomposition and guide the search toward the region of interest. The most common modification to the decomposition involves rearranging the reference vectors according to the preference information [3, 12, 15, 19]. Some other methods utilize the preference information to modify the approximation of the ideal point required by the decomposition-based MOEA [23, 24]. The IOPIS framework [27] is another example in this category, as it creates a new (typically lower-dimensional) preference incorporated space (consisting of a set of scalarization functions) to reformulate the problem. It is worth noting that IOPIS can also be applied to other types of MOEAs (e.g., dominance-based and indicator-based); however, it has only been tested with decomposition-based methods. Although the structure of the methods in this category is similar to the methods in the previous category, these methods typically include mechanisms to ensure their applicability to real-world problems (e.g., considering a limited number of solutions to be shown to the DM at each iteration, controlling the frequency of iterations and the size of the region of interest).

Finally, some methods add additional steps for each generation of the decomposition-based algorithm for managing the preference information. Such steps are commonly intended to update the reference vectors inside the evolution-

ary process (and not before running the method as in the previous category). MOEA/D-a [31], MOEA/D-b [31], and MOEA/D-c [31] are examples of methods in this category. Interactive WASF-GA [26] is another example, as it replaces the dominance relation of NSGA-II by utilizing an achievement scalarizing function, which directs the search toward the region of interest.

4 Properties of an interactive solution process

In an interactive solution process, a DM iterates by providing preference information to the method and studying the received solutions until the most preferred one is found. A DM learns about the trade-offs among the objective functions as well as the feasibility of the preferences after each iteration. As a result, the DM may change the preference information during the solution process. To ensure the practical usability of the method, it should limit the level of cognitive burden and provide solutions that help the DM gain insight into the problem. Thus, we can summarize the main desirable properties of an interactive method as follows [2, 28]:

1. The method provides accurate information about possible solutions.
2. The DM and the method can communicate quite easily.
3. The method identifies and produces Pareto optimal solutions.
4. The method provides the DM with a clear overview of the Pareto optimal set/Pareto front.
5. The method enables the DM to find a region of interest in a learning phase.
6. The method has a decision phase to enable the DM to fine-tune the solutions in the region of interest.
7. The method gives the DM confidence that the final solution is the most preferred one, or at least close enough to it.

These properties are directly applicable to scalarization-based methods. Although MOEAs have to meet these properties, they have somewhat different needs and characteristics. Instead of producing Pareto optimal solutions, MOEAs can provide a set of non-dominated solutions, as they are metaheuristics and cannot guarantee optimality. In addition, most interactive MOEAs focus only on the learning phase, representing a region of interest without helping the DM select the most preferred solution. In the next section, we present an architecture for developing interactive decomposition-based MOEAs that meet the above-mentioned properties.

5 Proposed architecture

We propose an architecture consisting of multiple modules that can be utilized to generate interactive decomposition-based MOEAs that meet the properties discussed in the previous section. Also, *a posteriori* or *a priori* decomposition-based algorithms can be converted to interactive ones with the help of the architecture.

The architecture has two types of modules: static and dynamic. Static modules consist of multiple steps that must be considered during the solution process. On the other hand, dynamic modules allow us to personalize the method according to our needs. Such modules present multiple alternatives from which we can select one or multiple. This architecture aims to provide a guideline for developing new interactive decomposition-based MOEAs that consider the structure and properties of an interactive solution process. The alternatives presented in the dynamic modules have been selected after analyzing the structure of multiple interactive MOEAs. This means that the interested user can incorporate new options that accomplish the main aim of each module. The architecture is illustrated in Figure 1. The static modules have a red marker in the upper right corner of the corresponding box, while the dynamic modules have a green marker. The architecture has seven modules: initialization, preference elicitation, component adaptation, optimization, spread adjustment, selection of solutions, and iteration. Below, we give details of each module.

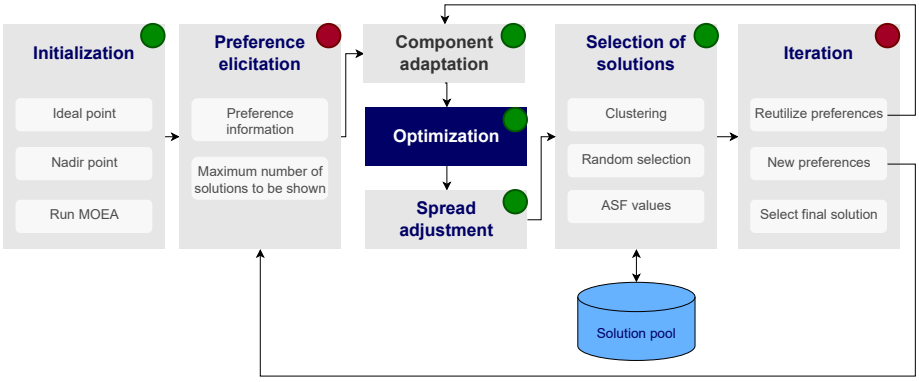

Fig. 1. Proposed architecture for developing interactive decomposition-based MOEAs.

Initialization module: This module provides a DM information for learning about feasible solutions to the problem before starting the solution process (property 1). The alternatives in this module are: computing the ideal point, estimating the nadir point, and running an MOEA for a fixed number of generations. Usually, showing the ideal and nadir points to the DM may help them provide more realistic preference information within the lower and upper bounds of the objective functions. If we want to show some feasible solutions to the DM before starting the solution process, running an *a posteriori* MOEA would be a good alternative. However, only a representative set of solutions should be displayed to the DM. E.g., a clustering method can be utilized to limit the number of solutions to display.

Preference elicitation module: This module retrieves the DM's preference information and the maximum number of solutions (Ns) they want to see at each

iteration (property 2). Here, we utilize reference points to represent the DM's preferences. However, we can extend the architecture to support more types of preference information. If an *a priori* method is employed in the optimization module, then the type of preference information is the same utilized by such a method. However, if the optimization module uses an *a posteriori* method, the preference information is selected according to the mechanism employed in the component adaptation module. The preference information provided during all the iterations is stored for further use.

Component adaptation module: This module is needed only when an *a posteriori* MOEA is utilized in the optimization module. It aims to modify some elements used by the decomposition-based MOEA to consider preference information. For example, modifying the distribution of the reference vectors, changing the problem formulation, or using the preferences information to replace the approximation of the ideal point required by some MOEAs (e.g., MOEA/D).

Optimization module: In this module, an MOEA is utilized to solve a multiobjective optimization problem considering the preference information provided by the DM (property 3). There are two alternatives for this module: utilizing an *a priori* MOEA or an *a posteriori* one. In both cases, the methods do not need to be modified.

Spread adjustment module: This module controls the size of the region of interest depending on the phase of the interactive solution process (properties 5 and 6). A higher spread value is utilized during the learning phase, as the aim is to learn and eventually find the region of interest. The value will be reduced during the decision phase to help the DM find the most preferred solution in the region of interest. The preference information stored in the preference elicitation module is used to identify the decision phase. If the preference information starts to be similar among multiple iterations, the decision phase has begun. Some *a priori* methods utilize a spread parameter. When using some of those methods in the optimization module, no additional procedures are needed to control the size of the region of interest, and the parameter is updated iteratively. On the other hand, if the optimization module employs an *a posteriori* method or *a priori* method that does not consider a spread parameter, a mechanism to select a subset of solutions from the region of interest is needed (e.g., the trimming procedure of the R-metric [13]).

Solution selection module: This module filters the solutions on the region of interest to show only a reduced set of Ns representative solutions to the DM (property 4). The solutions can be selected in multiple ways: randomly, dividing the solution set into Ns clusters and selecting the solutions closest to each centroid, or selecting the Ns solutions with the best values of a scalarizing function. In addition, this module stores the best solutions in an archive to avoid losing them. A scalarizing function can be utilized to determine which solutions to keep. These solutions can be used when the DM provides a reference point close to another one from a previous iteration.

Iteration module: In this module, the DM can decide whether or not to provide new preference information. When no new preference information is

available, the same values as in the previous iteration are utilized. The DM can also select the final solution among the ones displayed, in which case the solution process is finished (property 7).

We create an interactive decomposition-based MOEA in the following section to demonstrate how the proposed architecture can be used. In addition, we compare it with a method consisting of multiple *a priori* steps.

6 Example method and experiments

In this section, as a proof of concept, we demonstrate how a method created with the proposed architecture can support a DM during an interactive solution process. The method considered has the following configuration of modules:
Initialization: MOEA/D is run 200 generations. Then, the ideal and nadir points are estimated from the resulting population.
Preference elicitation: Reference points are utilized to represent the preferences of the DM. All the reference points provided during the solution process will be stored. A maximum of five solutions ($Ns = 5$) will be shown to the DM during each iteration.
Component adaptation: The reference vectors are rearranged utilizing the NUMS procedure [15]. It is worth noting that the adaptation is performed based on the initial values of the reference vectors (which are generated utilizing a simplex lattice design [6]).
Optimization: MOEA/D is utilized with 200 generations per each iteration.
Spread adjustment: We adapt the spread parameter of the NUMS procedure based on the differences among the reference points. Initially, the spread parameter is set as 0.5. If the DM utilizes a reference point close to another one provided in a previous iteration, the spread parameter is divided by two.
Selection of solutions: The five most representative solutions are shown to the DM. These are obtained by clustering the solution set utilizing the k-means method [18]. These solutions will be stored in the solution pool if other solutions do not dominate them. In addition, the dominated solutions are removed from the pool every time a new solution is incorporated.
Iteration: When a new reference point is provided, the solutions in the pool are included in the initial population of MOEA/D. If the DM does not provide new preferences, the optimization module is run again without modifying the population or the reference point. The process only stops if the DM has found a preferred solution.

6.1 Interactive solution process

In what follows, we refer to the method created with our architecture as MOEA/D-NUMS+, while the one consisting of multiple *a priori* steps is called MOEA/D-NUMS. We conduct an experiment with the crash-worthiness design of vehicles problem [17]. It is a real-world engineering problem whose goal is to make vehicles crash-safe. During a collision, the frontal structure of the vehicle absorbs energy,

which increases passenger safety. Increasing the mass of the vehicle generally improves its energy absorption capacity. By contrast, lightweight materials are necessary for a vehicle's mass to be reduced and, therefore, its fuel consumption. To achieve a proper design, we must find a compromise between higher energy absorption and lightweight construction. In this problem, five decision variables are used to represent the thickness of five reinforced components surrounding the frontal structure of the vehicle. Specifically, three objective functions need to be minimized: 1) the mass of a vehicle; 2) deceleration during full-frontal crashes (which influence passenger injuries); and 3) toe board intrusion during offset-frontal crashes (which affect the structural integrity of the vehicle). It is worth noting that we will take the role of the DM in this experiment, as its main aim is to exemplify how the method works. Experiments with real DMs will be considered as future work.

Now, we can describe the iterations of the solution process. At the beginning, the ideal and nadir points were shown to the DM, whose values are $\mathbf{z}^* = (1661.708, 6.986, 0.0708)$ and $\mathbf{z}^{nad} = (1666.211, 8.304, 0.106)$.

Iteration 1. First, the DM set the ideal point as the reference point to see how difficult it is to achieve these promising values. The five solutions (obtained after applying the clustering method) displayed to the DM are shown in Table 1. It is worth noting that the solutions are sorted in increasing order of f_1. However, the DM should be able to decide how to see the solutions displayed by the method (e.g., in an increasing or decreasing order of some of the objectives).

Table 1. Results of the first iteration

	Mass (kg)	Deceleration (m/s)	Intrusion (m)
1	1662.032	8.209	0.078
2	1662.420	8.095	0.086
3	1663.010	7.923	0.096
4	1663.839	7.680	0.104
5	1665.229	7.273	0.104

Table 2. Results of the second iteration

	Mass (kg)	Deceleration (m/s)	Intrusion (m)
1	1662.007	8.216	0.078
2	1662.077	8.196	0.079
3	1662.152	8.174	0.081
4	1662.233	8.150	0.082
5	1662.342	8.118	0.085

Iteration 2. Since the reference point had been too optimistic, the DM adjusted its components to be more realistic but focused on improving the value of the first objective. The new reference point was $(1661.9, 8.0, 1.0)$, and the obtained five solutions are shown in Table 2.

Iteration 3. Based on the solutions shown, the DM realized the trade-off between f_1 and f_2 and provided a new reference point $(1665, 7.4, 0.1)$ with the aim of obtaining better values for f_2. The results obtained are shown in Table 3. There was a good improvement on f_2, but f_1 and f_3 impaired.

Iteration 4. After noticing the impairment in f_1 and f_3, the DM decided to get similar results to the ones of Iteration 2. For this, the DM did not need to provide a new reference point, but it was taken from the list of reference points utilized during the solution process. In addition, the spread of the region of interest was reduced automatically. The results obtained are shown in Table 4.

Table 3. Results of the third iteration

	Mass (kg)	Deceleration (m/s)	Intrusion (m)
1	1663.181	7.873	0.098
2	1663.584	7.755	0.102
3	1664.070	7.612	0.105
4	1664.581	7.463	0.107
5	1664.999	7.340	0.106

Table 4. Results of the fourth iteration

	Mass (kg)	Deceleration (m/s)	Intrusion (m)
1	1662.016	8.214	0.078
2	1662.060	8.201	0.079
3	1662.106	8.187	0.080
4	1662.163	8.171	0.081
5	1662.221	8.154	0.082

Iteration 5. The DM was satisfied with the improvement on f_1 and f_3 and wanted to find more solutions in the same region. Thus, the DM kept the same reference point as the previous iteration, and the algorithm automatically reduced the spread of the region of interest. The results are shown in Table 5.

Iteration 6. The DM noticed that the solutions were refined and was satisfied with the fifth one. The DM selected it as the final solution, as its value on f_2 was better than the rest without losing too much on f_1 and f_3.

Table 5. Results of the fifth iteration

	Mass (kg)	Deceleration (m/s)	Intrusion (m)
1	1662.034	8.208	0.078
2	1662.054	8.203	0.079
3	1662.074	8.197	0.079
4	1662.097	8.190	0.080
5	**1662.123**	**8.182**	**0.080**

Now we summarize the advantages of our method compared to another one consisting of multiple *a priori* steps. Providing the ideal and nadir points before starting the solution process can help the DM give a reference point when the DM does not have a clear idea of the feasibility of the solutions, as was shown in iteration 1. The solutions provided in each iteration can be easily compared to identify trade-offs between objectives, as there is only a reduced number of representative solutions. We took advantage of this property on iterations 3 and 4. This reduces the cognitive burden of the DM compared with a method consisting of multiple *a priori* steps. As we stored the preferences provided during the solution process, the DM can easily re-utilize a previous reference point, as was shown in iteration 4. Then, the spread adjustment allowed us to refine the solutions when reaching the decision phase in iterations 5 and 6. If we had used a method consisting of *a priori* steps, the support during the solution process would not be enough for allowing the DM to learn about the problem trade-offs and the feasibility of solutions.

6.2 Algorithmic comparison

In the previous section, we showed how MOEA/D-NUMS+ could support a DM during an interactive solution process. In this section, we compare the performance of MOEA/D-NUMS+ and MOEA/D-NUMS utilizing the artificial deci-

sion maker (ADM) proposed in [1] (to replace the DM), and the R-IGD performance indicator [16]. The ADM adjusts the preference information according to the insight gained during each iteration, producing reference points differently depending on the phase of the solution process. The generated reference points simulate the exploration in the Pareto optimal set during the learning phase. On the other hand, the reference points mimic a progressive convergence on the region of interest obtained from the learning phase during the decision phase. We considered 4 iterations for the learning phase ($L = 4$) and 3 for the decision phase ($D = 3$) in this experiment. For each iteration, ADM computes the R-IGD for the results obtained by each method. After the run, the cumulative R-IGD for the learning phase is obtained by adding the R-IGD values of the first L iterations. The cumulative R-IGD for the decision phase is obtained by adding the R-IGD values of the last D iterations. The methods were tested utilizing the same parameters for both of them: 50 generations per iteration, a lattice resolution of 5, 0.7 as a spread parameter during the learning phase, and 0.3 during the decision phase. We did not employ the spread adjustment procedure in this experiment, as the ADM controls the value of the spread parameter according to the phase of the solution process. We considered two benchmark problems: DTLZ1 and DTLZ3 [10] with 4, 7, and 9 objectives, resulting in six different problems. The number of variables was set as $10 + k - 1$ [10]. We made ten independent runs for each problem. The median R-IGD values of MOEA/D-NUMS and MOEA/D-NUMS+ are shown in Table 6. The best results are highlighted in **boldface**.

Table 6. R-IGD values for DTLZ1 and DTLZ3 with 7, 5, and 9 objectives.

Problem	k	Phase	MOEA/D-NUMS		MOEA/D-NUMS+	
			Median	Std. dev.	Median	Std. dev
DTLZ1	4	Learning	2.804056	0.174749	**2.671504**	0.236763
		Decision	**2.722246**	0.283701	2.74031	0.269508
	7	Learning	4.663774	0.126492	**3.523776**	0.16947
		Decision	3.317874	0.210076	**3.046718**	0.515918
	9	Learning	4.332031	0.195637	**3.011431**	0.268385
		Decision	3.36567	0.21437	**3.113235**	0.658471
DTLZ3	4	Learning	0.451812	0.039745	**0.377324**	0.031323
		Decision	**1.472565**	0.019187	1.497671	0.000519
	7	Learning	**0.803064**	0.922071	1.958682	1.522243
		Decision	0.383067	0.429625	**0.234077**	0.122247
	9	Learning	1.182198	0.448281	**0.754516**	0.908085
		Decision	0.81725	0.299627	**0.377474**	0.111261

For DTLZ1, MOEA/D-NUMS+ outperformed MOEA/D-NUMS in most of the cases, except during the decision phase when four objectives were considered. In the DTLZ3 problem, MOEA/D-NUMS had a better performance than the proposed method only in the learning phase with seven objectives and the decision phase with four objectives. This experiment showed us that the proposed architecture can help in improving the quality of the solutions obtained during the learning and decision phase. However, more extensive experimentation considering different types of problems is needed.

6.3 Discussion

We utilized the proposed architecture to create an interactive version of MOEA/D. To this aim, we employed a procedure for re-arranging the reference vectors utilized in the optimization method. However, our proposal can also be applied to *a priori* methods. We do not need to use the component adaptation module in such a case, as the *a priori* method internally modifies the decomposition-based MOEA to handle the preference information. It is worth noting that although we only utilized reference points in this article, our proposal can be extended to other types of preference information. The type of preference information in the current architecture is related to the one required by the component adaptation and/or optimization modules. To handle more types of preferences, we would need an additional module for preference unification. Such a module should allow the DM to use any kind of preference information and then transform it into the one required by the rest of the modules. In addition, there are some methods that do not consider a spread parameter to control the size of the region of interest. In this case, we can utilize an external method like the one considered in the R-metric [16]. In this article, we did not consider the case where the same value of the spread parameter can be utilized in multiple iterations. Such behavior can be useful when the MOEA needs more generations to converge to the Pareto optimal set. However, the DM is usually not aware of the technical details of the method. Finally, the selection of the solutions to be shown to the DM can be performed in multiple ways, for example, through different clustering techniques or scalarizing functions.

7 Conclusions

Multiple interactive versions of decomposition-based MOEAs have been proposed in the literature, but they typically do not consider the DM's needs and cognitive load. We have introduced an architecture to create interactive decomposition-based MOEAs by integrating multiple modules with existing *a priori* or *a posteriori* methods. To demonstrate how our architecture can be employed, we created an interactive MOEA utilizing NUMS, a method for rearranging the reference vectors which has been mainly to convert an *a posteriori* method into an *a priori* one. We solved a real-world problem to demonstrate the advantages of using our proposal for improving the applicability of the methods and reducing the cognitive load of the DM. In addition, we compared the proposed method with another one (which does not include the properties of the architecture) consisting of multiple *a priori* steps. According to the results, utilizing the proposed architecture improves the performance in most test problems utilized. This is the first step toward improving the performance of interactive decomposition-based MOEAs. The proposed architecture can be improved in multiple directions, for example, by including a preference unification module to consider different types of preference information and also developing different methods for adapting the spread parameter. In addition, it can be extended to other types of MOEAs (e.g., dominance- and indicator-based MOEAs).

References

1. Afsar, B., Miettinen, K., Ruiz, A.B.: An artificial decision maker for comparing reference point based interactive evolutionary multiobjective optimization methods. In: Ishibuchi, H., Zhang, Q., Cheng, R., Li, K., Li, H., Wang, H., Zhou, A. (eds.) Evolutionary Multi-Criterion Optimization, 11th International Conference, EMO 2021, Proceedings. pp. 619–631. Springer (2021)
2. Afsar, B., Miettinen, K., Ruiz, F.: Assessing the performance of interactive multiobjective optimization methods: A survey. ACM Computing Surveys **54**(4) (2021). https://doi.org/10.1145/3448301, https://doi.org/10.1145/3448301
3. Aghaei Pour, P., Rodemann, T., Hakanen, J., Miettinen, K.: Surrogate assisted interactive multiobjective optimization in energy system design of buildings. Optimization and Engineering **23**, 303–327 (2022). https://doi.org/10.1007/S11081-020-09587-8, https://link.springer.com/article/10.1007/s11081-020-09587-8
4. Bechikh, S., Kessentini, M., Said, L.B., Ghédira, K.: Chapter four - preference incorporation in evolutionary multiobjective optimization: A survey of the state-of-the-art. Advances in Computers, vol. 98, pp. 141–207. Elsevier (2015). https://doi.org/https://doi.org/10.1016/bs.adcom.2015.03.001, http://www.sciencedirect.com/science/article/pii/S0065245815000273
5. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. IEEE Transactions on Evolutionary Computation **20**(5), 773–791 (2016). https://doi.org/10.1109/TEVC.2016.2519378
6. Cornell, J.A.: Experiments with Mixtures: Designs, Models, and the Analysis of Mixture Data. John Wiley & Sons (2011)
7. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
8. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. IEEE Transactions on Evolutionary Computation **18**(4), 577–601 (2014). https://doi.org/10.1109/TEVC.2013.2281535
9. Deb, K., Miettinen, K., S., C.: Towards an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. IEEE Transactions on Evolutionary Computation **14**(6), 821–841 (2010)
10. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization, pp. 105–145. Springer London, London (2005). https://doi.org/10.1007/1-84628-137-7_6, https://doi.org/10.1007/1-84628-137-7_6
11. Gong, M., Liu, F., Zhang, W., Jiao, L., Zhang, Q.: Interactive MOEA/D for multiobjective decision making. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary computation - GECCO '11. ACM, New York (2011)
12. Hakanen, J., Chugh, T., Sindhya, K., Jin, Y., Miettinen, K.: Connections of reference vectors and different types of preference information in interactive multiobjective evolutionary algorithms. In: 2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016. Institute of Electrical and Electronics Engineers Inc. (2017). https://doi.org/10.1109/SSCI.2016.7850220
13. Li, K., Deb, K., Yao, X.: R-metric: Evaluating the performance of preference-based evolutionary multi-objective optimization using reference points. IEEE Transactions on Evolutionary Computation pp. 1–1 (2018). https://doi.org/10.1109/TEVC.2017.2737781

14. Li, K.: Decomposition multi-objective evolutionary optimization: From state-of-the-art to future opportunities. CoRR **abs/2108.09588** (2021), https://arxiv.org/abs/2108.09588
15. Li, K., Chen, R., Min, G., Yao, X.: Integration of preferences in decomposition multiobjective optimization. IEEE Transactions on Cybernetics **48**(12), 3359–3370 (2018). https://doi.org/10.1109/TCYB.2018.2859363
16. Li, K., Deb, K., Yao, X.: R-metric: Evaluating the performance of preference-based evolutionary multiobjective optimization using reference points. IEEE Transactions on Evolutionary Computation **22**(6), 821–835 (2018). https://doi.org/10.1109/TEVC.2017.2737781
17. Liao, X., Li, Q., Yang, X., Zhang, W., Li, W.: Multiobjective optimization for crash safety design of vehicles using stepwise regression model. Structural and Multidisciplinary Optimization **35**(6), 561–569 (Nov 2008). https://doi.org/10.1007/s00158-007-0163-x, https://doi.org/10.1007/s00158-007-0163-x
18. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297. Oakland, CA, USA (1967)
19. Mazumdar, A., Chugh, T., Hakanen, J., Miettinen, K.: An interactive framework for offline data-driven multiobjective optimization. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 97–109. Springer (2020). https://doi.org/10.1007/978-3-030-63710-1_8, https://doi.org/10.1007/978-3-030-63710-1_8
20. Miettinen, K., Hakanen, J., Podkopaev, D.: Interactive nonlinear multiobjective optimization methods. In: Greco, S., Ehrgott, M., Figueira, J. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys, pp. 931–980. Springer, 2 edn. (2016)
21. Miettinen, K., Ruiz, F., Wierzbicki, A.: Introduction to multiobjective optimization: interactive approaches. In: Branke, J., Deb, K., Miettinen, K., Slowinski, R. (eds.) Multiobjective Optimization: Interactive and Evolutionary Approaches, pp. 27–57. Springer, Berlin, Heidelberg (2008)
22. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)
23. Nguyen, L., Bui, L.T.: A multi-point interactive method for multi-objective evolutionary algorithms. In: Proceedings - 4th International Conference on Knowledge and Systems Engineering, KSE 2012. pp. 107–112 (2012). https://doi.org/10.1109/KSE.2012.30
24. Nguyen, L., Duc, D.N., Thanh, H.N.: An Enhanced Multi-point Interactive Method for Multi-objective Evolutionary Algorithms. In: Advances in Intelligent Systems and Computing. vol. 1013, pp. 42–49. Springer (2020). https://doi.org/10.1007/978-981-32-9186-7_5, https://link.springer.com/chapter/10.1007/978-981-32-9186-7_5
25. Qi, Y., Li, X., Yu, J., Miao, Q.: User-preference based decomposition in MOEA/D without using an ideal point. Swarm and Evolutionary Computation **44**, 597–611 (2019). https://doi.org/10.1016/j.swevo.2018.08.002
26. Ruiz, A.B., Luque, M., Miettinen, K., Saborido, R.: An Interactive Evolutionary Multiobjective Optimization Method: Interactive WASF-GA. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9019**, 249–263 (2015). https://doi.org/10.1007/978-3-319-15892-1_17, https://link.springer.com/chapter/10.1007/978-3-319-15892-1_17

27. Saini, B.S., Hakanen, J., Miettinen, K.: A new paradigm in interactive evolutionary multiobjective optimization. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 12270 LNCS, pp. 243–256. Springer Science and Business Media Deutschland GmbH (2020). https://doi.org/10.1007/978-3-030-58115-2_17, https://doi.org/10.1007/978-3-030-58115-2_17
28. Thiele, L., Miettinen, K., Korhonen, P.J., Molina, J.: A preference-based evolutionary algorithm for multi-objective optimization. Evolutionary Computation **17**(3), 411–436 (2009). https://doi.org/10.1162/evco.2009.17.3.411
29. Zhang, J., Xing, L.: A survey of multiobjective evolutionary algorithms. In: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). vol. 1, pp. 93–100 (2017). https://doi.org/10.1109/CSE-EUC.2017.27
30. Zhang, Q., Li, H.: Moea/d: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**(6), 712–731 (2007). https://doi.org/10.1109/TEVC.2007.892759
31. Zheng, J., Yu, G., Zhu, Q., Li, X., Zou, J.: On decomposition methods in interactive user-preference based optimization. Applied Soft Computing Journal **52**, 952–973 (2017). https://doi.org/10.1016/j.asoc.2016.09.032