A Preference Based Interactive Evolutionary Algorithm for Multi-objective Optimization: PIE

Karthik Sindhya[1], Ana Belen Ruiz[2], and Kaisa Miettinen[1]

[1] Department of Mathematical Information Technology, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland
karthik.sindhya@jyu.fi
kaisa.miettinen@jyu.fi

[2] Department of Applied Economics (Mathematics)
University of Malaga
C/ Ejido 6, 29071, Malaga, Spain
abruiz@uma.es

Abstract. This paper describes a new Preference-based Interactive Evolutionary (PIE) algorithm for multi-objective optimization which exploits the advantages of both evolutionary algorithms and multiple criteria decision making approaches. Our algorithm uses achievement scalarizing functions and the potential of population based evolutionary algorithms to help the decision maker to direct the search towards the desired Pareto optimal solution. Starting from an approximated nadir point, the PIE algorithm improves progressively the objective function values of a solution by finding a better solution at each iteration that improves the previous one. The decision maker decides from which solution, in which direction, and at what distance from the Pareto front to find the next solution. Thus, the PIE algorithm is guided interactively by the decision maker. A flexible approach is obtained with the use of archive sets to store all the solutions generated during an evolutionary algorithm's run, as it allows the decision maker to freely navigate and inspect previous solutions if needed. The PIE algorithm is demonstrated using a pollution monitoring station problem and shown to be effective in helping the decision maker to find a solution that satisfies her/his preferences.

1 Introduction

Optimization problems arising in all areas of science, engineering, and economics often involve multiple conflicting objectives. Such problems are multi-objective optimization problems. A solution to a multi-objective optimization problem desirably belongs to a set of mathematically equally good compromise solutions with different trade-offs, called Pareto optimal solutions. In addition, it should satisfy the preferences of a decision maker (DM). The set of Pareto optimal solutions in the objective space is called Pareto front. Different solution approaches to solve multi-objective optimization problems exist in the literature, among them approaches of Evolutionary Multi-objective Optimization (EMO) [3,4] and Multiple Criteria Decision Making (MCDM) [2,16] are commonly used.

Evolutionary multi-objective optimization algorithms deal with a population of solutions and try to find a set of well-distributed non-dominated solutions that approximate the entire Pareto front. For the past two decades, several EMO algorithms have been proposed [3,4] and they have been shown to be very versatile in handling different types of variables and objectives. Compared to EMO, the main goal in MCDM is to aid the DM in finding a satisfying solution according to her/his preference information. Commonly, a multi-objective problem is scalarized into a single-objective problem taking into account the DM's preference information, which can be expressed e.g. with trade-offs, pairwise comparisons, aspiration levels, reference points and/or classification of the objective functions. The MCDM approaches can be broadly classified in to *a priori*, *a posteriori* and *interactive* methods, depending on when the preference information is asked from the DM [15]. For a detailed description of different MCDM approaches see [16]. Although only the most interesting Pareto optimal solutions are found by MCDM approaches, which saves computational time, their success in solving real world problems depends on the method used to solve scalarized single objective optimization problems (e.g. whether the solution is globally or locally optimal).

Advantages of both EMO and MCDM approaches can be utilized by hybridizing them. At least two possible ways can be identified: "evolutionary algorithm in MCDM" and "MCDM in EMO" approaches. Utilizing (single objective) population based approaches such as evolutionary algorithms to solve the scalarized functions formulated in MCDM approaches (we call this approach as "evolutionary algorithms in MCDM") has received some attention (see e.g. [12,17]). In this way, nonconvexity or discontinuity in the problem or possible binary or integer variables can be handled in a straightforward way. On the other hand, in "MCDM in EMO" approach based algorithms (see e.g. [1,8,9,21]), instead of approximating the entire Pareto front, the main goal is to find a solution or a set of solutions which approximates a region of the Pareto front that is interesting to the DM. Thus the DM gets a preferred set of solutions and (s)he can choose the most satisfactory solution among them.

On one hand, showing solutions only in the preferred region of the Pareto front allows the DM to inspect only a handful of solutions and can gain insights about the trade-offs between the objectives in the desired region and the final solution can be selected from a knowledgeable position. On the other hand, such an approach limits the interactions of the DM with the algorithm. In such methods, the effort is concentrated in looking just for the solutions in the preferred region(s) of the Pareto front, so a faster algorithm is obtained as irrelevant solutions are discarded, but the full potential of a population based approach to help the DM to freely navigate, inspect, and restate preferences during the solution process is yet to be fully explored. For us, any approach that adapts to the DM's desires as much as possible and that allows her/him to modify the preferences during an algorithm's run time is the best choice enabling learning of the problem and one's preferences. This is the basic idea of the Preference based Interactive Evolutionary algorithm for multi-objective optimization (PIE) that we propose in this paper.

In the PIE algorithm, the search for the most preferred solution satisfying the preference information of the DM follows nearly the same philosophy as the interactive NAUTILUS method [18]. The NAUTILUS method is based on the assumptions that past experiences affect DM's hopes and that people's reactions to gains and losses are asymmetric and, thus, a need for trading off between Pareto optimal solutions during the solution process may hinder the DM from finding desirable solutions. This method starts from a nadir point, a vector whose components are the worst objective function values of solutions and aids the DM to obtain improvement in each objective in the direction specified by her/him progressively without a need of trading off. In every step taken by the DM, information about the part of the Pareto front dominating the current solution (i.e., part which still can be reached without trade-off) is provided. With this information (s)he can re-specify her/his preference information and new solutions are generated in this direction. Although only the last solution will be Pareto optimal (guaranteed by MCDM tools used), a solution is obtained at each step which dominates the previous one and each new Pareto optimal solution is obtained by minimizing an achievement scalarizing function including preferences about desired improvements in objective function values.

In the NAUTILUS method, the starting solution is intentionally bad and the algorithm progressively improves the initial solution to obtain a Pareto optimal solution satisfying the DM, trying to avoid a possible anchoring effect [22] developed by expectations from previous (Pareto optimal) solutions. A typical evolutionary algorithm also has similar traits as that of the NAUTILUS method. In an evolutionary algorithm we start with a random initial population which is usually far from the optimal solutions, similar to the nadir point used in the NAUTILUS method. Additionally, an evolutionary algorithm drives the initial population to the optimal solutions progressively by generating better solutions than the initial population, as NAUTILUS does.

Our PIE algorithm is an "evolutionary algorithm in MCDM" approach. A starting solution is chosen from the initial population based on the preference information of the DM. Subsequently (s)he directs the search deciding from which solution and in which direction the algorithm has to look for solutions dominating the current one. Furthermore, (s)he sets the speed of approaching the Pareto front. The method enables the DM to consider solutions where all objectives can be improved without a need to sacrifice in anything. This should enable a free search towards a desirable part of Pareto optimal solutions. The idea is not to get into a situation where tradeoffs are needed "too early" in the solution process because the need to give up in some objective may hinder the DM from looking for new solutions. As in the NAUTILUS method, although we ensure Pareto optimality only for the final solution (once we have reached the Pareto front), we present solutions that improve the previous ones step by step, what is likely to be attractive for the DM, as people prefer to get better results.

When the DM chooses to select a new solution and direction to orient the search, (s)he is not forced to select it from the presented solutions but can decide to move backwards or present a new point describing her/his aspirations. The PIE algorithm stores all the solutions of the evolutionary algorithm in an archive set to be able to recover previous solutions if the DM desires to go backwards.

The rest of the paper is organized as follows. In Section 2, we introduce some concepts needed for our algorithm. Then, the PIE algorithm is presented in Section 3, where a detailed description of the step-by-step scheme is explained, with some technical issues. Section 4 describes a numerical example demonstrating the method. Finally, conclusions are drawn in Section 5.

2 Concepts

We consider multi-objective optimization problems of the form:

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

where we want to optimize k ($k \geq 2$) conflicting objective functions $f_i : S \to \mathbb{R}$ simultaneously. Here S is the *feasible set* in the *decision space* $\mathbb{R}^n$ and *decision vectors* are $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in S$. Vectors in the image set of S are *objective vectors* in the *objective space* $\mathbb{R}^k$. A decision vector $\overline{\mathbf{x}} \in S$ is *Pareto optimal* if no objective function can achieve an improvement without a loss in other objective(s), in other words, if there is no other $\mathbf{x} \in S$ such that $f_i(\mathbf{x}) \leq f_i(\overline{\mathbf{x}})$ for all $i = 1, ..., k$ and $f_j(\mathbf{x}) < f_j(\overline{\mathbf{x}})$ for at least one $j \in 1, ..., k$. The corresponding objective vector is called a *Pareto optimal* objective vector. We say that a vector $\mathbf{z}^1 \in \mathbb{R}^k$ dominates another vector $\mathbf{z}^2 \in \mathbb{R}^k$, denoted by $\mathbf{z}^1 \leq \mathbf{z}^2$, if $z_i^1 \leq z_i^2$ for all $i = 1, ..., k$ and $z_j^1 < z_j^2$ for at least one $j \in 1, ..., k$.

When we want to solve a multi-objective optimization problem, we are interested in finding the Pareto optimal solution that best satisfies the desires of a *decision maker*, a person who is interested in solving the problem and can express preferences in some way. Two important objective vectors are defined as they fix the ranges of the Pareto front, the *ideal objective vector* and the *nadir objective vector*. The components z_i^* of an *ideal objective vector* $\mathbf{z}^* \in \mathbb{R}^k$ are obtained by minimizing each objective function individually, that is, $z_i^* = \min\limits_{\mathbf{x} \in S} f_i(\mathbf{x})$ for all $i = 1, ..., k$. The components $\mathbf{z}_i^{nad}$ of a *nadir objective vector* $\mathbf{z}^{nad} \in \mathbb{R}^k$ are obtained by considering the maximum objective function value for each objective function across all solutions in the Pareto front. The nadir objective vector is generally more difficult to obtain and typically we approximate it [6,16].

The proposed PIE algorithm is based on reference points, introduced in [23]. A *reference point* is an objective vector $\mathbf{q} \in \mathbb{R}^k$ typically provided by the DM. Using a reference point, an *achievement scalarizing function* (ASF) [23] can be formed and optimized to find a solution that best satisfies the aspirations of the DM. By minimizing ASF, we project the reference point $\mathbf{q}$ in the direction specified by a weight vector $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$ to find the closest Pareto optimal solution. For details of ASFs, see [16].

Given a reference point $\mathbf{q} \in \mathbb{R}^k$, an example of an ASF [23] is given by:

$$\begin{array}{ll} \text{minimize} & \max_{i=1,\dots,k} \{ \mu_i(f_i(\mathbf{x}) - q_i) \} + \rho \sum_{i=1}^{k} (f_i(\mathbf{x}) - q_i) \\ \text{subject to} & \mathbf{x} \in S, \end{array} \tag{2}$$

where $\rho > 0$ is a so-called augmentation coefficient. Different values for the weights have been proposed in [15]. Among the main advantages of the use of ASFs, we can highlight that the optimal solution of (2) is always a Pareto optimal solution of (1) and that any (properly) Pareto optimal solution of (1) can be obtained by just solving problem (2) with different reference points [16]. Reference points can be of two types: aspiration points (objective function values that are desirable to the DM) and reservation points (objective function values that should be attained, if possible). Additionally, a reference point can be referred to as *attainable*, if the solution obtained by projecting the reference point on to the Pareto front dominates the reference point and otherwise referred to as *unattainable*. In our study, we consider a reference point ($\mathbf{q}$) to be a reservation point.

In [20,24], a different ASF is proposed if both aspiration values q_i^a and reservation values q_i for each objective function are available:

$$\begin{array}{ll} \text{minimize} & \max_{i=1,\dots,k} \begin{cases} -1 + \alpha \frac{f_i(\mathbf{x}) - q_i^a}{q_i^a - z_i^*} & \text{if } z_i^* \le f_i(\mathbf{x}) \le q_i^a \\ \frac{f_i(\mathbf{x}) - q_i}{q_i - q_i^a} & \text{if } q_i^a \le f_i(\mathbf{x}) \le q_i \\ \beta \frac{f_i(\mathbf{x}) - q_i}{z_i^{nad} - q_i} & \text{if } q_i \le f_i(\mathbf{x}) \le z_i^{nad} \end{cases} \\ \text{subject to} & \mathbf{x} \in S, \end{array} \tag{3}$$

where α and β are strictly positive numbers. The three cases of expression (3) are defined for three different cases: for attainable reference points in the first case, for unattainable reference points in the third case and the second case is suitable when $\mathbf{q}$ is attainable, but $\mathbf{q}^a$ is not. For more details, see [20]. In our PIE algorithm we can use (2) or (3) depending on the type of preference information available from the DM.

3 The Proposed PIE Algorithm

The PIE algorithm is an interactive preference based evolutionary algorithm, to aid the DM to find a solution satisfying her/his preference information. An evolutionary algorithm is used here because an evolutionary algorithm starts from an initial random population, which is usually far from the Pareto front. Subsequently, the population is driven progressively towards the Pareto front. The population at every generation can be saved in an archive, which not only helps the DM to examine previous solutions, but also to generate a new initial population for an evolutionary algorithm used to solve a new scalarized problem formulated with the new preference information from the DM.

Both the PIE algorithm and an evolutionary algorithm involve seperately a sequence of steps, which are repeated until a termination criterion is satisfied. To make a distinction, we refer to a step of the PIE algorithm as an iteration and a step of an evolutionary algorithm as a generation. Next, we describe the essential components of the PIE algorithm and then present the PIE algorithm.

3.1 Components of the PIE Algorithm

1. **Approximation of nadir point.** To start the PIE algorithm, we need an approximation of the nadir point, which will be the first reference point. We can generate an initial random population and the DM can either decide to select an approximation of the nadir point from a sample of solutions of this population or start from an approximated nadir point constructed by using the worst objective value in every objective function of this initial population. Alternatively, we can apply the approach of [6] or ask the DM to specify worst thinkable objective function values. During the algorithm, the DM is free to choose any other solution as a reference point according to her/his preferences.
2. **Preference information as weights.** Once the DM has chosen a reference point $\mathbf{z}^h$ at iteration h, (s)he has to specify some preference information. As in [15,18], the DM is asked to express her/his preference information and the search for more preferred solutions is oriented accordingly. This preference information can be provided e.g. either by ranking objectives into index sets or by sharing 100 credits among all the objectives.
   (a) **LPF1**: The DM can rank the objectives according to the relative importance of improving each current objective value. (S)he assigns objective functions to classes in an increasing order of importance for improving the corresponding current objective value z_i^h. This importance evaluation allows us to allocate the k objective functions into index sets J_r which represent the importance levels $r = 1, ..., s$, where $1 \leq s \leq k$. If $r < t$, then improving the current objective function values in the index set J_r is less important than improving the current objective function values in J_t. Each objective function can only belong to one index set, but several objectives can be assigned to the same index set J_r.
   The weights for objectives f_i in J_r are defined in [15,18] as follows:

$$\mu_i^h = \frac{1}{r(z_i^{nad} - z_i^{**})}, \; i = 1, ..., k.$$

   (b) **LPF2**: Alternatively, in order to find solutions that reflect the preferences of the DM, the DM can specify percentages expressing how much (s)he would like to improve the current objective values. Then, we ask the following question: *Assuming you have one hundred credits available, how would you distribute them among the current objective values so that the more credits you allocate, the more improvement of the corresponding current objective value is desired?.*

Let us assume that the DM has given p_i^h credits to the objective function f_i. We set $\Delta q_h^i = \frac{p_i^h}{100}$ and then, the weights are formulated in [15]:

$$\mu_i^h = \frac{1}{\Delta q_h^i (z_i^{nad} - z_i^{**})}, \ i = 1, ..., k.$$

We assume that $1 \leq p_i^h \leq 100$, for all $i = 1, ..., k$.

However, the mode of providing preference information may not be limited to the above two ways. Once this preference information is obtained from the DM, we formulate an ASF, which is solved using an evolutionary algorithm.

3. **Aspiration and reservation levels.** Instead of choosing the reference point, the DM can specify bounds on the objective functions as aspiration and reservation levels, $\mathbf{q}^a$ and $\mathbf{q}$ respectively. Here, we replace an ASF with an aspiration-reservation scalarizing function (3). It must be noted that the DM may provide unattainable aspiration levels, in which case, the Pareto optimal solution obtained by solving problem (3) will not satisfy her/his expectations completely. In order to make him/her conscious of that situation, we make the following remark: *The aspiration levels are too optimistic and cannot be achieved.*
4. **Termination criterion for evolutionary algorithm.** As mentioned earlier, we use an evolutionary algorithm to solve the ASF problem and define a termination criterion based on the ASF using the fact that the optimal ASF value is *zero* for Pareto optimal reference points [16,23]. At each η generations, we initialize the reference point used in the ASF with the objective function values of the solution having the smallest ASF value at this generation. Then, the ASF formulated with this reference point is tracked for every $\eta + 10$ generations. If the running average of the ASF values for $\eta + 10$ generations is close to *zero* i.e., 10^{-05}, then the evolutionary algorithm is terminated. This termination criterion ensures that no further improvement in the ASF value is possible or in other words, an individual, $\mathbf{z}^{(h)}$, with the lowest ASF value is in the proximity of the Pareto front.

Next, we describe the step-by-step procedure involved in the PIE algorithm.

3.2 PIE Algorithm

Step 1: Calculate the ideal point $\mathbf{z}^*$ and generate an initial random population P_0 of feasible solutions.

Step 2: The nadir point $\mathbf{z}^{nad}$ is found among the solutions of P_0 depending on the information the DM wishes to give:

(a) If the DM desires to choose $\mathbf{z}^{nad}$ from P_0, we ask: *How many solutions would you like to see to identify an approximation of the nadir point?* and we refer to the number of solutions specified by the DM as N. Next, P_0 is divided into N clusters and one solution from each cluster is presented to the DM. If a cluster has more than one solution, we select the solution closest to the centroid of that cluster. We present these N solutions to the DM, from which (s)he selects one as $\mathbf{z}^{nad}$ (or use ideas of Subsection 3.1).

(b) If the DM is not able to give any information, $\mathbf{z}^{nad}$ is constructed of the worst objective values present in the initial population P_0.

Step 3: Let A and AP be the archive sets of solutions and populations, respectively, defined in Subsection 3.3. Set $A = \varnothing$ and $AP = \varnothing$. Let $\mathbf{z}^1$ be the first reference point and h the iteration number. Set $\mathbf{z}^1 = \mathbf{z}^{nad}$ and $h = 1$.

Step 4: The DM has to provide information about the local improvement of each objective at the current reference point, $\mathbf{z}^h$, in one of the two proposed ways (LPF1 and LPF2) and the weights, μ_i^h are defined accordingly for all $i = 1, \ldots, k$.

Step 5: Set $\mathbf{q} = \mathbf{z}^h$ and $\mu_i = \mu_i^h$ for all $i = 1, \ldots, k$. An achievement scalarizing problem is formulated as in (2). Go to step 7.

Step 6: Formulate the aspiration-reservation scalarizing function (3) with $\mathbf{q}^a = \mathbf{z}^h$ and $\mathbf{q} = \mathbf{z}^{p_{h-1}}$.

Step 7: A single-objective evolutionary algorithm is used to solve the scalarizing function with the termination criterion of Subsection 3.1. Set the initial population needed to start the evolutionary algorithm as explained in Subsection 3.3. At each generation of the evolutionary algorithm, the solution with the lowest ASF value is stored in A and the current population is saved in AP.

Step 8: When the evolutionary algorithm has terminated, select the solution in the final population with the smallest ASF value, $\mathbf{z}^{(h)}$, to be an approximation of the projection of $\mathbf{z}^h$ in the Pareto front. If the solved problem is of the type (3) and $\mathbf{z}^{(h)}$ is dominated by $\mathbf{q}$, the DM is told: *The aspiration levels are too optimistic and cannot be achieved.*

Step 9: Calculate the Euclidean distance d_P between $\mathbf{z}^h$ and $\mathbf{z}^{(h)}$ as $d_P = \sum_{i=1}^{k} \frac{|z_i^{(h)} - z_i^h|}{k|z_i^{nad} - z_i^*|}$, where $|\cdot|$ is the absolute value. Ask the DM: *The distance is d_P between the current reference point $\mathbf{z}^h$ and the corresponding projected Pareto optimal solution $\mathbf{z}^{(h)}$. At what percentage of that distance from $\mathbf{z}^{(h)}$ do you like to investigate the next solution?*. The percentage is referred to as p_h. Next, calculate the point $\mathbf{z}^{p_h}$ in the objective space at p_h of the distance d_P from $\mathbf{z}^{(h)}$.

Step 10: Show current iteration solution. Show $\mathbf{z}^{p_h}$ to the DM. If the DM is satisfied with $\mathbf{z}^{p_h}$, set $\mathbf{z}^h = \mathbf{z}^{p_h}$ and go to step 12, else go to step 11.

Step 11: New reference point. Set $h = h + 1$. The DM has to select the next reference point $\mathbf{z}^h$ in one of the following ways:

(a) Otherwise ask the DM: *Would you like to continue the search from $\mathbf{z}^{p_{h-1}}$?* If yes, set $\mathbf{z}^h = \mathbf{z}^{p_{h-1}}$. Next, ask the DM: *Would you like to specify new local improvement information at the new reference point, $\mathbf{z}^h$?* If yes, go to step 4. Else, go to step 9.

(b) Otherwise ask the DM: *Would you like to examine some solutions in the previous generations?* If yes, we ask: *How many solutions would you like to examine?* and we refer to this number as N. Find N solutions in A as described in Subsection 3.3 and show them to the DM. If the DM finds an interesting solution (set it as $\mathbf{z}^h$) and wishes to examine more solutions around $\mathbf{z}^h$, we ask: *How many solutions would you like to examine around* $\mathbf{z}^h$*?* and we refer to this number as N. Find N solutions in AP as described in Subsection 3.3 and show them to the DM. Finally, when the DM has found a new reference point $\mathbf{z}^h$, ask: *Would you like to specify a new local improvement information at the new reference point,* $\mathbf{z}^h$*?* If yes, go to step 4. Else, set $\mu_i^h = \mu_i^{h-1}$ for all $i = 1, \ldots, k$ and go to step 5.

(c) Otherwise ask the DM: *Would you like to provide an aspiration point?* If yes, set it as $\mathbf{z}^h$ and go to step 6. If no, go to Step 12.

Step 12: Using $\mathbf{z}^h$ as the reference point, solve problem (2) with any suitable mathematical programming technique. The resulting solution is declared as the solution preferred by the DM satisfying her/his preference information.

Step 12 is applied to guarantee at least the local Pareto optimality of the final solution. Furthermore, the solution $\mathbf{z}^h$ is shown to the DM only if the DM sets a 0% distance.

3.3 The Archive Sets A and AP

Archive sets of solutions A and populations AP are maintained during the evolutionary algorithm. At each generation of the evolutionary algorithm, the solution with the ASF value closest to *zero* is stored in A and the corresponding population is saved in AP. The archive sets allow an easy access to previous solutions when the DM decides to move backwards or examine some solutions around the current solution. Additionally, they enable to find an initial population for a new run of the evolutionary algorithm when a new reference point is specified by the DM. Storing many solutions should not be considered as a disadvantage. The high performance and large memory of current computers enable us to store a large number of solutions and to look for some solutions among them. In addition, use of a *structured query language* to handle large number of solutions can be explored.

To show some solutions of previous generations or around $\mathbf{z}^{p_{h-1}}$ in Step 11 (b), N solutions in the archive set A with the smallest Euclidean distances to $\mathbf{z}^{p_{h-1}}$ are found, where N is the number of solutions the DM wants to examine. Additionally, if the DM wants to examine N solutions around any solution $\mathbf{s}^i$, we recover the population $P_{\mathbf{s}^i}$ corresponding $\mathbf{s}^i$ from AP. Next, we find the non-dominated solutions in $P_{\mathbf{s}^i}$ and cluster them into N clusters showing one solution from each cluster to the DM. If a cluster has more than one solution, we select the solution closest to the centroid of that cluster. Here, we present only the non-dominated solutions.

To set the initial population P needed to run the evolutionary algorithm for solving problem (2) or (3), the archive set AP may decrease the computational effort. If $h = 1$, P is set as the initial random population P_0 generated to find the approximation of the nadir point. If $h > 1$, let $\mathbf{s}$ be the point in A with the smallest Euclidean distance to $\mathbf{z}^h$ and let P_h be the population in AP it belongs to. Then, we set $P = P_h$. In this way, we can save on the number of function evaluations, as instead of starting from a random initial population during each evolutionary algorithm's run, we load a population of solutions that have already been evaluated around the current reference point.

A graphical illustration of the solution process in a bi-objective optimization problem is shown in Figure 1, where we start from a reference point $\mathbf{z}^h$ (approximate nadir point). In Figure 1, black circles represent the individuals in the population of an evolutionary algorithm. In every generation, the population in an evolutionary algorithm and the solution with the ASF value close to *zero* is saved in the archives AP and A respectively as shown in Figure 1. As it can be seen, the DM has specified a projection direction from the current reference point towards the Pareto front and the projection $\mathbf{z}^{(h)}$ of $\mathbf{z}^h$ is obtained after running the evolutionary algorithm. Once the DM has examined solutions at different percentages of the distance between $\mathbf{z}^{(h)}$ and $\mathbf{z}^h$, which are shown with black vertical bars on the line joining $\mathbf{z}^h$ and $\mathbf{z}^{(h)}$, the DM has chosen the solution at 50% of the distance from $\mathbf{z}^{(h)}$ as the next reference point $\mathbf{z}^{h+1}$. Next, a new projection direction for $\mathbf{z}^{h+1}$ towards the Pareto front is specified and a new iteration is started to obtain a new solution $\mathbf{z}^{(h+1)}$. Instead of selecting the solution at 50% of the distance from $\mathbf{z}^{(h)}$ as the next reference point, the DM could have examined some solutions in the archives around the solution or provided an aspiration point.

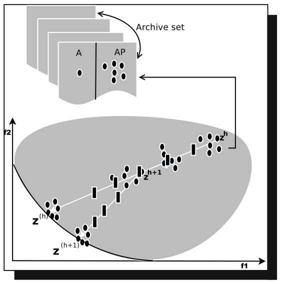

Fig. 1. Graphical idea of the PIE algorithm

4 Numerical Example

In this section, we illustrate how the PIE algorithm can be used to find the DM's preferred Pareto optimal solution. The selected problem is related to finding the location of a pollution monitoring station [19] and has five conflicting objective functions. The objectives pertain to the expected information loss as estimated by five different experts. The problem is formulated as follows:

$$\begin{aligned}
\text{minimize } & f_1(x_1, x_2) = f(x_1, x_2) \\
& f_2(x_1, x_2) = f(x_1 - 1.2, x_2 - 1.5) \\
& f_3(x_1, x_2) = f(x_1 + 0.3, x_2 - 3.0) \\
& f_4(x_1, x_2) = f(x_1 - 1.0, x_2 + 0.5) \\
& f_5(x_1, x_2) = f(x_1 - 0.5, x_2 - 1.7) \\
\text{subject to } & -4.9 \le x_1 \le 3.2 \\
& -3.5 \le x_2 \le 6,
\end{aligned} \tag{4}$$

where

$$f(x_1, x_2) = -3(1-x_1)^2 e^{-x_1^2-(x_2+1)^2} + 10(\frac{1}{4}x_1 - x_1^3 - x_2^5)e^{-x_1^2-x_2^2}$$
$$-\frac{1}{3}e^{-(x_1+1)^2-x_2^2} + 10.$$

Since the main emphasis of the PIE algorithm is to help the DM in finding a solution satisfying her/his preferences, we have chosen a practical problem with five objectives. It is very common in evolutionary computing literature to demonstrate the efficacy of an algorithm by using test problems. But, the objective functions of these problems have no physical meaning and hence it is very difficult to provide preference information by the DM and to illustrate the working of an interactive algorithm, which is our main goal. The example is used to demonstrate the working steps of the PIE algorithm.

A DM used the PIE algorithm to find an optimal location for a pollution monitoring station. Here, the DM was looking for a location that balances between the five possible losses. For simplicity, we used the preference information to be specified using LPF2 to formulate problem (2). The ideal and nadir vectors as specified in steps 1 and 2 of the PIE algorithm remained unaltered throughout the solution process. A real coded genetic algorithm (RGA) [4] was used as an evolutionary algorithm in the PIE algorithm. However, other evolutionary algorithms can also be used. The parameter used were, a) crossover probability = 0.9, b) mutation probability = 0.5, c) SBX distribution index [4] = 15, and d) mutation distribution index [4] = 20.

To start with the algorithm, we generate an initial random population P_0 and calculate the ideal point $\mathbf{z}^* = (1.87, 1.87, 1.87, 1.87, 1.87)^T$ by individually minimizing each of the objectives subject to constraints. Next we summarize the steps taken by the DM with PIE.

Step 1: Iteration 1: To set the first reference point (an approximation of $\mathbf{z}^{nad}$), the DM decided to investigate *three* solutions of P_0. The solutions provided by the algorithm were $(10.00, 10.00, 10.00, 10.00, 10.00)^T$, $(10.00, 9.88, 4.51, 10.00, 9.51)^T$ and $(9.88, 3.45, 7.37, 9.92, 7.14)^T$ and the DM chose to set the first reference point as $\mathbf{z}^1 = (10.00, 10.00, 10.00, 10.00, 10.00)^T$.

Step 2: The DM specified equal credits to all the objective functions i.e. $\Delta q_1 = (0.2, 0.2, 0.2, 0.2, 0.2)^T$. Using this preference information and $\mathbf{z}^1$, problem (2) was formulated and solved using RGA to get $\mathbf{z}^{(1)}$.

Step 3: The DM was asked to specify the percentage distance, p_1. One at a time, the DM investigated solutions at 80%, 50%, 30% and 20% of the total distance from $\mathbf{z}^{(1)}$.

Step 4: At the solution, $\mathbf{z}^{(20_1)} = (9.17, 9.16, 8.87, 9.14, 9.17)^T$ corresponding to the 20% of the total distance from $\mathbf{z}^{(1)}$, the DM wished to specify the aspiration level, $\mathbf{q}^a = (8, 8, 8, 8, 8)^T$ and we set $\mathbf{q} = \mathbf{z}^{(20_1)}$.

Step 5: Iteration 2: Using the preference information from Step 4, problem (3) was formulated (with $\alpha = 0.1$ and $\beta = 10.0$) and solved using RGA to get $\mathbf{z}^{(2)}$. The algorithm found the aspiration levels in $\mathbf{q}^a = (8, 8, 8, 8, 8)^T$ to be unattainable and subsequently, the DM was told: *The aspiration levels are too optimistic and cannot be achieved.*

Step 6: The DM was asked to specify the percentage distance, p_2. Next, the DM investigated individually solutions at 0%, 50% and 10% of the total distance from $\mathbf{z}^{(2)}$ one after another. Finally, the DM chose the solution at 10% of the distance, $\mathbf{z}^{(10_2)} = (8.98, 8.97, 8.61, 8.95, 8.97)^T$.

Step 7: Iteration 3: Next, the DM decided to examine 10 solutions from A, which were selected as described in Subsection 3.3. After investigating these solutions, the DM found a solution, $(8.95, 8.98, 8.61, 8.90, 8.92)^T$, to be set as the next reference point $\mathbf{z}^3$. Next the DM specified equal credits to all the objective functions i.e. $\Delta q_3 = (0.2, 0.2, 0.2, 0.2, 0.2)^T$.

Step 8: Using this preference information problem (2) was formulated and solved using RGA to get $\mathbf{z}^{(3)}$.

Step 9: The DM was asked to specify the percentage distance, p_3 and the DM investigated individually solutions at 10% and 0% of the total distance from $\mathbf{z}^{(3)}$ one after another.

Step 10: The DM was satisfied with $\mathbf{z}^{(0_3)} = \mathbf{z}^{(3)} = (8.95, 8.98, 8.61, 8.90, 8.94)^T$. Using $\mathbf{z}^{(3)}$ as the reference point, problem (2) was solved using SQP. The resulting Pareto optimal solution $(8.95, 8.98, 8.61, 8.90, 8.94)^T$ was declared as the solution preferred by the DM.

In the above demonstration, the DM learnt during the solution process of what solutions were attainable and did not have to trade off between objectives. He initially chose to give equal credits to all the objective functions as balancing between all five possible cases was equally important. At the end of the first iteration, the DM found a solution $\mathbf{z}^{(20_1)} = (9.17, 9.16, 8.87, 9.14, 9.17)^T$ and then looked for a solution satisfying the aspiration level $\mathbf{q}^a = (8, 8, 8, 8, 8)^T$. After solving problem (3), the algorithm found $\mathbf{z}^{(2)} = (8.96, 8.95, 8.59, 8.93, 8.96)^T$ being dominated by $\mathbf{q}^a$. The DM found a new solution by choosing to look for better solutions in the archive. It can be seen that the new solution $\mathbf{z}^3$ was similar to the previous solution $\mathbf{z}^{(10_2)}$, as there were no better solutions in the archive AP. The solution $\mathbf{z}^3$ was a satisfactory solution to the DM, as the objective function values are all between 8.0 and 9.0 and the DM has struck a balance among the five losses. Note that the solution $\mathbf{z}^{(3)}$ was not improved by SQP, as it was already (locally) Pareto optimal but we cannot know that without solving one more problem. For simplicity, we have carried out a small number of iterations. However, in large scale problems the number of iterations can be higher.

5 Conclusions

In this paper, a new preference based interactive evolutionary algorithm has been proposed. In the proposed algorithm, we start from an approximated nadir point and move progressively towards a desired Pareto optimal solution according to the preferences specified by the DM improving the values of all objectives simultaneously. In this, the DM does not need to compare too many solutions at a time. The ASF defined by the preference information is not only used to obtain better solutions, but also to define an effective termination criterion. At each iteration, we present to the DM just one solution at her/his desired distance from the Pareto front, and move progressively towards the Pareto front, allowing the DM to improve all objectives keeping the cognitive load set to the DM low. Thus, as better solutions are presented gradually, the DM is focussed in the solution process. An evolutionary algorithm has been used to find a near Pareto optimal solution and a mathematical programming technique has been used to guarantee convergence to (locally) Pareto optimal solution. All the populations in the evolutionary algorithm are stored in archive sets and used to show the DM some previous solutions if so desired. Also, we can save in computational cost in the PIE algorithm, if a new run of an evolutionary algorithm is needed. Then, we start from an initial population formed by already evaluated solutions that are located around the new reference point.

We have demonstrated the applicability of the PIE algorithm with a five objective optimization problem. The fact that the objective functions have real meanings enables a realistic interaction with the DM and permits to demonstrate how the PIE algorithm can be used in finding the most interesting solution in few iterations. Next we plan to apply the PIE algorithm with different DMs and problems. Additionally, we are conscious of the importance of a good graphical user-friendly interface in decision making, so a future research direction is to develop a user interface that supports the understandability and easiness of use of the PIE algorithm.

Acknowledgements

The authors would like to thank Professor Francisco Ruiz, Dr. Petri Eskelinen and Dr. Dmitry Podkopaev for their valuable comments.

References

1. Branke, J., Kaubler, T., Schmeck, H.: Guidence in evolutionary multi-objective optimization. Advances in Engineering Software 32(6), 499–507 (2001)
2. Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making Theory and Methodology. Elsevier Science Publishing Co., Inc., New York (1983)
3. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
4. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
5. Deb, K., Kumar, A.: Light beam search based multi-objective optimization using evolutionary algorithms. In: Proceedings of the Congress on Evolutionary Computation (CEC 2007), pp. 2125–2132. IEEE Press, Los Alamitos (2007)
6. Deb, K., Miettinen, K., Chaudhuri, S.: 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)

7. Deb, K., Saxena, D.K.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: Proceedings of the Congress on Evolutionary Computation (CEC 2006), pp. 3353–3360. IEEE Press, Los Alamitos (2006)
8. Deb, K., Sinha, A., Korhonen, P.J., Wallenius, J.: An interactive evolutionary multi-objective optimization method based on progressively approximated value functions. Tech. Rep. 2009005, KanGAL (2009)
9. Deb, K., Sundar, J., Rao, U.B., Chaudhuri, S.: Reference point based multi-objective optimization using evolutionary algorithms. International Journal of Computational Intelligence Research 2(3), 273–286 (2006)
10. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of the Congress on Evolutionary Computation (CEC 2002), vol. 1, pp. 825–830. IEEE Press, Los Alamitos (2002)
11. Hakanen, J., Miettinen, K., Mäkelä, M.M.: Using genetic algorithms in multiobjective process optimization. In: Bugeda, G., et al. (eds.) Proceedings of the Congress on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems (EUROGEN 2003), CD-Proceedings, CIMNE, Barcelona (2003)
12. Imai, A., Sasaki, K., Nishimura, E., Papadimitriou, S.: Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. European Journal of Operational Research 171(2), 373–389 (2006)
13. Knowles, J., Corne, D.: Quantifying the effects of objective space dimension in evolutionary multiobjective optimization. In: Obayashi, S., et al. (eds.) EMO 2007. LNCS, vol. 4403, pp. 757–771. Springer, Heidelberg (2007)
14. Luque, M., Miettinen, K., Eskelinen, P., Ruiz, F.: Incorporating preference information in interactive reference point methods for multiobjective optimization. Omega 37(2), 450–462 (2009)
15. Luque, M., Ruiz, F., Miettinen, K.: Global formulation for interactive multiobjective optimization. OR Spectrum 33(1), 27–48 (2011)
16. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
17. Miettinen, K.: Using interactive multiobjective optimization in continuous casting of steel. Materials and Manufacturing Processes 22, 585–593 (2007)
18. Miettinen, K., Eskelinen, P., Ruiz, F., Luque, M.: NAUTILUS method: An interactive technique in multiobjective optimization based on the nadir point. European Journal of Operational Research 206(2), 426–434 (2010)
19. Miettinen, K., Lotov, A.V., Kamenev, G.K., Berezkin, V.E.: Integration of two multiobjective optimization methods for nonlinear problems. Optimization Methods and Software 18(1), 63–80 (2003)
20. Ruiz, F., Luque, M., Cabello, J.M.: A classification of the weighting schemes in reference point procedures for multiobjective programming. Journal of the Operations Research Society 60, 544–553 (2009)
21. 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)
22. Tversky, A., Kahneman, D.: The framing of decisions and the psychology of choice. Science 211, 453–458 (1981)
23. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making: Theory and Application, pp. 468–486. Springer, Hagen (1980)
24. Wierzbicki, A.P.: On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spektrum 8, 73–87 (1986)