Complex & Intelligent Systems
https://doi.org/10.1007/s40747-021-00586-5

ORIGINAL ARTICLE

Comparing interactive evolutionary multiobjective optimization methods with an artificial decision maker

Bekir Afsar[1] · **Ana B. Ruiz**[2] · **Kaisa Miettinen**[1]

Received: 5 July 2021 / Accepted: 12 October 2021

Abstract
Solving multiobjective optimization problems with interactive methods enables a decision maker with domain expertise to direct the search for the most preferred trade-offs with preference information and learn about the problem. There are different interactive methods, and it is important to compare them and find the best-suited one for solving the problem in question. Comparisons with real decision makers are expensive, and artificial decision makers (ADMs) have been proposed to simulate humans in basic testing before involving real decision makers. Existing ADMs only consider one type of preference information. In this paper, we propose ADM-II, which is tailored to assess several interactive evolutionary methods and is able to handle different types of preference information. We consider two phases of interactive solution processes, i.e., learning and decision phases separately, so that the proposed ADM-II generates preference information in different ways in each of them to reflect the nature of the phases. We demonstrate how ADM-II can be applied with different methods and problems. We also propose an indicator to assess and compare the performance of interactive evolutionary methods.

Keywords Decision making · Preferences · Performance comparison · Many-objective optimization · Interactive methods

Introduction

Multiobjective optimization problems refer to optimizing multiple conflicting objectives and arise in many application areas such as engineering, economy, or industry. Usually, no solution exists in which all objectives achieve their individual optima at the same time. Instead, there exists a set of the so-called *Pareto optimal solutions*, at which an improvement in one objective is only possible at the expense of getting worse values in, at least, one of the others. All Pareto optimal solutions form the *Pareto optimal set*.

Pareto optimal solutions are mathematically incomparable, so additional preference information, usually coming from a *decision maker* (DM), who is an expert in the problem domain, is required to identify the *most preferred solution* (MPS) as the final solution. The role of the DM in the solution process varies depending on the type of multiobjective optimization method [19]. In *a priori methods*, the DM expresses her/his preferences before the solution process starts, while in *a posteriori methods*, preferences are used in selection once a representative set of Pareto optimal solutions has been generated. On the contrary, solution processes with *interactive methods* consist of *iterations*, where the DM is actively involved by directing the search with preference information. (S)he iteratively sees information about the solutions available, and expresses and fine-tunes or even changes her/his preference information at each iteration until (s)he is satisfied with some of the solutions.

The main benefit of interactive methods is that the DM can learn which types of solutions are feasible without dealing with large amounts of data at once. At the same time, (s)he can progressively adjust one's preferences based on the gained insight into the problem. Actually, in an interactive solution process, we can often distinguish two phases [21]: a learning phase, where the DM explores different solutions to find a *region of interest* (ROI) formed by the Pareto optimal solutions that satisfy her/him the most; and a decision phase,

✉ Bekir Afsar
bekir.b.afsar@jyu.fi

Ana B. Ruiz
abruiz@uma.es

Kaisa Miettinen
kaisa.miettinen@jyu.fi

[1] University of Jyvaskyla, Faculty of Information Technology, FI-40014 University of Jyvaskyla, Finland

[2] Department of Applied Economics (Mathematics), Universidad de Málaga, C/ Ejido 6, 29071 Málaga, Spain

Published online: 04 December 2021

where (s)he fine-tunes the search within this ROI to select her/his MPS.

Over the years, many interactive methods have been proposed [18–20]. Among them, interactive *evolutionary multiobjective optimization* (EMO) methods [4] are suitable for problems with, e.g., discontinuous and non-differentiable functions. EMO methods incorporate preference information into an evolutionary process to generate a population of solutions approximating the ROI best by reflecting the given preferences [18,29]. To find a suitable method among the many alternatives available, we must test and compare different interactive methods to understand their potential to fulfill different needs of the solution process successfully. However, the quantitative assessment of interactive methods involving real DMs is not a trivial task due to several reasons [2,16].

It is expensive to involve many DMs with the appropriate domain expertise to run a sufficient amount of tests. Naturally, DMs learn about the problem during the interactive solution process, and thus, the order in which methods are applied affects their performance. To compensate for this, we would need an even higher number of DMs. A survey of published comparisons of interactive methods is given in [2], where the need for characterizing desirable properties of interactive methods is emphasized. Overall, the survey supports the need for improved means for comparing interactive methods.

Overall, it is hard and time-consuming to design experiments with human DMs to compare different interactive methods due to their subjectivity, learning of the problem, human fatigue, and other limiting factors. However, to some degree, comparisons can be conducted without humans. According to [26], we can divide interactive methods into *non ad-hoc* and *ad-hoc* ones depending on whether a value function can be used to replace the DM or not, respectively. If the preference information used in the method cannot be derived from a value function, we need different means for assessing interactive methods, and this is the focus of this paper. So-called *artificial DMs* (ADMs) have been introduced in the literature to replace the DM and run ad-hoc methods conveniently. However, they are suited for methods involving a single type of preference information.

To the best of our knowledge, there are only a few ADMs to compare interactive methods: [1,3,13,24]. The one suggested in [13] simulates the learning of a DM by progressively narrowing the angle of a cone, which is defined based on a pre-defined MPS. In [3,24], ADMs are proposed for comparing reference point-based interactive methods (the former directed at EMO methods). Both of them consist of a predefined steady part that includes an aspiration point initially set (formed by aspiration levels for the objectives), which the solution process must converge to and which remains unchanged, and a current context that evolves based on the knowledge gained about the problem during the solution process. Note that these ADMs are based on a goal point set initially (an MPS in [13] and an aspiration point in [3,24]) and their performance highly relies on this point. These ADMs run each of the methods individually, which means that the preferences used at each iteration with each of them are different, since they are generated based on the output of every single method.

We focus here on ADMs for comparing interactive EMO methods of ad-hoc type. We extend our previous ADM [1], which was tailored to methods applying reference points, to compare interactive EMO methods applying different types of preference information. We call it ADM-II. Both ADMs are designed to run all the methods to be compared simultaneously using similar preference information at each iteration. Furthermore, these are the first ADMs that generate preferences depending on the phase (learning or decision) of the interactive solution process to allow a better analysis of the performance in each phase.

As said, the main novelty of ADM-II is its ability to generate different types of preference information, thus clearly extending the scope of the existing ADMs. Besides a reference point, the following types of preferences can be generated: selecting one or several preferred solution(s) or non-preferred solution(s) among a set of alternatives, specifying preferred ranges for the objective functions, and performing pairwise comparisons among solutions. The further novelty lies in the way preference information is generated in the decision phase (compared to [1]).

At each iteration, ADM-II generates preference information based on the solutions obtained so far by all methods that are compared. In this way, we adapt the preferences to the insight gained during the solution process. To perform a fair comparison, the same computational resources (i.e., number of function evaluations or generations per iteration) are internally assigned to each interactive EMO method compared.

To evaluate the performance of a method, we must measure the quality of the solutions produced at each iteration taking into account the preferences, and we propose a performance indicator for this purpose. This indicator counts the number of nondominated solutions which are in a composite front consisting of the nondominated solutions of populations of all compared methods. This composite front is updated, while ADM-II is performing iterations. Our indicator measures the number of nondominated solutions with which each method has contributed to building the composite front. It indicates the exploratory potential of each method and its adaptation capacity to the changes in the preference information in each phase. Furthermore, in the decision phase, the quality (i.e., Pareto optimality) of the final MPS reached can be evaluated using, e.g., an achievement scalarizing function [28], which provides a measurement of each method's convergence capability. It is important to note that the quality

of the results obtained depends on the methods themselves, not on ADM-II.

To summarize, our main contribution is proposing ADM-II to provide a computational tool to gain deeper knowledge about the performance of different interactive EMO methods without involving human DMs. This means that many repetitions can be done in stable conditions. Unlike [1], our ADM is able to compare methods using different types of preferences, which has not been done earlier in the literature. Thus, it can be used to test several interactive methods. We do not claim that ADM-II can investigate all human biases that can affect decision making, but ADMs provide good means for finding viable candidate methods that can be further tested with humans or directly applied to solve the problem in question.

In addition, we demonstrate how ADM-II can be applied by comparing some interactive EMO methods with a set of benchmark problems with up to nine objectives. Besides the new indicator, we also report the number of function evaluations used and apply a quality indicator developed for a priori EMO methods.

The rest of the paper is organized as follows. We present the background concepts of multiobjective optimization in "Background concepts". "Artificial decision maker for interactive EMO" gives a detailed description of the proposed ADM-II while, in "Computational experiments", we demonstrate the performance of ADM-II comparing several interactive EMO methods applying different types of preference using benchmark problems. Finally, we conclude and mention future research directions in "Conclusions".

Background concepts

In general, a *multiobjective optimization problem* can be formulated in the following form:

$$\begin{aligned} &\text{minimize } \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ &\text{subject to } \mathbf{x} \in S, \end{aligned} \tag{1}$$

where $f_i : S \to \mathbb{R}$ are the k conflicting objective functions (with $k \geq 2$) to be optimized at the same time. The *decision vectors* $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ belong to the feasible set $S \subset \mathbb{R}^n$, whose images in the objective space, denoted by $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T$, are called *objective vectors*.

Usually, the conflict degree among the objectives makes it impossible to find a solution where all the objectives can reach their individual optimum. Therefore, we are interested in the so-called Pareto optimal solutions, at which no objective function value can be improved without impairing, at least, one of the others. Given $\mathbf{z}^1, \mathbf{z}^2 \in \mathbb{R}^k$, we say that $\mathbf{z}^1$ *dominates* $\mathbf{z}^2$ if $z_i^1 \leq z_i^2$ for all $i = 1, 2, \ldots, k$ and $z_j^1 < z_j^2$ for, at least, one index j. If $\mathbf{z}^1$ and $\mathbf{z}^2$ do not dominate each other, they are *(mutually) nondominated*. Furthermore, a decision vector $\mathbf{x}^* \in S$ is *Pareto optimal* if there does not exist another $\mathbf{x} \in S$, such that $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{f}(\mathbf{x}^*)$. The corresponding objective vector $\mathbf{f}(\mathbf{x}^*)$ is called a *Pareto optimal objective vector*. The set formed by all Pareto optimal solutions is called the *Pareto optimal set*, denoted by E, and its image in the objective space is referred to as the *Pareto optimal front*, denoted by PF. Since we deal here with EMO methods, they cannot guarantee Pareto optimality, but we deal with nondominated solutions approximating Pareto optimal ones.

The ranges of the objective function values in the PF are defined by the ideal and the nadir points. The *ideal point* $\mathbf{z}^\star = (z_1^\star, \ldots, z_k^\star)^T$ is obtained by $z_i^\star = \min_{\mathbf{x} \in S} f_i(\mathbf{x}) = \min_{\mathbf{x} \in E} f_i(\mathbf{x})$ $(i = 1, \ldots, k)$ and contains the lowest objective function values. The *nadir point* $\mathbf{z}^{\text{nad}} = (z_1^{\text{nad}}, \ldots, z_k^{\text{nad}})^T$ can be defined as $z_i^{\text{nad}} = \max_{\mathbf{x} \in E} f_i(\mathbf{x})$ $(i = 1, \ldots, k)$ and is formed by the highest (i.e., the worst) objective function values between Pareto optimal solutions. In practice, the nadir point is usually approximated, since its computation is difficult as the set E is unknown (see, e.g., [9,19,27] and references therein). Alternatively, the DM can also be asked for the worst possible objective function values and consider them as the components of the nadir point.

There are different ways of expressing preferences [17, 19,25]. The options available for expressing preferences in the ADM-II that we propose here are the following:

- Giving a *reference point* $\mathbf{q} = (q_1, \ldots, q_k)^T$, where each q_i is a desirable aspiration value for the objective function f_i $(i = 1, \ldots, k)$.
- Selecting p (with $p \geq 1$) solutions as the most preferred ones among a set of solutions. Let us denote them by $\mathbf{PS}_1, \ldots, \mathbf{PS}_p$.
- Selecting np (with $np \geq 1$) solutions as the most non-preferred (unacceptable) ones among a set of solutions. Let us denote them by $\mathbf{NPS}_1, \ldots, \mathbf{NPS}_{np}$.
- Specifying preferred ranges with desirable values for the objective functions. We denote by $[f_i^l, f_i^u]$ the *preferred range* for the objective function f_i $(i = 1, \ldots, k)$. As a result, the preferences are determined by a k-dimensional hyper-box $[f_1^l, f_1^u] \times \cdots \times [f_k^l, f_k^u]$ in the objective space.
- Performing *pairwise comparisons of solutions*, i.e., given two solutions, the DM decides which one satisfies her/him the most.

In the literature, there exists a plethora of interactive methods for solving multiobjective optimization problems (surveyed, e.g., in [4,18–20,29]). As mentioned in the introduction, the solution process with interactive methods can often be observed to have two phases aimed at different purposes [21]. First, in the learning phase, the DM explores the problem to learn about the conflict degree among the

objectives and what kind of solutions are feasible reflecting different preferences. At the end of this phase, an ROI is identified according to the DM's desires. Second, in the decision phase, (s)he further explores this ROI by progressively fine-tuning her/his preferences to finally converge to her/his MPS.

Artificial decision maker for interactive EMO

In this section, we describe the new ADM-II for comparing the performance of interactive EMO methods. As mentioned, ADM-II can handle various types of preference information such as providing a reference point, selecting either the most preferred or the most non-preferred solution(s) among a set of alternative solutions, specifying desirable objective function ranges, or performing pairwise comparisons. To compare the methods in a meaningful way, all of them are run simultaneously using the same computational resources (i.e., number of function evaluations or generations per iteration). If all of the interactive methods being compared use the same preference type, ADM-II produces the preference information in the same way for all methods. In case methods utilizing different types of preferences are compared, ADM-II generates the preferences accordingly and produces the type of preference information each method expects, but the philosophy underlying the generating procedure for the different types is similar.

Internally, our ADM-II follows a different strategy to generate the preferences at the iterations of each of the two phases of the interactive solution process. In the learning phase, it simulates an exploratory search in the objective space to inspect possible solutions. To this aim, the preference information for each iteration of this phase is generated in a way that the search is oriented toward the least explored region of the PF. At the last iteration of this phase, an ROI is found. In the decision phase, the behavior of ADM-II pursues a finer search within this ROI to find an MPS. Thus, at each iteration of this phase, the preferences produced by ADM-II are generated within the ROI to refine the solutions in it. The number of iterations carried out in each phase is set at the beginning. In what follows, we refer to the number of iterations in the learning and the decision phases by L and D, respectively. They are parameters of ADM-II.

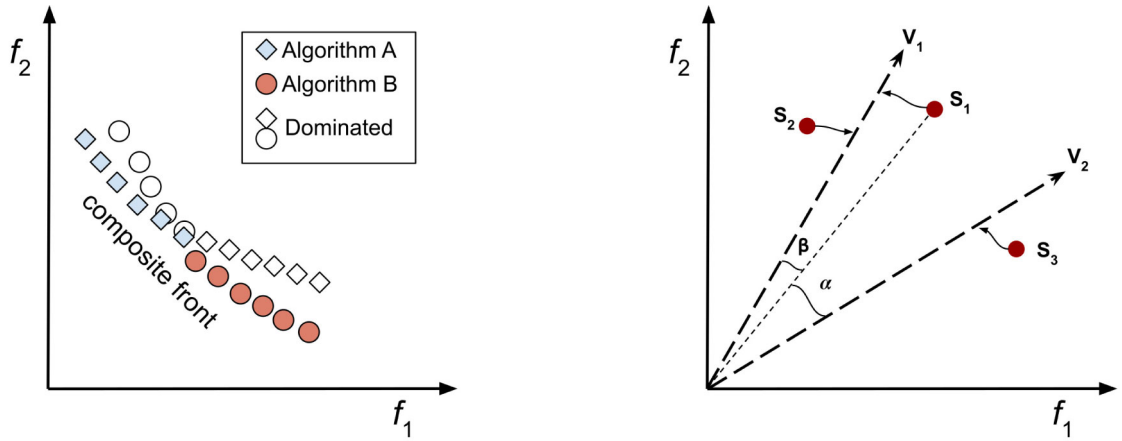
To generate new preferences depending on the responses of all the methods, ADM-II makes use of the solutions found so far by all methods included in the comparison. At each iteration, the solutions generated by the methods are combined, and a *composite front* is formed by deleting the dominated ones, as can be seen in Fig. 1a. To be more specific, at each iteration, the composite front is constituted by the nondominated solutions generated so far by all the methods. It is important to note that no information about the true PF is required to generate new preferences.

The generation of preferences in ADM-II is based on dividing the objective space into sub-areas. This is done following the philosophy of the so-called decomposition-based EMO methods (like [5]), although any other procedure allowing us to have information about sub-areas of the PF can also be used in ADM-II. EMO methods of this type usually decompose the original problem into several sub-problems. We use this idea to make a distinction between sub-areas of the PF that have already been explored more or less (i.e., the exploration degree of the different parts of the PF). Based on this, we decide how to generate the preferences at each iteration to guide the search for new nondominated solutions toward a specific part of the PF.

Let us describe the algorithm designed to divide the composite front into several sub-areas to identify the regions to be explored at each iteration. Initially, ADM-II creates a set of reference vectors uniformly distributed along the PF. To do this, the canonical simplex-lattice design method [7] is used, as suggested in [5,6]. In this method, the number of reference vectors that are created is controlled by a lattice resolution, which is given by $\binom{l+k-1}{k-1}$, where l is a pre-fixed parameter. Subsequently, at each iteration, ADM-II calculates the angles between each solution in the composite front and each reference vector. Then, each solution is assigned to the reference vector with the smallest angle. Figure 1b depicts an example for a bi-objective problem, where two reference vectors (V_1 and V_2) and three solutions (S_1, S_2 and S_3) are shown. Solution S_1 is assigned to V_1, given that the angle between S_1 and V_1 (denoted by β) is smaller than the angle between S_1 and V_2 (denoted by α). In the same way, S_2 is assigned to V_1 and S_3 is assigned to V_2.

Once all solutions in the composite front have been assigned to the reference vectors, ADM-II gets information about the exploration degree of each region of the PF at the current iteration by counting the number of assigned solutions to each reference vector. The more solutions assigned to a reference vector, the best explored the sub-area is, where the reference vector lies. This information is conveniently used in ADM-II to generate new preferences at each iteration of the learning and decision phases depending on the needs, as described in "Preference generation in the learning and decision phases".

ADM-II also considers how the interactive EMO methods compared reflect the preference information in the solutions they produce. As described in "Performance evaluation", it employs performance indicators that internally consider the preferences used along the iterations. Furthermore, cumulative indicator values are also computed separately in the learning and decision phases, since they allow us to evaluate the performances of the methods in both phases. In addition, we propose a new performance indicator, called *contribution*



(a) Composite front with nondominated solutions

(b) Reference vectors and assignment of solutions

Fig. 1 Division of the PF internally performed in ADM-II

to CF, which is obtained as the number of nondominated solutions each method has contributed to build/extend the composite front along the iterations.

Algorithm 1 contains the main steps of ADM-II.

Algorithm 1 Main steps of ADM-II

Step 0: Initialize all methods and provide the first preferences randomly.

Step 1: Run all methods with the same computational budget (number of generations or function evaluations) and the previously generated preferences.

Step 2: Build or update the composite front using the solutions obtained by each method until this iteration.

Step 3: Evaluate the methods' performances taking into account the preferences used.

Step 4: Generate new preferences for the next iteration based on the composite front and phase (learning or decision) of the iteration being performed, according to the strategy designed for each preference type:

- **a)** In the learning phase, generate the new preferences for the least explored area of the composite front. At the end of the learning phase, identify the best explored area as the ROI.
- **b)** In the decision phase, generate the new preferences within the ROI identified at the end of the learning phase.

Step 5: If a termination criterion is met, terminate the process and calculate cumulative indicator values for each phase. Else, continue with Step 1.

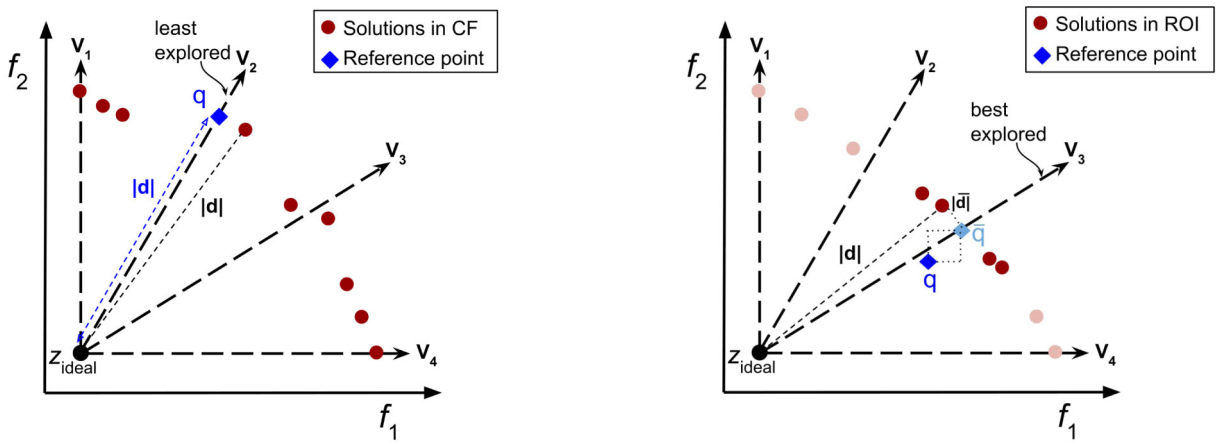
Preference generation in the learning and decision phases

As previously mentioned, in the learning phase, the main purpose of ADM-II is to explore the whole set of Pareto optimal solutions. Therefore, it progressively inspects the sub-areas of the PF that have been poorly covered so far. At each iteration of this phase, a set of uniformly distributed reference vectors on the composite front is first obtained, and the solutions of the composite front are assigned to reference vectors, as described before. Then, the least explored area of the composite front is determined based on the reference vector that has the lowest number of assigned solutions. ADM-II generates preferences for the next iteration to direct the search for new nondominated solutions toward this region. In the example of Fig. 2a, V_2 would be selected as the vector defining the least explored area. It should be noted that ADM-II does not consider the vectors with no solutions assigned. This prevents algorithms from being compelled to search for solutions in areas where no solutions exist in the true PF. Once the iterations corresponding to the learning phase have been completed (until iteration L), ADM-II finds the reference vector with the highest number of assigned solutions. Let us refer to this vector as V_D. The part of the PF where the vector V_D is located can be assumed to be the best explored area of the composite front. Then, the ROI to be explored in detail at the next D iterations (corresponding to the decision phase) is formed by the solutions assigned to V_D, and the preference information is obtained at each iteration based on this vector.

As one can note, the way of generating the preference information in both phases depends on the reference vector selected. In what follows, we describe the strategy designed to generate different types of preference information for the two phases.

Giving a reference point

In the learning phase, the reference vector selected at each iteration (identifying the least explored area) and the solu-



(a) Learning phase

(b) Decision phase

Fig. 2 Reference point as preference information in ADM-II

tions assigned to this vector are used to find the location of the new reference point. For this, the distances of these solutions to the ideal point of the current composite front are calculated, and the one with the minimum distance (denoted by $|d|$) is identified. The next reference point is then located on the selected reference vector according to this distance $|d|$. Figure 2a graphically shows how the new reference point is generated, which is denoted by $\mathbf{q}$.

At the iterations of the decision phase, reference points are generated to progressively converge toward the PF by performing a finer search in the ROI identified at the end of the learning phase to find an MPS. ADM-II finds the solution assigned to the reference vector V_D with the minimum distance $|d|$ to the ideal point of the composite front. Then, $|d|$ allows us to generate the next reference point, as shown in Fig. 2b. First, we obtain the point labeled as $\bar{\mathbf{q}}$ along the reference vector V_D using $|d|$, and then, the new reference point $\mathbf{q}$ is generated by applying a perturbation to each component of this point. For the perturbation, the distance $|\bar{d}|$ between $\bar{\mathbf{q}}$ and the nearest solution to $\bar{\mathbf{q}}$ from the composite front is calculated, and then, each component of $\mathbf{q}$ is obtained as the corresponding component of $\bar{\mathbf{q}}$ minus $|\bar{d}|$. This way of producing reference points in the decision phase assures that the new reference point always dominates the previously generated points. In practice, this means that the reference points generated in this phase get progressively closer to the ideal point as the iterations are performed, since the solutions provided by the methods, which are used to update the composite front, progressively converge to the PF.

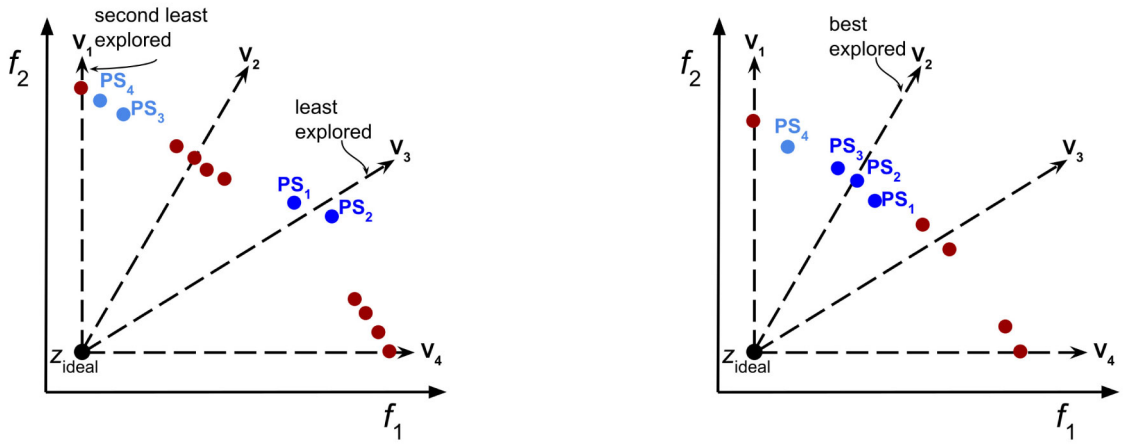
It is noteworthy that the way of generating reference points in the learning phase is similar to the procedure designed in the previous ADM [1]. Nevertheless, in the decision phase, the behavior of ADM-II for producing new reference points is totally different from [1] (where the reference points generated in this phase always lie on the reference vector V_D of the ROI being explored).

Selecting the preferred solution(s)

At each iteration of the learning phase, the most preferred solution(s) are selected among the solutions assigned to the reference vector of the least explored area. First, the distance of each solution to the ideal point of the current composite front is calculated. Then, all the assigned solutions are ranked in descending order according to their distances to the ideal point, and the first p solutions are selected as the most preferred solutions $\mathbf{PS}_1, \ldots, \mathbf{PS}_p$. It may happen that ADM-II has to find a higher number of preferred solutions than the number of assigned solutions to the selected reference vector. In this case, the reference vector of the second least explored area is found, and ADM-II selects the necessary number of preferred solutions from the solutions assigned to this vector, in a similar way, based on their distances to the ideal point.

In the decision phase, ADM-II selects the most preferred solutions in a similar way, considering the solutions assigned to the reference vector V_D (which represents the best explored area at the end of the learning phase). If the number of assigned solutions to V_D is lower than the required number of preferred solutions, the remaining preferred solutions are the closest ones to V_D based on the angle values, even if these solutions are assigned to other reference vectors. Since the purpose of ADM-II is to refine the search for solutions in the ROI defined by V_D, we provide the preference information within this ROI and near to it if needed.

This preference generation in the learning phase is exemplified in Fig. 3a, where V_3 is the vector representing the least



(a) Learning phase (b) Decision phase

Fig. 3 Selecting the p most preferred solutions as preference information in ADM-II

explored area (since it has the least number of assigned solutions). If only one preferred solution is required by a method, $\mathbf{PS}_1$ is the one selected given that it is the closest one to the ideal point. If a method expects two preferred solutions, then $\mathbf{PS}_1$ and $\mathbf{PS}_2$ are selected. If, e.g., four preferred solutions have to be selected, as V_3 has only two assigned solutions, the reference vector V_1 is found as the second least explored area. Then, the remaining preferred solutions are chosen among the ones assigned to V_1 based on their distances to the ideal point. In this case, besides $\mathbf{PS}_1$ and $\mathbf{PS}_2$, solutions $\mathbf{PS}_3$ and $\mathbf{PS}_4$ are chosen as preferred solutions.

The behavior in the decision phase is shown in Fig. 3b. If a method expects, e.g., four preferred solutions, ADM-II selects the three solutions assigned to the reference vector of the best explored area (V_2 in this case), and the solution $\mathbf{PS}_4$ (even though it is assigned to V_1), because it is the one with the smallest angle to V_2.

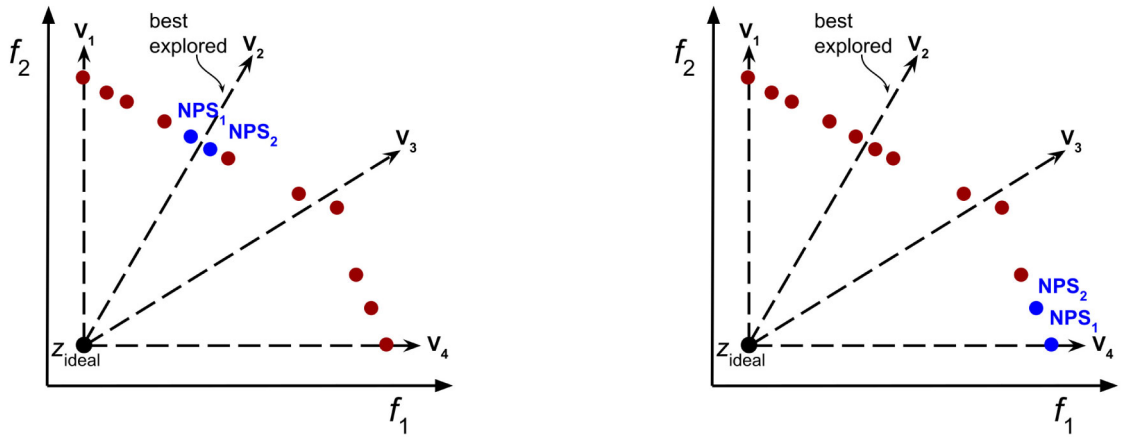
Selecting the non-preferred solution(s)

For finding non-preferred solutions, we follow a similar procedure to the previous one, but select the most unwanted solutions, so that the methods would not converge to the regions where unwanted solutions are. At each iteration of the learning phase, ADM-II finds the reference vector of the best explored area (with the highest number of assigned solutions). Then, the non-preferred solutions are found among the solutions assigned to this vector. In this way, in the learning phase, ADM-II avoids producing more nondominated solutions in the best approximated area, so that the regions with fewer solutions are emphasized. On the other hand, at each iteration in the decision phase, ADM-II selects as the non-preferred ones the solutions with the largest angles to the reference vector V_D representing the ROI that is being explored. This means that ADM-II avoids selecting the solutions outside the ROI or the furthest ones from V_D.

Figure 4a represents a case where a method expects the two most non-preferred solutions in the learning phase. As shown, $\mathbf{NPS}_1$ and $\mathbf{NPS}_2$ are selected among the solutions assigned to the vector V_2 associated with the best explored area. With this, ADM-II avoids getting more solutions around V_2 by selecting $\mathbf{NPS}_1$ and $\mathbf{NPS}_2$ as non-preferred solutions, because the region where V_2 is has already been explored. In this way, ADM-II seeks to search for solutions from the least explored areas, which is the purpose of the learning phase. In Fig. 4b, the solutions selected in the decision phase are shown. In this case, V_2 is the vector that represents the ROI. $\mathbf{NPS}_1$ and $\mathbf{NPS}_2$ are the furthest solutions from V_2, based on the angles to this vector, and ADM-II chooses them as the two most non-preferred solutions.

Specifying preferred ranges

To generate preferred ranges for objective functions, first, ADM-II generates a reference point $\mathbf{q}$ as indicated in "Giving a reference point", depending on the phase in question. Then, the ranges are calculated at each iteration by perturbing the components of $\mathbf{q}$ using the distance $|\bar{d}|$ of the nearest solution of the composite front to $\mathbf{q}$. That is, for every $i = 1, \ldots, k$, the desirable range for the objective function f_i is defined as $[q_i - |\bar{d}|, q_i + |\bar{d}|]$, where q_i is the component i of $\mathbf{q}$. This is illustrated in Fig. 5.



(a) Learning phase (b) Decision phase

Fig. 4 Selecting the *np* most non-preferred solutions as preference information in ADM-II

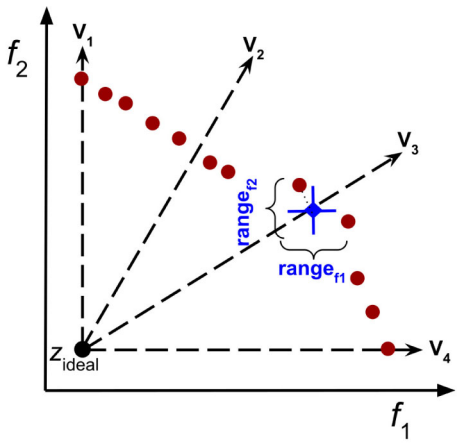


Fig. 5 Specifying desirable objective function ranges as preference information in ADM-II

Performing pairwise comparisons

For pairwise comparisons, ADM-II compares the solutions generated by each method in the following way. At each iteration in the learning phase, the solution that is finally chosen is the one with the minimum angle to the reference vector of the least explored area. On the other hand, at each iteration in the decision phase, from the two ones provided by the method, the solution with the minimum distance to the ideal point of the composite front is chosen.

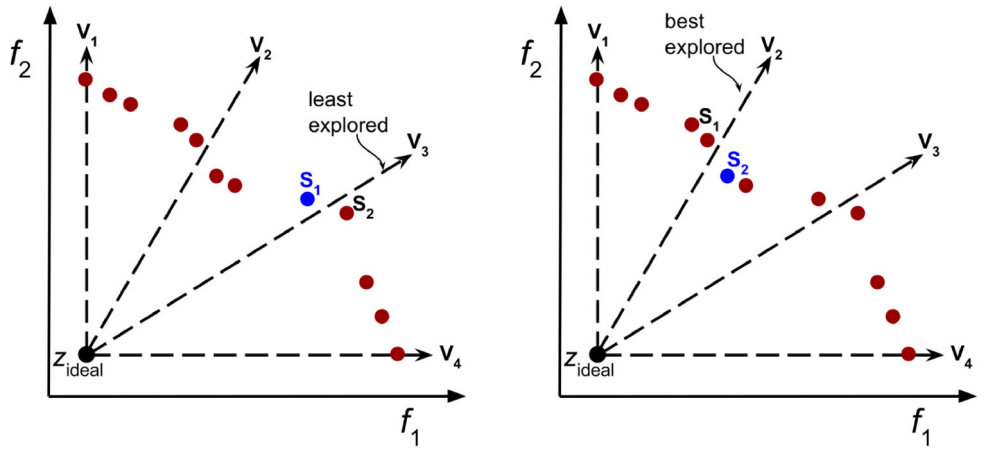
Figure 6a illustrates the comparison of solutions in the learning phase, where solution $\mathbf{S}_1$ is selected rather than $\mathbf{S}_2$, since it has the smallest angle to V_3 (which represents the least explored area). In Fig. 6b, we show the behavior in the decision phase. Between $\mathbf{S}_1$ and $\mathbf{S}_2$, ADM-II selects $\mathbf{S}_2$, since it is closer to the ideal point.

Performance evaluation

To evaluate the performance of methods aimed at generating solutions reflecting preferences, it is not enough to quantify the convergence (closeness to the PF) and diversity (spread over the PF and uniformity among solutions) among the nondominated solutions obtained. Basically, this is the information that is assessed by commonly used performance indicators of (a posteriori) EMO methods [15]. However, when preferences are considered, the performance should also be assessed regarding the ROI defined by the preferences.

Moreover, ideally, aspects in relation to the interaction with the DM should also be considered to evaluate the performance of interactive methods. Indeed, besides evaluating how well each method obeys the preferences (i.e., the method's ability to generate solutions reflecting the different preferences), the quality of the solutions generated should be measured differently in the two phases of the solution process, since they have different goals. In the learning phase, one should measure how well each method responses to the given preference information at different parts of the PF; and in the decision phase, how well the method converges when exploring solutions within a specific ROI.

In the literature, we can find performance indicators for EMO methods that incorporate preferences given a priori (before the solution process starts) [12,14,23,30]. However, to the best of our knowledge, quality indicators for measuring the quality of the solutions found by interactive methods



(a) Learning phase (b) Decision phase

Fig. 6 Performing pairwise comparisons as preference information in ADM-II

are not available in the literature, and we propose a performance indicator for this purpose here. This indicator is named as *contribution to CF*, and it is aimed at evaluating the contribution of each method being compared to build the composite front. That is, our indicator quantifies the number of nondominated solutions which each method has provided for building the composite front. The more nondominated solutions a method has contributed to forming the composite front, the better the performance of this method is.

Note that the composite front is updated at each iteration, and it is obtained by merging the nondominated solutions produced by all the methods. Thus, this indicator offers an insight into the exploratory potential of each method and its capacity to adapt the search for new nondominated solutions toward the area of the PF corresponding to the preferences used (i.e., the responsiveness of each method for the generated preference information). Specifically, since ADM-II generates preference information to explore different regions of the PF during the learning phase, at the iterations of this phase, the number of solutions provided by each method to the composite front reflects its exploration potential. If one method has contributed with a higher number of solutions than other one(s), we could say that it has a better exploration capability than the other(s). However, because ADM-II focuses on finding better solutions in a specific ROI during the decision phase, the method that provides more solutions to the composite front in this phase has a stronger convergence capability than the other(s). Furthermore, in the decision phase, an achievement scalarizing function, such as the one in [28], can be used to measure the quality of the final solution(s) found by each method. This function projects the final solution to the PF, and one can study the distance to evaluate its closeness to the PF.

Additionally, ADM-II assesses the performance of all the methods based on commonly used performance indicators as follows. Let us denote by m_i the value of a performance indicator m at iteration i for a method. For each method, ADM-II calculates the value of m_i for the solution set obtained by each method at each iteration i. When the solution process has finished, cumulative indicator values are calculated for all the methods to evaluate their performance for the learning and decision phases, separately. The performance in the learning phase is evaluated as $\sum_{i=1}^{L} m_i$, using the indicator values until iteration L, while the one in the decision phase is calculated as $\sum_{i=L+1}^{L+D} m_i$, with the indicator values from iteration $L+1$ until the end.

Because of the way of generating preferences in the proposed ADM-II, indicators developed for a priori reference point-based EMO methods can be employed to measure the performance of interactive methods based on some of the preference types considered, like methods using preferred ranges, because ADM-II internally produces a reference point to generate this type of preferences (see "Specifying preferred ranges"). We use the R-metric [14] that applies regular performance indicators to measure the quality of a set of solutions to approximate the ROI associated with a reference point. Internally, a parameter Δ is employed to control the size of the considered ROI. To be more specific, we use the R-IGD metric (i.e., R-metric using IGD), because it allows quantifying both convergence and diversity of the solutions and is computationally efficient for problems with a high number of objectives. The lower the R-IGD value, the better the quality of the solutions is to approximate the ROI.

Computational experiments

When it comes to choosing the most suited interactive method for a specific problem, assessing and comparing the performance of interactive methods are crucial. For this purpose,

different experiments can be designed using our ADM-II based on different needs. Here, we demonstrate its applicability and usefulness to assess the performance of several interactive EMO methods from different perspectives.

We have conducted two separate experiments. In the first one, we use ADM-II to compare the performance of interactive reference point-based versions of the EMO methods RVEA [5] and NSGA-III [8]. The main purpose of this experiment is to demonstrate the contributions made in comparison to the previous ADM [1] in relation to the quality indicators. Interactive RVEA (named as iRVEA) was proposed in [11], and interactive NSGA-III (named as iNSGAIII) has been implemented in a corresponding way. In the second experiment, we compare iRVEA using preferred ranges and reference points as preference information to illustrate ADM-II's ability to handle different types of preferences. The Python implementations of the aforementioned methods and ADM-II are openly available under the DESDEO framework [22] (https://desdeo.it.jyu.fi).

Experimental settings

In what follows, we describe the experimental settings for the two experiments. We solved the DTLZ1-4 benchmark problems [10] with 3–9 objectives (k), which implied a total of 28 different problems. As recommended in [10], for each problem, the number of decision variables was set as $10 + k - 1$. We provided the same computational budget to all the methods: 100 generations per iteration. Besides, we used the parameter values for iRVEA and iNSGAIII proposed in [5] and [8], respectively.

We used the same parameter values in ADM-II as we considered in [1] for the ADM proposed there. That is, the number of iterations for the learning phase was $L = 4$, the number of iterations for the decision phase $D = 3$, and the lattice resolution 5. However, note that ADM-II can be easily adapted to test different capabilities of interactive methods according to the needs. If the method's responsiveness for learning purposes is to be assessed, the number of iterations in the learning phase can be set to a high value. On the other hand, if the purpose is to evaluate the convergence ability of interactive methods, one can increase the number of iterations in the decision phase. Furthermore, it is also possible to apply ADM-II to test the methods in only one phase by setting the number of iterations for the other one as zero. In addition, ADM-II can be configured to run multiple phases in different orders. We have conducted our experiments here, so that the solution process begins with the learning phase and continues with the decision phase, which is the typical order followed in most real-world applications.

Last but not least, to apply the R-metric, we have to set a value for the parameter Δ. We set $\Delta = 0.3$ for the learning phase and $\Delta = 0.2$ for the decision phase, as suggested in [1]. In the R-metric calculation, this parameter is utilized to specify the ROI based on the reference point provided by the DM and significantly affects the results. Therefore, we chose a greater Δ for the learning phase, because the ADM-II is still exploring, and a slightly lower value for the decision phase, because the ROI has already been found. However, there is no widely accepted way to set Δ in the literature, and further investigation and sensitivity analysis on this parameter are required.

To ensure the reliability of the numerical results presented, we performed 21 independent runs with ADM-II. We used the same experimental settings for each experiment and each problem.

Numerical results

As mentioned earlier, in addition to the R-IGD metric, we evaluate the methods based on the new quality indicator *contribution to CF* proposed in "Performance evaluation". This indicator helps understand the method's responsiveness to the provided preference information, which is generated in ADM-II differently in both phases.

Even though we specified the same budget of 100 generations per iteration to all methods in the comparison, the actual number of function evaluations varied among them due to the way they internally handle populations of solutions. Therefore, in addition to R-IGD and the proposed indicator, we also show the number of function evaluations (denoted by FEs) required by each method to comprehend the actual budget utilized at every iteration. This information is crucial, since it tells us more about the methods' performance when we combine it with the R-IGD results and the contribution to CF. For example, if a method has a lower R-IGD value and contributes with more solutions to build the composite front using fewer function evaluations, this information indicates that the method is preferable for that case.

As previously stated, we examine cumulative indicator values for each phase, which means that we sum the results of each iteration dedicated to the learning and decision phases separately. Eventually, we list the mean and standard deviations of the aforementioned indicator values of 21 independent runs at the end of each phase.

iRVEA vs. iNSGAIII

In the first experiment, we compared iRVEA with iNSGAIII, both using reference points as preference information. Table 1 shows the mean and standard deviations of the results of iRVEA and iNSGAIII for both phases (iterations 1–4 for the learning phase, and iterations 5–7 for the decision phase) on the DTLZ problems. To save space, we just show in Table 1 the results for 4, 7, and 9 objectives. The rest of the results

Table 1 Cumulative results of the comparison of iRVEA and iNSGAIII

Problem	k	Phase	R-IGD				Contribution to CF				FEs			
			iRVEA		iNSGAIII		iRVEA		iNSGAIII		iRVEA		iNSGAIII	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
DTLZ1	4	Learning	2.426	0.153	**2.418**	0.132	94	99	**442**	74	123,402	21,742	192,000	0
		Decision	2.071	0.359	**2.059**	0.349	273	79	**358**	4	136,582	6812	144,000	0
	7	Learning	**2.650**	0.204	2.682	0.330	**296**	21	247	31	171,102	8327	235,200	0
		Decision	**2.005**	0.200	2.093	0.200	**266**	3	141	74	172,995	5029	176,400	0
	9	Learning	**2.728**	0.201	2.750	0.179	**615**	38	514	80	425,365	35,933	597,600	0
		Decision	**1.907**	0.155	2.005	0.230	**518**	2	175	132	441,361	12,085	448,200	0
DTLZ2	4	Learning	0.418	0.289	**0.199**	0.167	285	35	**478**	4	119,035	16,010	192,000	0
		Decision	0.554	0.540	**0.247**	0.273	188	114	**346**	6	85,133	56,111	144,000	0
	7	Learning	1.198	0.532	**0.602**	0.116	246	75	**334**	4	132,682	30,634	235,200	0
		Decision	1.210	0.200	**0.543**	0.454	134	116	**245**	3	92,069	77,792	176,400	0
	9	Learning	0.657	0.419	**0.536**	0.233	579	115	**650**	12	379,611	84,929	597,600	0
		Decision	2.061	1.908	**0.949**	0.419	371	220	**490**	6	310,989	179,610	448,200	0
DTLZ3	4	Learning	0.178	0.051	**0.173**	0.055	44	61	**470**	36	144,331	8773	192,000	0
		Decision	0.185	0.415	**0.079**	0.041	268	86	**358**	3	126,203	29,540	144,000	0
	7	Learning	**0.450**	0.214	0.457	0.165	**269**	26	217	31	195,078	11,585	235,200	0
		Decision	**0.222**	0.091	0.324	0.109	**259**	12	78	34	173,449	12,571	176,400	0
	9	Learning	**0.610**	0.307	0.857	0.464	475	69	**495**	46	411,815	76,073	597,600	0
		Decision	**0.327**	0.136	0.960	0.807	**432**	111	225	63	343,458	94,103	448,200	0
DTLZ4	4	Learning	0.255	0.128	**0.171**	0.179	282	11	**482**	1	169,013	6052	192,000	0
		Decision	0.636	0.437	**0.617**	0.430	336	8	**356**	2	142,216	2098	144,000	0
	7	Learning	1.267	0.889	**0.701**	0.151	297	63	**330**	13	172,607	38,083	235,200	0
		Decision	0.747	0.350	**0.645**	0.354	244	38	**248**	5	168,479	30,595	176,400	0
	9	Learning	0.800	0.254	**0.708**	0.262	**685**	8	663	1	537,059	29,818	597,600	0
		Decision	1.118	0.325	**1.093**	0.358	**518**	1	496	2	438,909	14,570	448,200	0

are available in Table 3 of the Appendix. We highlight the best results in bold font in both tables.

Since we provided a fixed number of generations to both methods, the number of function evaluations is proportional to the population size. For iNSGAIII, there is no standard deviation in the number of function evaluations, since its internal procedure always assures the same population sizes, and its total number of function evaluations varies depending on the number of objectives of the problems. Because iRVEA employs adaptive reference vector management, the population size varies throughout the solution process, resulting in a variation of the number of function evaluations.

In terms of DTLZ1 results, iRVEA performed better when the number of objectives was high, and iNSGAIII had better R-IGD values and contributed more to building the composite front with four objectives. However, it consumed more function evaluations. On the other hand, iRVEA outperformed iNSGAIII in both R-IGD values and the number of solutions contributing to the composite front for seven and nine objectives. To attain this, iRVEA consumed fewer function evaluations making its performance significantly better than iNSGAIII. The same conclusions can be made for the DTLZ3 problem.

The R-IGD values reached by iNSGAIII were better than those of iRVEA for every DTLZ2 and DTLZ4 problem, but it required a higher number of function evaluations. An interesting observation can be made for the nine-objective DTLZ4 problem. Despite the fact that iNSGAIII had higher R-IGD values for both phases, iRVEA contributed more to the composite front with fewer function evaluations.

These findings show that the quality indicators employed serve a variety of purposes and complement one another. In general, with iNSGAIII, a larger function evaluation budget is required to ensure the quality of the solutions. On the other hand, it appears that iRVEA is more efficient in terms of budget utilization vs. contribution to the composite front.

Table 2 Cumulative results of the comparison of reference point- and preferred range-based iRVEA

Problem	k	Phase	R-IGD				Contribution to CF				FEs			
			iRVEA-RP		iRVEA-Ranges		iRVEA-RP		iRVEA-Ranges		iRVEA-RP		iRVEA-Ranges	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
DTLZ1	4	Learning	**2.762**	0.096	2.896	0.152	**210**	114	177	108	157,746	6878	159,461	24,309
		Decision	1.787	0.095	**1.775**	0.104	**349**	22	275	126	134,653	6757	167,615	3311
	7	Learning	**2.916**	0.225	3.505	0.336	211	107	**312**	134	216,973	5618	256,392	21,765
		Decision	2.035	0.375	**2.025**	0.336	**200**	57	185	142	180,099	5427	187,227	108,020
	9	Learning	**2.835**	0.150	3.637	0.333	**577**	85	570	238	524,079	31,435	594,523	46,099
		Decision	1.983	0.407	**1.980**	0.367	**514**	3	449	253	452,580	10,169	460,364	241,346
DTLZ2	4	Learning	**0.564**	0.421	0.710	0.476	**257**	99	208	97	100,852	34,106	102,171	43,516
		Decision	0.997	0.792	**0.879**	0.838	**131**	135	59	59	62,446	61,676	59,605	69,507
	7	Learning	**0.486**	0.216	0.940	0.343	248	62	**276**	87	136,385	37,755	184,031	45,542
		Decision	1.155	1.565	**0.897**	1.369	**177**	109	166	99	122,086	73,071	177,301	93,222
	9	Learning	**0.528**	0.274	0.908	0.213	**510**	142	423	221	341,056	112,124	389,291	180,654
		Decision	2.909	2.381	**1.562**	1.444	222	215	**397**	231	172,533	162,562	358,663	204,173
DTLZ3	4	Learning	**0.234**	0.147	0.651	0.338	**174**	44	44	44	148,334	23,573	110,230	56,902
		Decision	**0.427**	0.643	0.570	0.611	**249**	134	25	38	107,189	47,229	71,661	70,361
	7	Learning	**0.525**	0.416	1.461	0.447	190	62	**233**	137	182,259	31,951	225,480	41,383
		Decision	0.575	0.969	**0.379**	0.293	**185**	74	173	117	158,003	37,831	205,151	49,427
	9	Learning	**0.369**	0.093	1.548	0.587	**503**	83	374	182	470,856	52,312	466,521	80068
		Decision	0.658	0.544	**0.330**	0.248	**323**	217	296	150	275,755	184,561	471,451	173,463
DTLZ4	4	Learning	**0.182**	0.163	0.358	0.111	**380**	52	263	169	169,891	7496	116,161	56,912
		Decision	**0.268**	0.342	0.313	0.364	**318**	13	127	40	141,123	3540	139,299	15,072
	7	Learning	**2.400**	2.691	2.989	2.526	212	128	**247**	141	133,483	82,945	158,791	95,112
		Decision	**1.469**	1.458	1.875	2.312	168	94	**241**	117	119,533	65,213	199,386	92,390
	9	Learning	**0.931**	0.323	1.618	0.467	684	6	**724**	148	570,285	11,818	642,717	70788
		Decision	0.743	0.346	**0.539**	0.509	**515**	3	462	155	446,762	13,336	510,922	59,213

iRVEA with reference points vs. preferred ranges

In the second experiment, we compared reference point-based iRVEA (denoted by iRVEA-RP) and iRVEA using preferred ranges (denoted by iRVEA-Ranges). The purpose is to demonstrate the ability of ADM-II to handle different types of preferences in the comparative study. The results are summarized in Table 2. As in the previous experiment, we report the results for 4, 7, and 9 objectives, while the remaining results are available in Table 4 of the Appendix. In both tables, we highlight the best results in bold font.

In relation to the R-IGD metric, we can see that iRVEA-Ranges outperformed iRVEA-RP in the decision phase of almost all the problems, while iRVEA-RP had better R-IGD values in the learning phase of most of them. This means that, on one hand, iRVEA-Ranges exploited and refined the solutions better in the ROI than iRVEA-RP according to our experiment with ADM-II. Besides, on the other hand, iRVEA responded better to the drastic changes in the provided reference points than those in the provided preferred ranges.

Moreover, iRVEA-RP contributed more to the composite front than iRVEA-Ranges using slightly fewer function evaluations in almost all cases, with a few exceptions. However, when iRVEA-Ranges contributed more solutions to the composite front, it consumed more function evaluations than iRVEA-RP. Therefore, a higher allocation of resources is required to get better-qualified solutions using iRVEA-Ranges.

With these experiments, we have demonstrated that the preference generation in ADM-II and the quality indicators introduced support comparing different perspectives of interactive methods. Before applying interactive methods in real-world applications, one can utilize our ADM to find particular methods for various needs. For example, one can apply iRVEA-RP to support the DM in exploring different regions of the objective space, gaining insight into the problem, and understanding the feasibility of preferences. In this experiment, we have seen that iRVEA-Ranges is more suitable to find better solutions in a specific ROI in the decision phase. However, this was the case for DTLZ problems and

the findings reached may differ in other problems, meaning that some specific methods may be appropriate for specific problems with different needs. It should be highlighted that our intention is not to generalize our findings of methods compared but rather demonstrate how to apply ADM-II in various situations. Besides, our ADM does not alter the internal functioning of the interactive methods that are compared, it just compares them by providing preference information in a different manner for the learning and decision phases, separately.

Conclusions

In this paper, we have proposed a novel artificial decision maker, ADM-II, to compare interactive EMO methods without involving human DMs. As preference information, ADM-II can generate reference points, preferred and nonpreferred solutions among a set of solutions, preferred ranges for the objective functions, and pairwise comparisons. Thus, it can be used to compare methods utilizing these types of preferences. It generates preferences using a similar philosophy to allow a fair comparison of interactive methods handling different types of preference information. Besides, ADM-II produces the preferences in different ways in the learning and decision phases of interactive methods to reflect their different goals in either exploring various solutions or converging in the region of interest, respectively.

In addition, we have proposed a new performance indicator. It counts how many nondominated solutions each method contributes in building a composite front formed by nondominated solutions of all methods compared. This indicator enables the evaluation of different capabilities of interactive methods. Since the composite front is built and updated after each iteration, this indicator informs about each method's responsiveness to the provided preference information.

Our two experiments have demonstrated the applicability of ADM-II for different purposes. The benefits of the new indicator, along with the R-IGD metric and the number of function evaluations, were shown in the first experiment by comparing two interactive methods using the same type of preference information. Furthermore, a second experiment was conducted to show the potential of ADM-II in generating different types of preferences. Based on the results of these experiments, we can say that ADM-II helps to understand the methods' ability to respond to the preference information provided in the learning and decision phases. Before involving a real DM in the solution process of real-world problems, ADM-II can be applied with various candidate interactive methods, even handling different types of preference information, and the most suited method(s) can be found. It is also possible to select some methods based on a particular preference type to gain insight into the problem in the learning phase and then apply another method to find an MPS in the decision phase.

We used fixed values for the number of iterations dedicated to the learning and decision phases. In our future research, we plan to make ADM-II adaptive, so that it automatically decides when to switch from the learning phase to the decision phase and when to end the decision phase. Another important future research topic is to develop new quality indicators for measuring desirable properties of interactive methods.

Acknowledgements The authors would like to express their gratitude to Professor Francisco Ruiz for his helpful suggestions and comments on this research. The authors would like to thank the financial support received from the Spanish government (Grant ECO2017-88883-R), the regional government of Andalusia (Grant UMA18-FEDERJA-024 and PAI group SEJ-532), and the Academy of Finland (Grants 322221 and 311877). This work is related to the thematic research area Decision Analytic utilizing Causal Models and Multiobjective Optimization (DEMO), https://jyu.fi/demo, at the University of Jyvaskyla.

Declaration

Conflicts of interest/competing interests The authors declared that they have no conflicts of interest regarding the publication of this manuscript.

Code availability The implementation of ADM-II in Python is openly available under the DESDEO framework at https://desdeo.it.jyu.fi

Appendix

We utilized the DTLZ1-4 problems with a number of objectives ranging from 3 to 9 in the computational experiments. The results of the comparison of the reference point-based iRVEA and iNSGA-III methods and the comparison of the reference point- and preferred range-based iRVEA methods can be found in Tables 3 and 4, respectively. In both tables, we highlight the best results in bold font.

Table 3 iRVEA vs. iNSGAIII: numerical results with 100 generations per iteration for the objectives ranging from 3 to 9

Problem	k	Phase	R-IGD				Contribution to CF				FEs			
			iRVEA		iNSGAIII		iRVEA		iNSGAIII		iRVEA		iNSGAIII	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
DTLZ1	3	Learning	2.502	0.091	**2.470**	0.119	5	10	**418**	5	77,863	5871	127,200	0
		Decision	1.797	0.147	**1.792**	0.146	180	49	**311**	17	91,137	2744	95,400	0
	4	Learning	2.426	0.153	**2.418**	0.132	94	99	**442**	74	123,402	21,742	192,000	0
		Decision	2.071	0.359	**2.059**	0.349	273	79	**358**	4	136,582	6812	144,000	0
	5	Learning	2.446	0.172	**2.396**	0.109	269	100	**347**	147	137,139	45,616	252,000	0
		Decision	1.876	0.108	**1.846**	0.186	333	58	**356**	27	159,539	29,086	189,000	0
	6	Learning	**2.516**	0.133	2.795	0.326	**420**	99	253	96	218,453	14,184	302,400	0
		Decision	**1.875**	0.086	1.964	0.156	**386**	24	280	107	220,743	12,253	226,800	0
	7	Learning	**2.650**	0.204	2.682	0.330	**296**	21	247	31	171,102	8327	235,200	0
		Decision	**2.005**	0.200	2.093	0.200	**266**	3	141	74	172,995	5029	176,400	0
	8	Learning	**2.724**	0.114	3.014	0.230	**427**	31	369	48	268,776	19,937	384,000	0
		Decision	**2.211**	0.263	2.770	0.432	**380**	1	319	33	286,155	6493	288,000	0
	9	Learning	**2.728**	0.201	2.750	0.179	**615**	38	514	80	425,365	35,933	597,600	0
		Decision	**1.907**	0.155	2.005	0.230	**518**	2	175	132	441,361	12,085	448,200	0
DTLZ2	3	Learning	0.173	0.119	**0.132**	0.056	147	25	**422**	1	89,746	6321	127,200	0
		Decision	0.271	0.492	**0.067**	0.035	176	65	**312**	4	68,667	24,823	95,400	0
	4	Learning	0.418	0.289	**0.199**	0.167	285	35	**478**	4	119,035	16,010	192,000	0
		Decision	0.554	0.540	**0.247**	0.273	188	114	**346**	6	85,133	56,111	144,000	0
	5	Learning	0.949	0.487	**0.499**	0.398	280	91	**501**	3	126,647	30,617	252,000	0
		Decision	0.766	0.751	**0.318**	0.332	159	141	**364**	5	74,435	62,600	189,000	0
	6	Learning	0.512	0.324	**0.337**	0.048	443	80	**498**	4	202,037	45,378	302,400	0
		Decision	0.412	0.269	**0.386**	0.251	345	50	**369**	6	191,934	39,591	226,800	0
	7	Learning	1.198	0.532	**0.602**	0.116	246	75	**334**	4	132,682	30,634	235,200	0
		Decision	1.210	0.200	**0.543**	0.454	134	116	**245**	3	92,069	77,792	176,400	0
	8	Learning	0.680	0.306	**0.517**	0.076	409	117	**469**	5	247,429	70,769	384,000	0
		Decision	**0.458**	0.162	0.483	0.243	288	78	**349**	7	195,865	68,138	288,000	0
	9	Learning	0.657	0.419	**0.536**	0.233	579	115	**650**	12	379,611	84,929	597,600	0
		Decision	2.061	1.908	**0.949**	0.419	371	220	**490**	6	310,989	179,610	448,200	0
DTLZ3	3	Learning	0.280	0.241	**0.135**	0.062	21	25	**391**	33	86,658	17,667	127,200	0
		Decision	0.116	0.070	**0.128**	0.073	149	111	**309**	10	83,371	5569	95,400	0
	4	Learning	0.178	0.051	**0.173**	0.055	44	61	**470**	36	144,331	8773	192,000	0
		Decision	0.185	0.415	**0.079**	0.041	268	86	**358**	3	126,203	29,540	144,000	0
	5	Learning	**0.248**	0.183	0.353	0.135	121	65	**476**	52	199,067	30,650	252,000	0
		Decision	0.109	0.016	**0.081**	0.070	**347**	72	328	94	186,264	7447	189,000	0
	6	Learning	**0.372**	0.046	0.473	0.193	**346**	70	333	83	231,555	8571	302,400	0
		Decision	0.238	0.074	**0.225**	0.128	**385**	4	294	99	221,365	9023	226,800	0
	7	Learning	**0.450**	0.214	0.457	0.165	**269**	26	217	31	195,078	11,585	235,200	0
		Decision	**0.222**	0.091	0.324	0.109	**259**	12	78	34	173449	12571	176400	0
	8	Learning	**0.588**	0.100	1.017	0.430	**357**	74	245	44	280,347	45,752	384,000	0
		Decision	**0.559**	0.397	1.246	1.592	**299**	135	117	48	233,937	104,281	288,000	0
	9	Learning	**0.610**	0.307	0.857	0.464	475	69	**495**	46	411,815	76,073	597,600	0
		Decision	**0.327**	0.136	0.960	0.807	**432**	111	225	63	343,458	94,103	448,200	0

Table 3 continued

Problem	k	Phase	R-IGD				Contribution to CF				FEs			
			iRVEA		iNSGAIII		iRVEA		iNSGAIII		iRVEA		iNSGAIII	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
DTLZ4	3	Learning	0.105	0.029	**0.070**	0.025	157	31	**421**	2	107,486	4066	127,200	0
		Decision	0.085	0.047	**0.077**	0.040	222	29	**307**	6	89,427	3968	95,400	0
	4	Learning	0.255	0.128	**0.171**	0.179	282	11	**482**	1	169,013	6052	192,000	0
		Decision	0.636	0.437	**0.617**	0.430	336	8	**356**	2	142,216	2098	144,000	0
	5	Learning	0.747	0.748	**0.536**	0.358	425	83	**505**	3	224,266	38,507	252,000	0
		Decision	**0.627**	0.404	0.650	0.445	361	45	**373**	2	186,999	22,702	189,000	0
	6	Learning	0.523	0.199	**0.471**	0.247	484	15	**504**	2	280,748	3146	302,400	0
		Decision	0.554	0.462	**0.519**	0.460	**385**	2	376	3	230,480	1674	226,800	0
	7	Learning	1.267	0.889	**0.701**	0.151	297	63	**330**	13	172,607	38,083	235,200	0
		Decision	0.747	0.350	**0.645**	0.354	244	38	**248**	5	168,479	30,595	176,400	0
	8	Learning	2.260	1.383	**0.665**	0.089	393	92	**483**	1	259,205	74,122	384,000	0
		Decision	0.842	0.453	**0.694**	0.476	**380**	1	360	1	281,918	10,469	288,000	0
	9	Learning	0.800	0.254	**0.708**	0.262	**685**	8	663	1	537,059	29,818	597,600	0
		Decision	1.118	0.325	**1.093**	0.358	**518**	1	496	2	438,909	14,570	448,200	0

Table 4 iRVEA-RP vs. iRVEA-Ranges: numerical results with 100 generations per iteration for the objectives ranging from 3 to 9

Problem	k	Phase	R-IGD				Contribution to CF				FEs			
			iRVEA-RP		iRVEA-Ranges		iRVEA-RP		iRVEA-Ranges		iRVEA-RP		iRVEA-Ranges	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	std.
DTLZ1	3	Learning	**2.473**	0.106	2.604	0.196	**150**	69	34	52	102,132	11,138	72,373	31,324
		Decision	**1.859**	0.216	1.873	0.203	**292**	48	64	46	91,521	2117	95,012	20,800
	4	Learning	**2.762**	0.096	2.896	0.152	**210**	114	177	108	157,746	6878	159,461	24,309
		Decision	1.787	0.095	**1.775**	0.104	**349**	22	275	126	134,653	6757	167,615	3311
	5	Learning	**2.780**	0.117	2.977	0.180	**302**	86	234	135	214,526	23,343	225,120	20,445
		Decision	2.106	0.241	**2.103**	0.239	**358**	59	291	148	189,002	5388	212,499	46,312
	6	Learning	**2.805**	0.127	3.170	0.154	331	75	**376**	173	262,012	9682	288,029	32,166
		Decision	**1.959**	0.332	1.961	0.291	**384**	17	352	155	220,375	7395	247,406	96,321
	7	Learning	**2.916**	0.225	3.505	0.336	211	107	**312**	134	216,973	5618	256,392	21,765
		Decision	2.035	0.375	**2.025**	0.336	**200**	57	185	142	180,099	5427	187,227	108,020
	8	Learning	**2.964**	0.163	3.627	0.412	347	112	**350**	244	337,488	26,067	388,893	39,496
		Decision	**1.953**	0.342	2.011	0.277	**348**	62	203	206	287,031	8905	220,664	189,815
	9	Learning	**2.835**	0.150	3.637	0.333	**577**	85	570	238	524,079	31,435	594,523	46099
		Decision	1.983	0.407	**1.980**	0.367	**514**	3	449	253	452,580	10,169	460,364	241,346
DTLZ2	3	Learning	**0.195**	0.207	0.362	0.195	**204**	56	62	62	89747	17,216	24,349	22,056
		Decision	**0.319**	0.481	0.881	0.597	**192**	79	14	5	70,330	28,593	15,228	21,819
	4	Learning	**0.564**	0.421	0.710	0.476	**257**	99	208	97	100,852	34,106	102,171	43,516
		Decision	0.997	0.792	**0.879**	0.838	**131**	135	59	59	62,446	61,676	59,605	69,507
	5	Learning	**0.280**	0.190	0.560	0.148	**396**	59	301	117	169,164	30,901	140,704	66,235
		Decision	**0.530**	0.517	0.751	0.614	**309**	124	126	96	157,756	61,041	153,731	81,323
	6	Learning	**0.529**	0.337	0.566	0.090	**394**	118	372	118	188971	62,602	205,655	71,323
		Decision	1.689	1.849	**1.414**	1.583	**269**	162	242	129	159,317	94,462	195,966	100,939

Table 4 continued

Problem	k	Phase	R-IGD				Contribution to CF				FEs			
			iRVEA-RP		iRVEA-Ranges		iRVEA-RP		iRVEA-Ranges		iRVEA-RP		iRVEA-Ranges	
			Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	std.
	7	Learning	**0.486**	0.216	0.940	0.343	248	62	**276**	87	136,385	37,755	184,031	45,542
		Decision	1.155	1.565	**0.897**	1.369	**177**	109	166	99	122,086	73,071	177,301	93,222
	8	Learning	**0.476**	0.267	0.784	0.285	**362**	55	335	142	209,208	36,673	224,348	104,465
		Decision	2.458	2.049	**0.969**	0.424	182	140	**385**	180	127,167	99,390	322,144	108,602
	9	Learning	**0.528**	0.274	0.908	0.213	**510**	142	423	221	341,056	112,124	389,291	180654
		Decision	2.909	2.381	**1.562**	1.444	222	215	**397**	231	172,533	162,562	358,663	204,173
DTLZ3	3	Learning	**0.188**	0.237	0.649	0.440	**88**	44	16	14	91,701	9474	53,486	27128
		Decision	0.213	0.305	**0.172**	0.175	**175**	102	27	31	71,843	33,198	61,302	41,075
	4	Learning	**0.234**	0.147	0.651	0.338	**174**	44	44	44	148,334	23,573	110,230	56,902
		Decision	**0.427**	0.643	0.570	0.611	**249**	134	25	38	107,189	47,229	71,661	70,361
	5	Learning	**0.464**	0.174	1.094	0.667	**175**	84	155	156	185,903	28,649	140,813	77,061
		Decision	**0.223**	0.291	0.533	0.402	**283**	113	94	128	157,741	49,214	81,671	93,460
	6	Learning	**0.707**	0.451	1.079	0.273	**224**	109	157	118	205,280	44,914	209,504	56,752
		Decision	**0.272**	0.120	0.432	0.409	**293**	85	117	106	171,301	37,604	192,897	114,324
	7	Learning	**0.525**	0.416	1.461	0.447	190	62	**233**	137	182,259	31,951	225,480	41,383
		Decision	0.575	0.969	**0.379**	0.293	**185**	74	173	117	158,003	37,831	205,151	49,427
	8	Learning	**0.732**	0.418	1.548	0.675	**323**	51	260	189	288,271	35,718	354,430	79,190
		Decision	0.710	0.873	**0.482**	0.773	**190**	154	180	104	154,163	123,620	272,603	125,257
	9	Learning	**0.369**	0.093	1.548	0.587	**503**	83	374	182	470,856	52,312	466,521	80,068
		Decision	0.658	0.544	**0.330**	0.248	**323**	217	296	150	275,755	184,561	471,451	173,463
DTLZ4	3	Learning	**0.065**	0.035	0.634	0.723	**258**	46	99	97	107,643	2929	50,189	28,180
		Decision	**0.097**	0.052	0.157	0.081	**251**	19	80	34	90,690	1392	88,773	6430
	4	Learning	**0.182**	0.163	0.358	0.111	**380**	52	263	169	169,891	7496	116,161	56,912
		Decision	**0.268**	0.342	0.313	0.364	**318**	13	127	40	141,123	3540	139,299	15,072
	5	Learning	**0.777**	1.239	0.834	0.699	**433**	109	410	129	212,099	57,517	208,734	63783
		Decision	**0.577**	0.626	0.628	0.537	**332**	78	134	69	174,246	41,674	146,119	52,942
	6	Learning	**0.552**	0.169	1.101	0.667	500	12	**511**	97	286,270	13,417	291,382	57,995
		Decision	0.657	0.423	**0.635**	0.489	**381**	8	319	96	226,506	7408	259,833	11,953
	7	Learning	**2.400**	2.691	2.989	2.526	212	128	**247**	141	133,483	82,945	158,791	95112
		Decision	**1.469**	1.458	1.875	2.312	168	94	**241**	117	119,533	65,213	199,386	92,390
	8	Learning	**1.446**	2.097	1.561	1.054	457	141	**561**	85	343,166	103,225	423,696	70,082
		Decision	0.885	0.359	**0.664**	0.518	368	32	**407**	113	281,463	27,964	353,865	69,693
	9	Learning	**0.931**	0.323	1.618	0.467	684	6	**724**	148	570,285	11,818	642,717	70,788
		Decision	0.743	0.346	**0.539**	0.509	**515**	3	462	155	446,762	13,336	510,922	59,213

References

1. Afsar B, Miettinen K, Ruiz AB (2021) 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. Springer, pp 619–631
2. Afsar B, Miettinen K, Ruiz F (2021) Assessing the performance of interactive multiobjective optimization methods: a survey. ACM Comput Surv 54(4):85
3. Barba-González C, Ojalehto V, García-Nieto J.M, Nebro AJ, Miettinen K, Aldana-Montes JF (2018) Artificial decision maker driven by PSO: an approach for testing reference point based interactive methods. In: Auger A, Fonseca CM, Lourenço N, Machado P, Paquete L, Whitley D (eds) Parallel problem solving from nature—PPSN XV, 15th international conference, Proceedings, Part I. Springer, pp 274–285
4. Branke J, Deb K, Miettinen K, Slowinski R (eds) (2008) Multiobjective optimization. Interactive and evolutionary approaches. Springer, Berlin

5. Cheng R, Jin Y, Olhofer M, Sendhoff B (2016) A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput 20(5):773–791
6. Chugh T, Jin Y, Miettinen K, Hakanen J, Sindhya K (2018) A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. IEEE Trans Evol Comput 22(1):129–142
7. Cornell JA (2011) Experiments with mixtures: designs, models, and the analysis of mixture data. Wiley, New York
8. Deb K, Jain H (2013) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Trans Evol Comput 18(4):577–601
9. Deb K, Miettinen K, Chaudhuri S (2010) Towards an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches. IEEE Trans Evol Comput 14(6):821–841
10. Deb K, Thiele L, Laumanns M, Zitzler E (2002) Scalable multi-objective optimization test problems. In: 2002 congress on evolutionary computation, Proceedings, pp 825–830
11. Hakanen J, Chugh T, Sindhya K, Jin Y, Miettinen K (2016) Connections of reference vectors and different types of preference information in interactive multiobjective evolutionary algorithms. In: 2016 IEEE symposium series on computational intelligence, Proceedings. IEEE, pp 1–8
12. Hou Z, Yang S, Zou J, Zheng J, Yu G, Ruan G (2018) A performance indicator for reference-point-based multiobjective evolutionary optimization. In: 2018 IEEE symposium series on computational intelligence, Proceedings. IEEE, pp 1571–1578
13. Huber S, Geiger MJ, Sevaux M (2015) Simulation of preference information in an interactive reference point-based method for the bi-objective inventory routing problem. J Multi-Criteria Decis Anal 22(1–2):17–35
14. Li K, Deb K, Yao X (2018) R-metric: evaluating the performance of preference-based evolutionary multiobjective optimization using reference points. IEEE Trans Evol Comput 22(6):821–835
15. Li M, Yao X (2019) Quality evaluation of solution sets in multiobjective optimisation: a survey. ACM Comput Surv 52(2):26
16. López-Ibánez M, Knowles J (2015) Machine decision makers as a laboratory for interactive EMO. In: Gaspar-Cunha A, Henggeler-Antunes C, Coello CC (eds) Evolutionary multi-criterion optimization, 8th international conference, Proceedings, Part II. Springer, pp 295–309
17. Luque M, Ruiz F, Miettinen K (2011) Global formulation for interactive multiobjective optimization. OR Spectrum 33(1):27–48
18. Meignan D, Knust S, Frayret JM, Pesant G, Gaud N (2015) A review and taxonomy of interactive optimization methods in operations research. ACM Trans Interact Intell Syst 5(3):171–1743
19. Miettinen K (1999) Nonlinear multiobjective optimization. Kluwer Academic Publishers, Boston
20. Miettinen K, Hakanen J, Podkopaev D (2016) Interactive nonlinear multiobjective optimization methods. In: Greco S, Ehrgott M, Figueira J (eds) Multiple criteria decision analysis: state of the art surveys, 2 edn. Springer, pp 931–980
21. Miettinen K, Ruiz F, Wierzbicki AP (2008) Introduction to multiobjective optimization: interactive approaches. In: Branke J, Deb K, Miettinen K, Słowiński R (eds) Multiobjective optimization: interactive and evolutionary approaches. Springer, Berlin, pp 27–57
22. Misitano G, Saini BS, Afsar B, Shavazipour B, Miettinen K (2021) DESDEO: The modular and open source framework for interactive multiobjective optimization. IEEE Access 9:148277–148295
23. Mohammadi A, Omidvar MN, Li X(2013) A new performance metric for user-preference based multi-objective evolutionary algorithms. In: 2013 IEEE congress on evolutionary computation, Proceedings. IEEE, pp 2825–2832
24. Ojalehto V, Podkopaev D, Miettinen K (2016) Towards automatic testing of reference point based interactive methods. In: Handl J, Hart E, Lewis PR, López-Ibánez M, Ochoa G, Paechter B (eds) Parallel problem solving from nature—PPSN XIV, 14th international conference, Proceedings. Springer, pp 483–492
25. Ruiz F, Luque M, Miettinen K (2012) Improving the computational efficiency in a global formulation (GLIDE) for interactive multiobjective optimization. Ann Oper Res 197(1):47–70
26. Steuer RE (1986) Multiple criteria optimization: theory, computation and application. Wiley, New York
27. Szczepanski M, Wierzbicki AP (2003) Application of multiple criteria evolutionary algorithm to vector optimization, decision support and reference-point approaches. J Telecommun Inf Technol 3(3):16–33
28. Wierzbicki AP (1980) The use of reference objectives in multiobjective optimization. In: Fandel G, Gal T (eds) Multiple criteria decision making, theory and applications. Springer, Berlin, pp 468–486
29. Xin B, Chen L, Chen J, Ishibuchi H, Hirota K, Liu B (2018) Interactive multiobjective optimization: a review of the state-of-the-art. IEEE Access 6:41256–41279
30. Yu G, Zheng J, Li X (2015) An improved performance metric for multiobjective evolutionary algorithms with user preferences. In: 2015 IEEE congress on evolutionary computation, Proceedings. IEEE, pp 908–915