Chapter 15
Multi-objective Optimization

Kalyanmoy Deb

15.1 Introduction

Multi-objective optimization is an integral part of optimization activities and has a tremendous practical importance, since almost all real-world optimization problems are ideally suited to be modeled using multiple conflicting objectives. The classical means of solving such problems were primarily focused on scalarizing multiple objectives into a single objective, whereas the evolutionary means have been to solve a multi-objective optimization problem as it is. In this chapter, we discuss the fundamental principles of multi-objective optimization, the differences between multi-objective optimization and single-objective optimization, and describe a few well-known classical and evolutionary algorithms for multi-objective optimization. Two application case studies reveal the importance of multi-objective optimization in practice. A number of research challenges are then highlighted. The chapter concludes by suggesting a few tricks of the trade and mentioning some key resources to the field of multi-objective optimization.

Many real-world search and optimization problems are naturally posed as non-linear programming problems having multiple conflicting objectives. Due to lack of suitable solution techniques, such problems were artificially converted into a single-objective problem and solved. The difficulty arose because such problems give rise to a set of trade-off optimal solutions (known as *Pareto-optimal* solutions), instead of a single optimum solution. It then becomes important to find not just one

K. Deb (✉)
Koenig Endowed Chair Professor, Department of Electrical and Computer Engineering, Michigan State University, East Lansing, 428 S. Shaw Lane, 2120 EB, MI 48824, USA

Professor, Department of Computer Science, Michigan State University, East Lansing, MI, USA

Department of Mechanical Engineering, Michigan State University, East Lansing, MI, USA
e-mail: kdeb@egr.msu.edu

Pareto-optimal solution, but as many of them as possible. This is because any two such solutions constitute a trade-off between the objectives, and users will be in a better position to make a choice when such trade-off solutions are unveiled.

Classical methods use a different philosophy in solving these problems, mainly because of a lack of a suitable optimization methodology to find multiple optimal solutions efficiently. They usually require repetitive applications of an algorithm to find multiple Pareto-optimal solutions and on some occasions such applications do not even guarantee the finding of any Pareto-optimal solutions. In contrast, the population approach of evolutionary algorithms (EAs) is an efficient way to find multiple Pareto-optimal solutions simultaneously in a single simulation run. This aspect has made research and applications in evolutionary multi-objective optimization (EMO) popular over the past one-and-half decades. The interested reader may explore current research issues and other important studies in various texts (Deb 2001; Coello et al. 2002; Goh and Tan 2009; Bagchi 1999), conference proceedings (Zitzler et al. 2001a; Fonseca et al. 2003; Coello et al. 2005; Obayashi et al. 2007; Ehrgott et al. 2009; Takahashi et al. 2011) and numerous research papers (archived and maintained in Coello 2003).

In this tutorial, we discuss the fundamental differences between single- and multi-objective optimization tasks. The conditions for optimality in a multi-objective optimization problem are described and a number of state-of-the-art multi-objective optimization techniques, including one evolutionary method are presented. To demonstrate that the evolutionary multi-objective methods are capable and ready for solving real-world problems, we present a couple of interesting case studies. Finally, a number of important research topics in the area of EMO are discussed.

A multi-objective optimization problem (MOOP) deals with more than one objective function. In most practical decision-making problems, multiple objectives or multiple criteria are evident. Because of a lack of suitable solution methodologies, a MOOP has been mostly cast and solved as a single-objective optimization problem in the past. However, there exist a number of fundamental differences between the working principles of single- and multi-objective optimization algorithms because of which the solution of a MOOP must be attempted using a multi-objective optimization technique. In a single-objective optimization problem, the task is to find one solution (except in some specific multi-modal optimization problems, where multiple optimal solutions are sought) which optimizes the sole objective function. Extending the idea to multi-objective optimization, it may be wrongly assumed that the task in a multi-objective optimization is to find an optimal solution corresponding to each objective function. Certainly, multi-objective optimization is much more than this simple idea. We describe the concept of multi-objective optimization by using an example problem.

Let us consider the decision-making involved in buying an automobile car. Cars are available at prices ranging from a few thousand to few hundred thousand dollars. Let us take two extreme hypothetical cars, i.e. one costing about 10,000 dollars (solution 1) and another costing about a 100,000 dollars (solution 2), as shown in Fig. 15.1. If the cost is the only objective of this decision-making process, the optimal choice is solution 1. If this were the only objective to all buyers, we would have

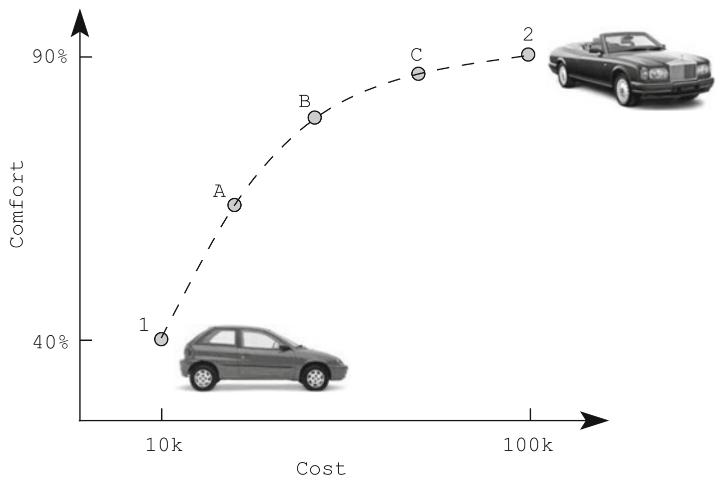

Fig. 15.1 Hypothetical trade-off solutions are illustrated for a car-buying decision-making problem

seen only one type of car (solution 1) on the road and no car manufacturer would have produced any expensive cars. Fortunately, this decision-making process is not a single-objective one. Barring some exceptions, it is expected that an inexpensive car is likely to be less comfortable. The figure indicates that the cheapest car has a hypothetical comfort level of 40 %. To rich buyers for whom comfort is the only objective of this decision-making, the choice is solution 2 (with a hypothetical maximum comfort level of 90 %, as shown in the figure). This so-called two-objective optimization problem need not be considered as the two independent optimization problems, the results of which are the two extreme solutions discussed above. Between these two extreme solutions, there exist many other solutions, where a trade-off between cost and comfort exists. A number of such solutions (solutions A, B and C) with differing costs and comfort levels are also shown in the figure. Thus, between any two such solutions, one is better in terms of one objective, but this betterment comes only from a sacrifice on the other objective. In this sense, all such trade-off solutions are optimal solutions to a MOOP. Often, such trade-off solutions provide a clear *front* on an objective space plotted with the objective values. This front is called the *Pareto-optimal* front and all such trade-off solutions are called Pareto-optimal solutions.

15.1.1 How Is It Different from Single-Objective Optimization?

It is clear from the above description that there exist a number of differences between single- and multi-objective optimization tasks. The latter have the following properties:

- Cardinality of the optimal set is usually more than one,
- There are two distinct goals of optimization, instead of one, and
- They possess two different search spaces.

We discuss each of the above properties in the following paragraphs.

First of all, we have seen from the above car-buying example that a multi-objective optimization with conflicting objectives results in a number of Pareto-optimal solutions, unlike the usual notion of only one optimal solution associated with a single-objective optimization task. However, there exist some single-objective optimization problems which also contain multiple optimal solutions (of equal or unequal importance). In some sense, multi-objective optimization is similar to that in such *multi-modal optimization* tasks. However, in principle, there is a difference, which we would like to highlight here. In most MOOPs, the Pareto-optimal solutions have certain similarities in their decision variables (Deb 2003). On the other hand, between one local or global optimal solution and another in a multi-modal optimization problem, there may not exist any such similarity. For a number of engineering case studies (Deb 2003), an analysis of the obtained trade-off solutions revealed the following properties:

- Among all Pareto-optimal solutions, some decision variables take identical values. Such a property of the decision variables means that the solution is an optimum solution.
- Other decision variables take different values causing the solutions to have a trade-off in their objective values.

Secondly, unlike the sole goal of finding the optimum in a single-objective optimization, here there are two distinct goals:

- Convergence to the Pareto-optimal solutions and
- Maintenance of a set of maximally spread Pareto-optimal solutions.

In a sense, these goals are independent of each other. An optimization algorithm must have specific properties for achieving each of the goals.

One other difference between single-objective and multi-objective optimization is that in multi-objective optimization the objective functions constitute a multi-dimensional space, in addition to the usual decision variable space common to all optimization problems. This additional space is called the *objective space*, $\mathcal{Z}$. For each solution $\mathbf{x}$ in the, there exists a point in the objective space, denoted by $\mathbf{f}(\mathbf{x}) = \mathbf{z} = (z_1, z_2, \ldots, z_M)^T$. The mapping takes place between an n-dimensional solution vector and an M-dimensional objective vector. Figure 15.2 illustrates these two spaces and a mapping between them. Although the search process of an algorithm takes place on the decision variables space, many interesting algorithms (particularly MOEAs) use the objective space information in their search operators. However, the presence of two different spaces introduces a number of interesting flexibilities in designing a search algorithm for multi-objective optimization.

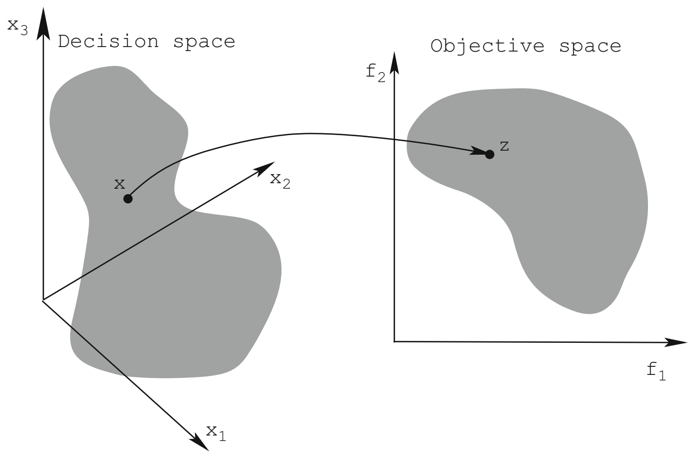

Fig. 15.2 Representation of the decision variable space and the corresponding objective space

15.2 Two Approaches to Multi-objective Optimization

Although the fundamental difference between single- and multiple-objective optimization lies in the cardinality in the optimal set, from a practical standpoint a user needs only one solution, no matter whether the associated optimization problem is single-objective or multi-objective. In the case of multi-objective optimization, the user is now in a dilemma. Which of these optimal solutions must one choose? Let us try to answer this question for the case of the car-buying problem. Knowing the number of solutions that exist in the market with different trade-offs between cost and comfort, which car does one buy? This is not an easy question to answer. It involves many other considerations, such as the total finance available to buy the car, distance to be driven each day, number of passengers riding in the car, fuel consumption and cost, depreciation value, road conditions where the car is to be mostly driven, physical health of the passengers, social status and many other factors. Often, such higher-level information is non-technical, qualitative and experience-driven. However, if a set of trade-off solutions are already worked out or available, one can evaluate the pros and cons of each of these solutions based on all such non-technical and qualitative, yet still important, considerations and compare them to make a choice. Thus, in a multi-objective optimization, ideally the effort must be in finding the set of trade-off optimal solutions by considering all objectives to be important. After a set of such trade-off solutions are found, a user can then use higher-level qualitative considerations to make a choice. Therefore, we suggest the following principle for an *ideal multi-objective optimization procedure*:

Step 1 Find multiple trade-off optimal solutions with a wide range of values for objectives.

Step 2 Choose one of the obtained solutions using higher-level information.

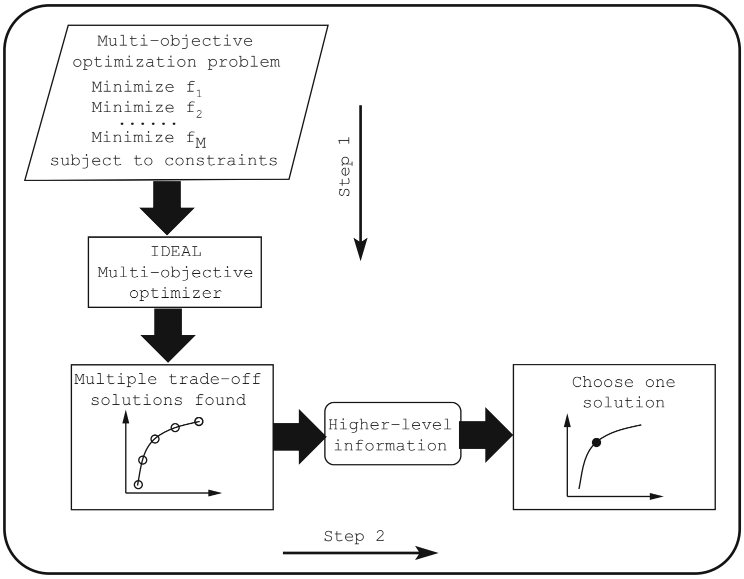

Fig. 15.3 Schematic of an ideal multi-objective optimization procedure

Figure 15.3 shows schematically the principles in an ideal multi-objective optimization procedure. In Step 1 (vertically downwards), multiple trade-off solutions are found. Thereafter, in Step 2 (horizontally, towards the right), higher-level information is used to choose one of the trade-off solutions. With this procedure in mind, it is easy to realize that single-objective optimization is a degenerate case of multi-objective optimization. In the case of single-objective optimization with only one global optimal solution, Step 1 will find only one solution, thereby not requiring us to proceed to Step 2. In the case of single-objective optimization with multiple global optima, both steps are necessary to first find all or many of the global optima and then to choose one from them by using the higher-level information about the problem.

If thought of carefully, each trade-off solution corresponds to a specific order of importance of the objectives. It is clear from Fig. 15.1 that solution A assigns more importance to cost than to comfort. On the other hand, solution C assigns more importance to comfort than to cost. Thus, if such a relative preference factor among the objectives is known for a specific problem, there is no need to follow the above principle for solving a MOOP. A simple method would be to form a composite objective function as the weighted sum of the objectives, where a weight for an objective is proportional to the preference factor assigned to that particular objective. This method of scalarizing an objective vector into a single composite objective function converts the MOOP into a single-objective optimization problem. When such a composite objective function is optimized, in most cases it is possible to obtain one particular trade-off solution. This procedure of handling MOOPs is much simpler, though still being more subjective than the above ideal procedure. We call

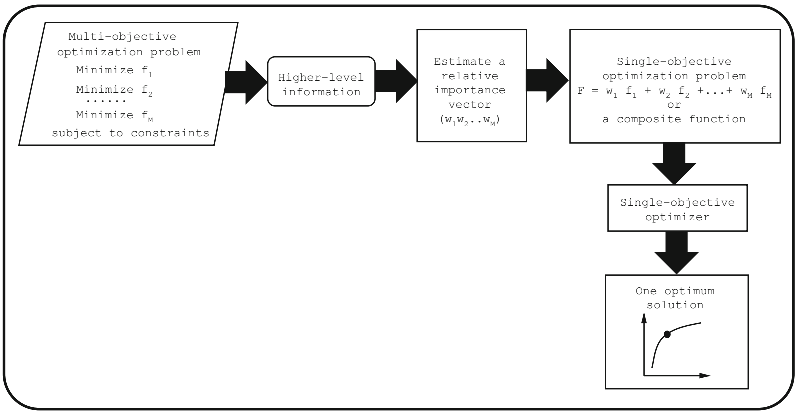
Multi-objective optimization problem
Minimize f_1
Minimize f_2
......
Minimize f_M
subject to constraints
Higher-level information
Estimate a relative importance vector $(w_1 w_2 .. w_M)$
Single-objective optimization problem
$F = w_1 f_1 + w_2 f_2 + ... + w_M f_M$
or
a composite function
Single-objective optimizer
One optimum solution

Fig. 15.4 Schematic of a preference-based multi-objective optimization procedure

this procedure a *preference-based* multi-objective optimization. A schematic of this procedure is shown in Fig. 15.4. Based on the higher-level information, a preference vector **w** is first chosen. Thereafter, the preference vector is used to construct the composite function, which is then optimized to find a single trade-off optimal solution by a single-objective optimization algorithm. Although not often practiced, the procedure can be used to find multiple trade-off solutions by using a different preference vector and repeating the above procedure.

It is important to appreciate that the trade-off solution obtained by using the preference-based strategy is largely sensitive to the relative preference vector used in forming the composite function. A change in this preference vector will result in a (hopefully) different trade-off solution. Besides this difficulty, it is intuitive to realize that finding a relative preference vector itself is highly subjective and not straightforward. This requires an analysis of the non-technical, qualitative and experience-driven information to find a quantitative relative preference vector. Without any knowledge of the likely trade-off solutions, this is an even more difficult task. Classical multi-objective optimization methods which convert multiple objectives into a single objective by using a relative preference vector of objectives work according to this preference-based strategy. Unless a reliable and accurate preference vector is available, the optimal solution obtained by such methods is highly subjective to the particular user.

The ideal multi-objective optimization procedure suggested earlier is less subjective. In Step 1, a user does not need any relative preference vector information. The task there is to find as many different trade-off solutions as possible. Once a well-distributed set of trade-off solutions is found, Step 2 then requires certain problem information in order to choose one solution. It is important to mention that in Step 2, the problem information is used to evaluate and compare each of the obtained trade-off solutions. In the ideal approach, the problem information is not used to search for a *new* solution; instead, it is used to choose one solution from a set of already obtained trade-off solutions. Thus, there is a fundamental difference in using the problem information in both approaches. In the preference-based approach, a relative preference vector needs to be supplied without any knowledge of the possible consequences. However, in the proposed ideal approach, the problem information is used to choose one solution from the obtained set of trade-off solutions. We argue that the ideal approach in this matter is more methodical, more practical, and less subjective. At the same time, we highlight the fact that if a reliable relative preference vector is available to a problem, there is no reason to find other trade-off solutions. In such a case, a preference-based approach would be adequate.

In the next section, we make the above qualitative idea of multi-objective optimization more quantitative.

15.3 Non-dominated Solutions and Pareto-Optimal Solutions

Most multi-objective optimization algorithms use the concept of dominance in their search. Here, we define the concept of dominance and related terms and present a number of techniques for identifying dominated solutions in a finite population of solutions.

15.3.1 Special Solutions

We first define some special solutions which are often used in multi-objective optimization algorithms.

15.3.1.1 Ideal Objective Vector

For each of the M conflicting objectives, there exists one different optimal solution. An objective vector constructed with these individual optimal objective values constitutes the ideal objective vector.

Definition 15.1. The mth component of the ideal objective vector $\mathbf{z}^*$ is the constrained minimum solution of the following problem:

$$\left.\begin{array}{l}\text{Minimize } f_m(\mathbf{x}) \\ \text{subject to } \mathbf{x} \in \mathcal{S}\end{array}\right\}. \tag{15.1}$$

Thus, if the minimum solution for the mth objective function is the decision vector $\mathbf{x}^{*(m)}$ with function value f_m^*, the ideal vector is as follows:

$$\mathbf{z}^* = \mathbf{f}^* = (f_1^*, f_2^*, \ldots, f_M^*)^T.$$

In general, the ideal objective vector ($\mathbf{z}^*$) corresponds to a non-existent solution (Fig. 15.5). This is because the minimum solution of Eq. (15.1) for each objective function need not be the same solution. The only way an ideal objective vector

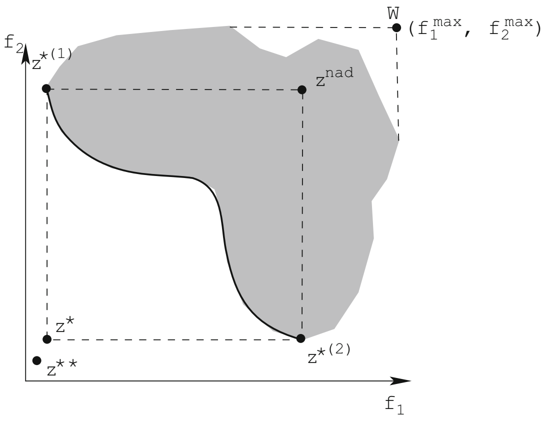

Fig. 15.5 The ideal, utopian and nadir objective vectors

corresponds to a feasible solution is when the minimal solutions to all objective functions are identical. In this case, the objectives are not conflicting to each other and the minimum solution to any objective function would be the only optimal solution to the MOOP. Although the ideal objective vector is usually non-existent, it is also clear from Fig. 15.5 that solutions closer to the ideal objective vector are better. Moreover, many algorithms require the knowledge of the lower bound on each objective function to normalize objective values in a common range.

15.3.1.2 Utopian Objective Vector

The ideal objective vector denotes an array of the lower bound of all objective functions. This means that for every objective function there exists at least one solution in the feasible search space sharing an identical value with the corresponding element in the ideal solution. Some algorithms may require a solution which has an objective value strictly better than (and not equal to) that of any solution in the search space. For this purpose, the utopian objective vector is defined as follows.

Definition 15.2. A utopian objective vector $\mathbf{z}^{**}$ has each of its components marginally smaller than that of the ideal objective vector, or $\mathbf{z}_i^{**} = \mathbf{z}_i^{*} - \varepsilon_i$ with $\varepsilon_i > 0$ for all $i = 1, 2, \ldots, M$.

Figure 15.5 shows a utopian objective vector. Like the ideal objective vector, the utopian objective vector also represents a non-existent solution.

15.3.1.3 Nadir Objective Vector

Unlike the ideal objective vector which represents the lower bound of each objective in the entire feasible search space, the nadir objective vector $\mathbf{z}^{\mathrm{nad}}$ represents the upper bound of each objective in the entire Pareto-optimal set, and not in the entire search space. A nadir objective vector must not be confused with a vector of objectives (marked as "W" in Fig. 15.5) found by using the worst feasible function values $f_i^{\max}$ in the entire search space. The nadir objective vector may represent an existent or a non-existent solution, depending on the convexity and continuity of the Pareto-optimal set. In order to normalize each objective in the entire range of the Pareto-optimal region, the knowledge of nadir and ideal objective vectors can be used as follows:

$$f_i^{\text{norm}} = \frac{f_i - z_i^*}{z_i^{\text{nad}} - z_i^*}. \tag{15.2}$$

15.3.2 Concept of Domination

Most multi-objective optimization algorithms use the concept of domination. In these algorithms, two solutions are compared on the basis of whether one dominates the other or not. We will describe the concept of domination in the following paragraph.

We assume that there are M objective functions. In order to cover both minimization and maximization of objective functions, we use the operator $\triangleleft$ between two solutions i and j as $i \triangleleft j$ to denote that solution i is better than solution j on a particular objective. Similarly, $i \triangleright j$ for a particular objective implies that solution i is worse than solution j on this objective. For example, if an objective function is to be minimized, the operator $\triangleleft$ would mean the "$<$" operator, whereas if the objective function is to be maximized, the operator $\triangleleft$ would mean the "$>$" operator. The following definition covers mixed problems with minimization of some objective functions and maximization of the rest of them.

Definition 15.3. A solution $\mathbf{x}^{(1)}$ is said to dominate the other solution $\mathbf{x}^{(2)}$ if both conditions 1 and 2 are true:

1. The solution $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in all objectives, or $f_j(\mathbf{x}^{(1)}) \not\triangleright f_j(\mathbf{x}^{(2)})$ for all $j = 1, 2, \ldots, M$.
2. The solution $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective, or $f_{\bar{j}}(\mathbf{x}^{(1)}) \triangleleft f_{\bar{j}}(\mathbf{x}^{(2)})$ for at least one $\bar{j} \in \{1, 2, \ldots, M\}$.

If either of these conditions is violated, the solution $\mathbf{x}^{(1)}$ does not dominate the solution $\mathbf{x}^{(2)}$. If $\mathbf{x}^{(1)}$ dominates the solution $\mathbf{x}^{(2)}$ (or mathematically $\mathbf{x}^{(1)} \preceq \mathbf{x}^{(2)}$), it is also customary to write any of the following:

- $\mathbf{x}^{(2)}$ is dominated by $\mathbf{x}^{(1)}$
- $\mathbf{x}^{(1)}$ is non-dominated by $\mathbf{x}^{(2)}$ or
- $\mathbf{x}^{(1)}$ is non-inferior to $\mathbf{x}^{(2)}$.

Let us consider a two-objective optimization problem with five different solutions shown in the objective space, as illustrated in Fig. 15.6a. Let us also assume

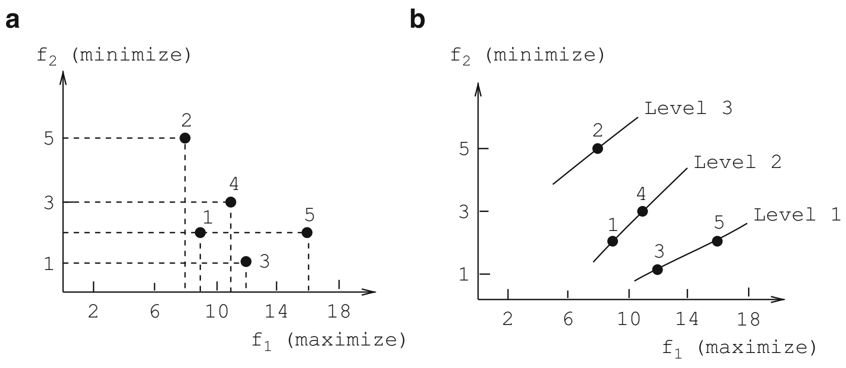

Fig. 15.6 A set of five solutions and the corresponding non-dominated fronts

that the objective function 1 needs to be maximized while the objective function 2 needs to be minimized. Five solutions with different objective function values are shown in this figure. Since both objective functions are of importance to us, it is usually difficult to find one solution which is best with respect to both objectives. However, we can use the above definition of domination to decide which solution is better among any two given solutions in terms of both objectives. For example, if solutions 1 and 2 are to be compared, we observe that solution 1 is better than solution 2 in objective function 1 and solution 1 is also better than solution 2 in objective function 2. Thus, both of the above conditions for domination are satisfied and we may write that solution 1 dominates solution 2. We take another instance of comparing solutions 1 and 5. Here, solution 5 is better than solution 1 in the first objective and solution 5 is no worse (in fact, they are equal) than solution 1 in the second objective. Thus, both the above conditions for domination are also satisfied and we may write that solution 5 dominates solution 1.

It is intuitive that if a solution $\mathbf{x}^{(1)}$ dominates another solution $\mathbf{x}^{(2)}$, the solution $\mathbf{x}^{(1)}$ is better than $\mathbf{x}^{(2)}$ in the parlance of multi-objective optimization. Since the concept of domination allows a way to compare solutions with multiple objectives, most multi-objective optimization methods use this domination concept to search for non-dominated solutions.

15.3.3 Properties of Dominance Relation

Definition 15.3 defines the dominance relation between any two solutions. There are three possibilities that can be the outcome of the dominance check between two solutions 1 and 2. That is (i) solution 1 dominates solution 2, (ii) solution 1 gets dominated by solution 2, or (iii) solutions 1 and 2 do not dominate each other. Let us now discuss the different binary relation properties (Cormen et al. 1990) of the dominance operator.

- *Reflexive*. The dominance relation is *not reflexive*, since any solution p does not dominate itself according to Definition 15.3. The second condition of dominance relation in Definition 15.3 does not allow this property to be satisfied.
- *Symmetric*. The dominance relation is also *not symmetric*, because $p \preceq q$ does not imply $q \preceq p$. In fact, the opposite is true. That is, if p dominates q, then q does not dominate p. Thus, the dominance relation is *asymmetric*.
- *Antisymmetric*. Since the dominance relation is not symmetric, it cannot be antisymmetric as well.
- *Transitive*. The dominance relation is *transitive*. This is because if $p \preceq q$ and $q \preceq r$, then $p \preceq r$.

There is another interesting property that the dominance relation possesses. If solution p does not dominate solution q, this does not imply that q dominates p.

In order for a binary relation to qualify as an ordering relation, it must be at least transitive (Chankong and Haimes 1983). Thus, the dominance relation qualifies as an ordering relation. Since the dominance relation is not reflexive, it is a *strict partial order*. In general, if a relation is reflexive, antisymmetric and transitive, it is loosely called a *partial order* and a set on which a partial order is defined is called a *partially ordered set*. However, it is important to note that the dominance relation is not reflexive and is not antisymmetric. Thus, the dominance relation is not a partial-order relation in its general sense. The dominance relation is only a strict partial-order relation.

15.3.4 Pareto Optimality

Continuing with the comparisons in the previous section, let us compare solutions 3 and 5 in Fig. 15.6, because this comparison reveals an interesting aspect. We observe that solution 5 is better than solution 3 in the first objective, while solution 5 is worse than solution 3 in the second objective. Thus, the first condition is not satisfied for both of these solutions. This simply suggests that we cannot conclude that solution 5 dominates solution 3, nor can we say that solution 3 dominates solution 5. When this happens, it is customary to say that solutions 3 and 5 are non-dominated with respect to each other. When both objectives are important, it cannot be said which of the two solutions 3 and 5 is better.

For a given finite set of solutions, we can perform all possible pair-wise comparisons and find which solution dominates which and which solutions are non-dominated with respect to each other. At the end, we expect to have a set of solutions, any two of which do not dominate each other. This set also has another property. For any solution outside of this set, we can always find a solution in this set which will dominate the former. Thus, this particular set has a property of dominating all other solutions which do not belong to this set. In simple terms, this means that the solutions of this set are better compared to the rest of the solutions. This set is given a special name. It is called the *non-dominated set* for the given set of

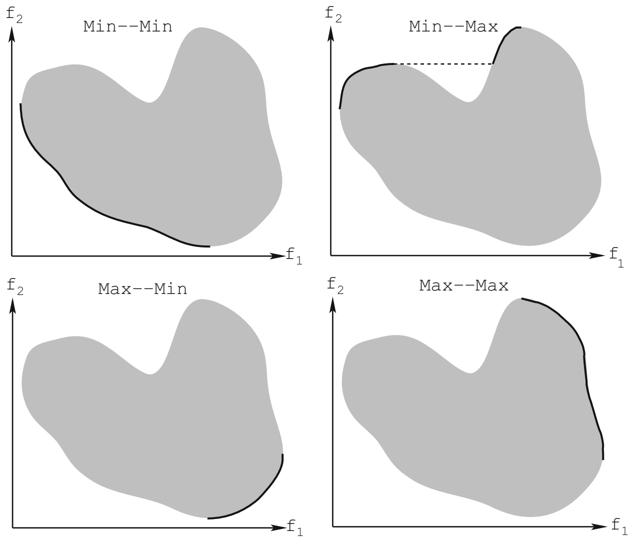

Fig. 15.7 Pareto-optimal solutions are marked with continuous curves for four combinations of two types of objectives

solutions. In the example problem, solutions 3 and 5 constitute the non-dominated set of the given set of five solutions. Thus, we define a set of non-dominated solutions as follows.

Definition 15.4 (Non-dominated set). Among a set of solutions P, the non-dominated set of solutions P' are those that are not dominated by any member of the set P.

When the set P is the entire search space, or $P = \mathcal{S}$, the resulting non-dominated set P' is called the *Pareto-optimal set*. Figure 15.7 marks the Pareto-optimal set with continuous curves for four different scenarios with two objectives. Each objective can be minimized or maximized. In the top-left panel, the task is to minimize both objectives f_1 and f_2. The solid curve marks the Pareto-optimal solution set. If f_1 is to be minimized and f_2 is to be maximized for a problem having the same search space, the resulting Pareto-optimal set is different and is shown in the top-right panel. Here, the Pareto-optimal set is a union of two disconnected Pareto-optimal regions. Similarly, the Pareto-optimal sets for two other cases—(maximizing f_1, minimizing f_2) and (maximizing f_1, maximizing f_2)—are shown in the bottom-left and bottom-right panels, respectively. In any case, the Pareto-optimal set always consists of solutions from a particular edge of the feasible search region.

It is important to note that an MOEA can be easily used to handle all of the above cases by simply using the domination definition. However, to avoid any confusion, most applications use the duality principle (Deb 1995) to convert a maximization problem into a minimization problem and treat every problem as a combination of

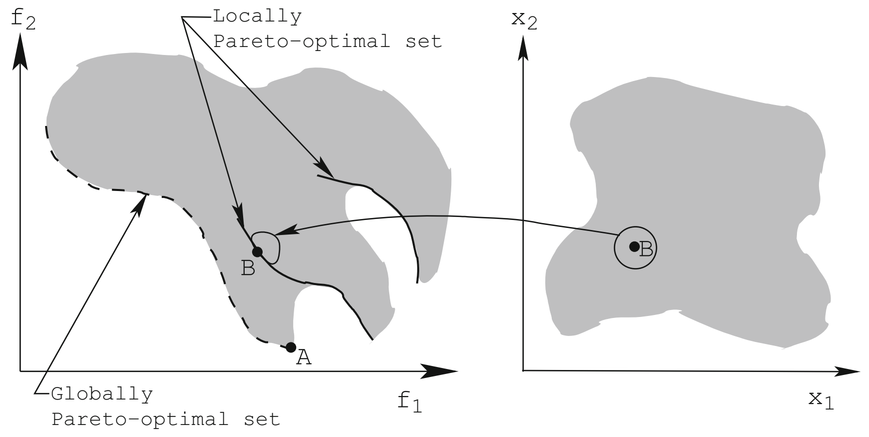

Fig. 15.8 Locally and globally Pareto-optimal solutions

minimizing all objectives. Like global and local optimal solutions in the case of single-objective optimization, there could be global and local Pareto-optimal sets in multi-objective optimization.

Definition 15.5 (Globally Pareto-optimal set). The non-dominated set of the entire feasible search space $\mathcal{S}$ is the globally Pareto-optimal set.

Definition 15.6. If for every member $\mathbf{x}$ in a set $\underline{P}$ there exists no solution $\mathbf{y}$ (in the neighborhood of $\mathbf{x}$ such that $\|\mathbf{y}-\mathbf{x}\|_\infty \leq \varepsilon$, where ε is a small positive number) dominating any member of the set $\underline{P}$, then solutions belonging to the set $\underline{P}$ constitute a locally Pareto-optimal set.

Figure 15.8 shows two locally Pareto-optimal sets (marked by continuous curves). When any solution (say "B") in this set is perturbed locally in the decision variable space, no solution can be found dominating any member of the set. It is interesting to note that for continuous search space problems, the locally Pareto-optimal solutions need not be continuous in the decision variable space and the above definition will still hold good. Zitzler (1999) added a neighborhood constraint on the objective space in the above definition to make it more generic. By the above definition, it is also true that a globally Pareto-optimal set is also a locally Pareto-optimal set.

15.3.5 Procedure for Finding Non-dominated Solutions

Finding the non-dominated set of solutions from a given set of solutions is similar in principle to finding the minimum of a set of real numbers. In the latter case, when two numbers are compared to identify the smaller number, a '$<$' relation operation is used. In the case of finding the non-dominated set, the dominance relation $\preceq$ can be used to identify the better of two given solutions. Here, we discuss one simple procedure for finding the non-dominated set (we call here the best non-dominated front). Many MOEAs require to find the best non-dominated solutions of a population and some MOEAs require to sort a population according to different non-domination levels. We present one algorithm for each of the tasks.

15.3.5.1 Finding the Best Non-dominated Front

In this approach, every solution from the population is checked with a partially filled population for domination. To start with, the first solution from the population is kept in an empty set P'. Thereafter, each solution i (the second solution onwards) is compared with all members of the set P', one by one. If the solution i dominates any member of P', then that solution is removed from P'. In this way non-members of the non-dominated solutions get deleted from P'. Otherwise, if solution i is dominated by any member of P', the solution i is ignored. If solution i is not dominated by any member of P', it is entered in P'. This is how the set P' grows with non-dominated solutions. When all solutions of the population are checked, the remaining members of P' constitute the non-dominated set.

Identifying the non-dominated set

Step 1 Initialize $P' = \{1\}$. Set solution counter $i = 2$.

Step 2 Set $j = 1$.

Step 3 Compare solution i with j from P' for domination.

Step 4 If i dominates j, then delete the jth member from P' or else update $P' = P' \backslash \{P'^{(j)}\}$. If $j < |P'|$, increment j by one and then go to Step 3. Otherwise, go to Step 5. Alternatively, if the jth member of P' dominates i, increment i by one and then go to Step 2.

Step 5 Insert i in P' or update $P' = P' \cup \{i\}$. If $i < N$, increment i by one and go to Step 2. Otherwise, stop and declare P' as the non-dominated set.

Here, we observe that the second element of the population is compared with only one solution P', the third solution with at most two solutions of P', and so on. This requires a maximum of $1 + 2 + \cdots + (N - 1)$ or $N(N - 1)/2$ domination checks. This computation is also $O(MN^2)$. It is interesting to note that the size of P' may not always increase (dominated solutions will get deleted from P') and not every solution in the population may be required to be checked with all solutions in the current P' set (the solution may get dominated by a solution of P'). Thus, the actual computational complexity may be smaller than the above estimate.

Another study (Kung et al. 1975) suggested a binary-search-like algorithm for finding the best non-dominated front with a complexity $O\left(N(\log N)^{M-2}\right)$ for $M \geq 4$ and $O(N \log N)$ for $M = 2$ and 3.

15.3.5.2 A Non-dominated Sorting Procedure

Using the above procedure, each front can be identified with at most $O(MN^2)$ computations. In certain scenarios, this procedure may demand more than $O(MN^2)$ computational effort for the overall non-dominated sorting of a population. Here, we suggest a completely different procedure which uses a better bookkeeping strategy requiring $O(MN^2)$ overall computational complexity.

First, for each solution we calculate two entities: (i) *domination count* n_i, the number of solutions which dominate the solution i, and (ii) S_i, a set of solutions which the solution i dominates. This requires $O(MN^2)$ comparisons. At the end of this procedure, all solutions in the first non-dominated front will have their domination count as zero. Now, for each of these solutions (each solution i with $n_i = 0$), we visit each member (j) of its set S_i and reduce its domination count by one. In doing so, if for any member j the domination count becomes zero, we put it in a separate list P'. After such modifications on S_i are performed for each i with $n_i = 0$, all solutions of P' would belong to the second non-dominated front. The above procedure can be continued with each member of P' and the third non-dominated front can be identified. This process continues until all solutions are classified.

An $O(MN^2)$ non-dominated sorting algorithm

```
Step 1  For each $i \in P$, $n_i = 0$ and initialize $S_i = \emptyset$. For all $j \neq i$ and $j \in P$,
   perform Step 2 and then proceed to Step 3.
Step 2  If $i \preceq j$, update $S_p = S_p \cup \{j\}$. Otherwise, if $j \preceq i$, set $n_i = n_i + 1$.
Step 3  If $n_i = 0$, keep $i$ in the first non-dominated front $P_1$ (we called this set
   $P'$ in the above paragraph). Set a front counter $k = 1$.
Step 4  While $P_k \neq \emptyset$, perform the following steps.
Step 5  Initialize $Q = \emptyset$ for storing next non-dominated solutions. For each
   $i \in P_k$ and for each $j \in S_i$,
   Step 5a  Update $n_j = n_j - 1$.
   Step 5b  If $n_j = 0$, keep $j$ in $Q$, or perform $Q = Q \cup \{j\}$.
Step 6  Set $k = k + 1$ and $P_k = Q$. Go to Step 4.
```

Steps 1–3 find the solutions in the first non-dominated front and require $O(MN^2)$ computational complexity. Steps 4–6 repeatedly find higher fronts and require at most $O(N^2)$ comparisons, as argued below. For each solution i in the second- or higher level of non-domination, the domination count n_i can be at most $N-1$. Thus, each solution i will be visited at most $N-1$ times before its domination count becomes zero. At this point, the solution is assigned a particular non-domination level and will never be visited again. Since there are at most $N-1$ such solutions, the complexity of identifying second and more fronts is $O(N^2)$. Thus, the overall complexity of the procedure is $O(MN^2)$. It is important to note that although the time complexity has reduced to $O(MN^2)$, the storage requirement has increased to $O(N^2)$.

When the above procedure is applied to the five solutions of Fig. 15.6a, we obtain three non-dominated fronts as shown in Fig. 15.6b. From the dominance relations, the solutions 3 and 5 are the best, followed by solutions 1 and 4. Finally, solution 2 belongs to the worst non-dominated front. Thus, the ordering of solutions in terms of their non-domination level is as follows: ((3,5), (1,4), (2)). A study (Jensen 2003b) suggested a divided-and-conquer method to reduce the complexity of sorting to $O(N \log^{M-1} N)$.

15.4 Some Approaches to Multi-objective Optimization

In this section, we briefly mention two commonly used classical multi-objective optimization methods and thereafter present a commonly used EMO method.

15.4.1 Classical Method: Weighted-Sum Approach

The weighted-sum method, as the name suggests, scalarizes a set of objectives into a single objective by pre-multiplying each objective with a user-supplied weight. This method is the simplest approach and is probably the most widely used classical approach. If we are faced with the two objectives of minimizing the cost of a product and minimizing the amount of wasted material in the process of fabricating the product, one naturally thinks of minimizing a weighted sum of these two objectives. Although the idea is simple, it introduces a not-so-simple question. What values of the weights must one use? Of course, there is no unique answer to this question. The answer depends on the importance of each objective in the context of the problem and a scaling factor. The scaling effect can be avoided somewhat by normalizing the objective functions. After the objectives are normalized, a composite objective function $F(\mathbf{x})$ can be formed by summing the weighted normalized objectives and the problem is then converted to a single-objective optimization problem as follows:

$$\left.\begin{array}{lll} \text{Minimize } & F(\mathbf{x}) = \sum_{m=1}^{M} w_m f_m(\mathbf{x}) & \\ \text{subject to } & g_j(\mathbf{x}) \geq 0, & j = 1,2,\ldots,J \\ & h_k(\mathbf{x}) = 0, & k = 1,2,\ldots,K \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1,2,\ldots,n. \end{array}\right\} \tag{15.3}$$

Here, w_m ($\in [0,1]$) is the weight of the mth objective function. Since the minimum of the above problem does not change if all weights are multiplied by a constant, it is the usual practice to choose weights such that their sum is one, or $\sum_{m=1}^{M} w_m = 1$.

Mathematically oriented readers may find a number of interesting theorems regarding the relationship between the optimal solution of the above problem to the true Pareto-optimal solutions in classical texts (Chankong and Haimes 1983; Miettinen 1999; Ehrgott 2000).

Let us now illustrate how the weighted-sum approach can find Pareto-optimal solutions of the original problem. For simplicity, we consider the two-objective problem shown in Fig. 15.9. The feasible objective space and the corresponding

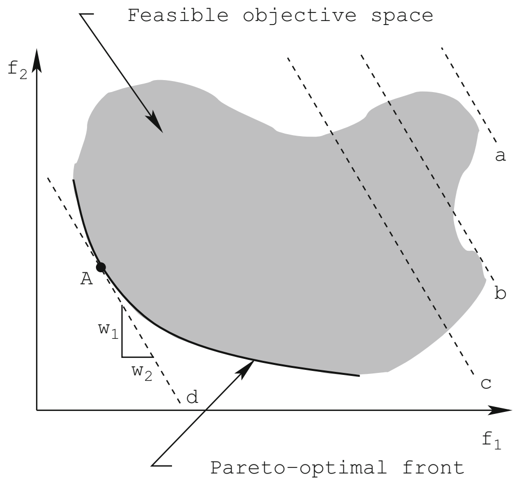

Fig. 15.9 Illustration of the weighted-sum approach on a convex Pareto-optimal front

Pareto-optimal solution set are shown. With two objectives, there are two weights w_1 and w_2, but only one is independent. Knowing any one, the other can be calculated by simple subtraction. It is clear from the figure that a choice of a weight vector corresponds to a pre-destined optimal solution on the Pareto-optimal front, as marked by the point A. By changing the weight vector, a different Pareto-optimal point can be obtained. However, there are a couple of difficulties with this approach:

1. A uniform choice of weight vectors does not necessarily find a uniform set of Pareto-optimal solutions on the Pareto-optimal front (Deb 2001).
2. The procedure cannot be used to find Pareto-optimal solutions which lie on the non-convex portion of the Pareto-optimal front.

The former issue makes it difficult for the weighted-sum approach to be applied reliably to any problem in order to find a good representative set of Pareto-optimal solutions. The latter issue arises due to the fact that a solution lying on the non-convex Pareto-optimal front can never be the optimal solution of the problem given in Eq. (15.3).

15.4.2 Classical Method: ε-Constraint Method

In order to alleviate the difficulties faced by the weighted-sum approach in solving problems having non-convex objective spaces, the ε-constraint method is used. Haimes et al. (1971) suggested reformulating the MOOP by just keeping one of the objectives and restricting the rest of the objectives within user-specified values. The modified problem is as follows:

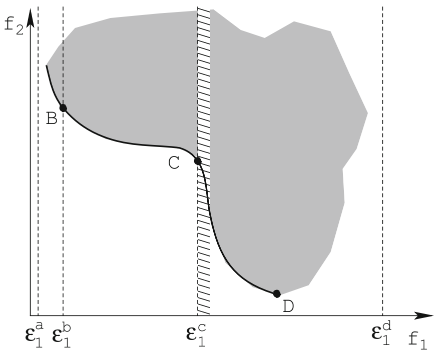

Fig. 15.10 The ε-constraint method

$$\left.\begin{array}{lll}\text{Minimize } f_\mu(\mathbf{x}) & & \\ \text{subject to } f_m(\mathbf{x}) \le \varepsilon_m, & m = 1,2,\ldots,M \text{ and } m \neq \mu & \\ \qquad g_j(\mathbf{x}) \ge 0, & j = 1,2,\ldots, & \\ \qquad h_k(\mathbf{x}) = 0, & k = 1,2,\ldots,K & \\ \qquad x_i^{(L)} \le x_i \le x_i^{(U)}, & i = 1,2,\ldots,n. & \end{array}\right\} \qquad (15.4)$$

In the above formulation, the parameter ε_m represents an upper bound of the value of f_m and need not necessarily mean a small value close to zero.

Let us say that we retain f_2 as an objective and treat f_1 as a constraint: $f_1(\mathbf{x}) \le \varepsilon_1$. Figure 15.10 shows four scenarios with different ε_1 values. Let us consider the third scenario with $\varepsilon_1 = \varepsilon_1^c$ first. The resulting problem with this constraint divides the original feasible objective space into two portions, $f_1 \le \varepsilon_1^c$ and $f_1 > \varepsilon_1^c$. The left portion becomes the feasible solution of the resulting problem stated in Eq. (15.4). Now, the task of the resulting problem is to find the solution which minimizes this feasible region. From Fig. 15.10, it is clear that the minimum solution is C. In this way, intermediate Pareto-optimal solutions can be obtained in the case of non-convex objective space problems by using the ε-constraint method.

One of the difficulties of this method is that the solution to the problem stated in Eq. (15.4) largely depends on the chosen $\boldsymbol{\varepsilon}$ vector. Let us refer to Fig. 15.10 again. Instead of choosing ε_1^c, if ε_1^a is chosen, there exists no feasible solution to the stated problem. Thus, no solution would be found. On the other hand, if ε_1^d is used, the entire search space is feasible. The resulting problem has the minimum at D. Moreover, as the number of objectives increases, there exist more elements in the $\boldsymbol{\varepsilon}$ vector, thereby requiring more information from the user.

15.4.3 Evolutionary Multi-objective Optimization (EMO) Method

Over the years, a number of multi-objective EAs () emphasizing non-dominated solutions in a EA population have been suggested. In this section, we shall describe one state-of-the-art algorithm popularly used in EMO studies.

15.4.3.1 Elitist Non-dominated Sorting GA (NSGA-II)

The non-dominated sorting GA or NSGA-II procedure (Deb et al. 2002) for finding multiple Pareto-optimal solutions in a MOOP has the following three features:

1. It uses an elitist principle,
2. It uses an explicit diversity preserving mechanism, and
3. It emphasizes the non-dominated solutions.

In NSGA-II, the offspring population Q_t is first created by using the parent population P_t and the usual genetic operators (Goldberg 1989). Thereafter, the two populations are combined to form R_t of size $2N$. Then, a non-dominated sorting is used to classify the entire population R_t. Once the non-dominated sorting is over, the new population is filled by solutions of different non-dominated fronts, one at a time. The filling starts with the best non-dominated front and continues with solutions of the second non-dominated front, followed by the third non-dominated front, and so on. Since the overall population size of R_t is $2N$, not all fronts may be accommodated in N slots available in the new population. All fronts which could not be accommodated are simply deleted. When the last allowed front is being considered, there may exist more solutions in the last front than the remaining slots in the new population. This scenario is illustrated in Fig. 15.11. Instead of arbitrarily discarding some members from the last acceptable front, the solutions which will make the diversity of the selected solutions the highest are chosen. The NSGA-II procedure is outlined in the following.

NSGA-II

Step 1 Combine parent and offspring populations and create $R_t = P_t \cup Q_t$. Perform a non-dominated sorting to R_t and identify different fronts: $\mathcal{F}_i$, $i = 1, 2, \ldots$, etc.

Step 2 Set new population $P_{t+1} = \emptyset$. Set a counter $i = 1$.
Until $|P_{t+1}| + |\mathcal{F}_i| < N$, perform $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$ and $i = i + 1$.

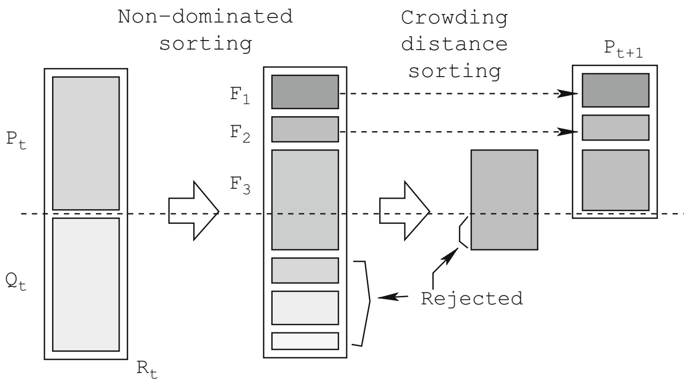

Fig. 15.11 Schematic of the NSGA-II procedure

Step 3 Perform the `Crowding-sort`$(\mathcal{F}_i, <_c)$ procedure and include the most widely spread $(N - |P_{t+1}|)$ solutions by using the crowding distance values in the sorted $\mathcal{F}_i$ to P_{t+1}.

Step 4 Create offspring population Q_{t+1} from P_{t+1} by using the crowded tournament selection, crossover and mutation operators.

In Step 3, the crowding-sorting of the solutions of front i (the last front which could not be accommodated fully) is performed by using a *crowding distance metric*, which we describe later. The population is arranged in descending order of magnitude of the crowding distance values. In Step 4, a crowding tournament selection operator, which also uses the crowding distance, is used.

The crowded comparison operator $(<_c)$ compares two solutions and returns the winner of the tournament. It assumes that every solution i has two attributes:

1. A non-domination rank r_i in the population,
2. A local (d_i) in the population.

The crowding distance d_i of a solution i is a measure of the normalized search space around i which is not occupied by any other solution in the population. Based on these two attributes, we can define the crowded tournament selection operator as follows.

Definition 15.7. *Crowded tournament selection operator.* A solution i wins a tournament with another solution j if any of the following conditions are true:

1. If solution i has a better rank, that is, $r_i < r_j$.
2. If they have the same rank but solution i has a better crowding distance than solution j, that is, $r_i = r_j$ and $d_i > d_j$.

The first condition makes sure that the chosen solution lies on a better non-dominated front. The second condition resolves the tie of both solutions being on the same non-dominated front by deciding on their crowded distance. The one residing in a less crowded area (with a larger crowding distance d_i) wins. The crowding distance d_i can be computed in various ways. However, in NSGA-II, we use a crowding distance metric, which requires $\mathrm{O}(MN\log N)$ computations.

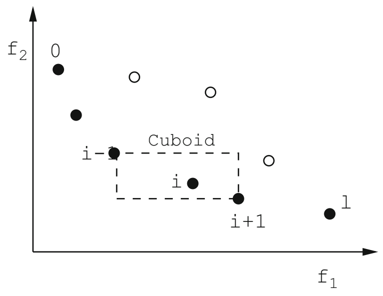

Fig. 15.12 The crowding distance calculation

To get an estimate of the density of solutions surrounding a particular solution i in the population, we take the average distance of two solutions on either side of solution i along each of the objectives. This quantity d_i serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices (we call this the *crowding distance*). In Fig. 15.12, the crowding distance of the ith solution in its front (marked with filled circles) is the average side-length of the cuboid (shown by a dashed box). The following algorithm is used to calculate the crowding distance of each point in the set $\mathcal{F}$.

Crowding distance assignment procedure: `Crowding-sort`$(\mathcal{F}, <_c)$

Step C1 Call the number of solutions in $\mathcal{F}$ as $l = |\mathcal{F}|$. For each i in the set, first assign $d_i = 0$.

Step C2 For each objective function $m = 1, 2, \ldots, M$, sort the set in worse order of f_m or, find the sorted indices vector: $I^m = \text{sort}(f_m, >)$.

Step C3 For $m = 1, 2, \ldots, M$, assign a large distance to the boundary solutions, or $d_{I_1^m} = d_{I_l^m} = \infty$, and for all other solutions $j = 2$ to $(l-1)$, assign

$$d_{I_j^m} = d_{I_j^m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{\max} - f_m^{\min}}.$$

Index I_j denotes the solution index of the jth member in the sorted list. Thus, for any objective, I_1 and I_l denote the lowest and highest objective function values, respectively. The second term on the right-hand side of the last equation is the difference in objective function values between two neighboring solutions on either side of solution I_j. Thus, this metric denotes half of the perimeter of the enclosing cuboid with the nearest-neighboring solutions placed on the vertices of the cuboid (Fig. 15.12). It is interesting to note that for any solution i the same two solutions $(i+1)$ and $(i-1)$ need not be neighbors in all objectives, particularly for $M \geq 3$. The parameters $f_m^{\max}$ and $f_m^{\min}$ can be set as the population-maximum and population-minimum values of the mth objective function. The above metric requires M sorting calculations in Step C2, each requiring $O(N \log N)$ computations. Step C3 requires N computations. Thus, the complexity of the above distance metric computation is $O(MN \log N)$ and the overall complexity of one generation of NSGA-II is $O(MN^2)$, governed by the non-dominated sorting procedure.

15.4.4 Sample Simulation Results

In this section, we show the simulation results of NSGA-II on two test problems. The first problem (SCH1) is simple two-objective problem with a convex Pareto-optimal front:

$$\text{SCH1}: \begin{cases} \text{Minimize } f_1(x) = x^2 \\ \text{Minimize } f_2(x) = (x-2)^2 \\ \qquad -10^3 \leq x \leq 10^3. \end{cases} \tag{15.5}$$

The second problem (KUR) has a disjointed set of Pareto-optimal fronts:

$$\text{KUR}: \begin{cases} \text{Minimize } f_1(\mathbf{x}) = \sum_{i=1}^{2}\left[-10\exp(-0.2\sqrt{x_i^2+x_{i+1}^2}\right] \\ \text{Minimize } f_2(\mathbf{x}) = \sum_{i=1}^{3}\left[|x_i|^{0.8}+5\sin(x_i^3)\right] \\ \qquad\qquad -5 \le x_i \le 5, \quad i = 1,2,3. \end{cases} \tag{15.6}$$

NSGA-II is run with a population size of 100 and for 250 generations. Figure 15.13 shows that NSGA-II converges on the Pareto-optimal front and maintains a good spread of solutions. In comparison to NSGA-II, another competing EMO method—the Pareto archived evolution strategy (PAES) (Knowles and Corne 2000)—is run for an identical overall number of function evaluations and an inferior distribution of solutions on the Pareto-optimal front is observed.

On the KUR problem, NSGA-II is compared with another elitist EMO methodology—the strength Pareto EA or SPEA (Zitzler and Thiele 1998)—for an identical number of function evaluations. Figures 15.14 and 15.15 clearly show the superiority of NSGA-II in achieving both tasks of convergence and maintaining diversity of optimal solutions.

15.4.5 Other State-of-the-Art MOEAs

Besides the above elitist EMO method, there exist a number of other methods which are also quite commonly used. Of them, the strength Pareto-EA or SPEA2 (Zitzler et al. 2001b), which uses an EA population and an archive in a synergistic manner and the Pareto envelope-based selection algorithm or PESA (Corne et al. 2000),

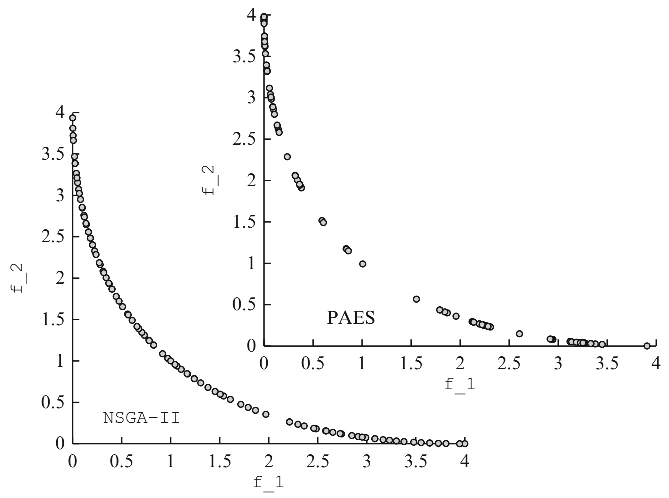

Fig. 15.13 NSGA-II finds better spread of solutions than PAES on SCH

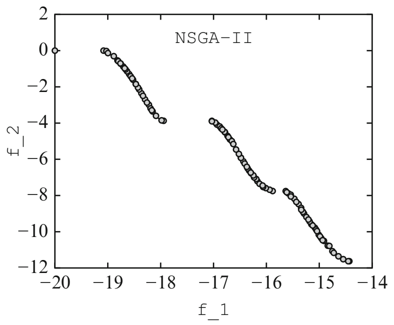

Fig. 15.14 NSGA-II on KUR

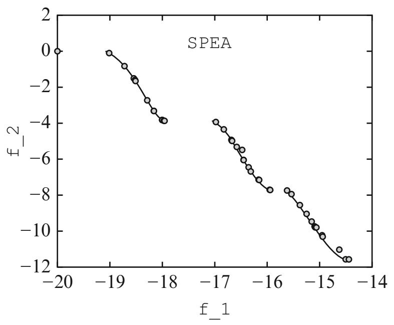

Fig. 15.15 SPEA on KUR

which emphasizes non-dominated solutions residing in a less-crowded hyper-box in both the selection and the offspring-acceptance operators, are common. The ε-MOEA procedure (Deb et al. 2003b) is found to be a superior version of PESA, in which only one solution is allowed to occupy a hyper-box for obtaining a better distribution of solutions. In addition, the ε-dominance concept (Laumanns et al. 2002a) makes the MOEA a practical approach for solving complex problems with a large number of objectives. The ε-MOEA is also demonstrated to find a well-converged and well-distributed set of solutions in a very small computational time (two to three orders of magnitude smaller) compared to a number of state-of-the-art MOEAs (Deb et al. 2003b), such as SPEA2 and PAES. There also exist other competent MOEAs, such as multi-objective messy GA (MOMGA) (Veldhuizen and Lamont 2000), multi-objective micro-GA (Coello and Toscano 2000), neighborhood constraint GA (Loughlin and Ranjithan 1997), and others. Further, there exist other EA-based methodologies, such as particle swarm EMO (Coello and Lechuga 2002; Mostaghim and Teich 2003), ant-based EMO (McMullen 2001; Gravel et al. 2002) and differential evolution-based EMO (Babu and Jehan 2003).

15.5 Constraint Handling

Constraints can be simply handled by modifying the definition of domination in an EMO method.

Definition 15.8. A solution $\mathbf{x}^{(i)}$ is said to “constrain-dominate” a solution $\mathbf{x}^{(j)}$ (or $\mathbf{x}^{(i)} \preceq_c \mathbf{x}^{(j)}$) if any of the following conditions are true:

1. Solution $\mathbf{x}^{(i)}$ is feasible and solution $\mathbf{x}^{(j)}$ is not.
2. Solutions $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are both infeasible, but solution $\mathbf{x}^{(i)}$ has a smaller constraint violation.
3. Solutions $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are feasible and solution $\mathbf{x}^{(i)}$ dominates solution $\mathbf{x}^{(j)}$ in the usual sense (see Definition 15.3).

This definition allows a feasible solution to be always dominating an infeasible solution and compares two infeasible solutions based on constraint violation values and two feasible solutions in terms of their objective values.

In the following, we show simulation results of NSGA-II applied with the above constraint handling mechanism to two test problems—the CONSTR and the problem TNK described below:

CONSTR

$$\begin{aligned} &\text{Minimize } f_1(\mathbf{x}) = x_1 \\ &\text{Minimize } f_2(\mathbf{x}) = \frac{1+x_2}{x_1} \\ &\qquad x_2 + 9x_1 \geq 6 \\ &\qquad -x_2 + 9x_1 \geq 1 \end{aligned}$$

TNK

$$\begin{aligned} &\text{Minimize } f_1(\mathbf{x}) = x_1 \\ &\text{Minimize } f_2(\mathbf{x}) = x_2 \\ &\qquad x_1^2 + x_2^2 - 1 - \tfrac{1}{10}\cos\left(16\tan^{-1}\tfrac{x_1}{x_2}\right) \geq 0 \\ &\qquad (x_1-0.5)^2 + (x_2-0.5)^2 \leq 0.5 \end{aligned}$$

With identical parameter settings as in Sect. 15.4.4, NSGA-II finds a good distribution of solutions on the Pareto-optimal front in both problems (Figs. 15.16 and 15.17, respectively).

15.6 Some Applications

Since the early development of MOEAs in 1993, they have been applied to many real-world and interesting optimization problems. Descriptions of some of these studies can be found in books (Deb 2001; Coello et al. 2002; Osyczka 2002), conference proceedings (Zitzler et al. 2001a), and domain-specific journals and conference proceedings. In this section, we describe two case studies.

15.6.1 Spacecraft Trajectory Design

Coverstone-Carroll et al. (2000) proposed a multi-objective optimization technique using the original non-dominated sorting (NSGA) (Srinivas and Deb 1994) to find

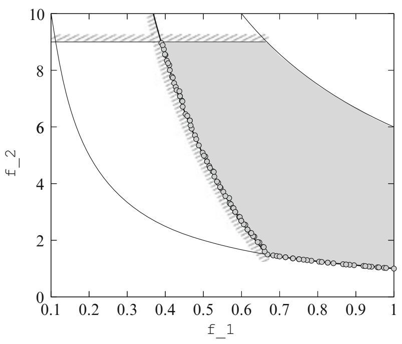

Fig. 15.16 Obtained non-dominated solutions with NSGA-II on the constrained problem CONSTR

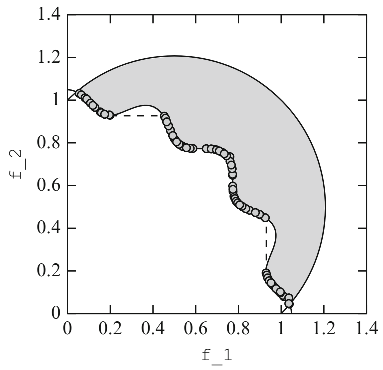

Fig. 15.17 Obtained non-dominated solutions with NSGA-II on the constrained problem TNK

multiple trade-off solutions in a spacecraft trajectory optimization problem. To evaluate a solution (trajectory), the SEPTOP software is called for, and the delivered payload mass and the total time of flight are calculated. In order to reduce the computational complexity, the SEPTOP program is run for a fixed number of generations. The MOOP had eight decision variables controlling the trajectory, as well as three objective functions, i.e. (i) maximize the delivered payload at destination, (ii) maximize the negative of the time of flight, and (iii) maximize the total number of heliocentric revolutions in the trajectory, and three constraints, i.e. (i) limiting the SEPTOP convergence error, (ii) limiting the minimum heliocentric revolutions, and (iii) limiting the maximum heliocentric revolutions in the trajectory.

On the Earth–Mars rendezvous mission, the study found interesting trade-off solutions. Using a population of size 150, the NSGA was run for 30 generations on a Sun Ultra 10 Workstation with a 333 MHz ULTRA Sparc IIi processor. The obtained non-dominated solutions are shown in Fig. 15.18 for two of the three objectives. It is clear that there exist short-time flights with smaller delivered payloads (solution marked as 44) and long-time flights with larger delivered payloads (solution

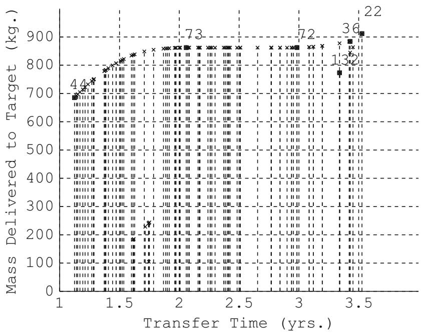

Fig. 15.18 Obtained non-dominated solutions

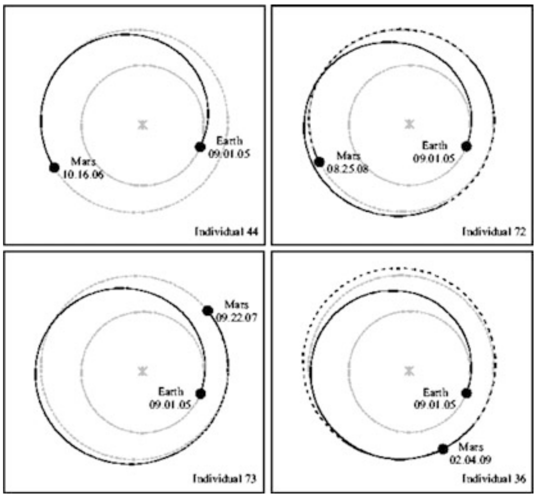

Fig. 15.19 Four trade-off trajectories

marked as 36). To the surprise of the original investigators, two different types of trajectories emerged. The representative solutions of the first set of trajectories are shown in Fig. 15.19. Solution 44 can deliver a mass of 685.28 kg and requires about 1.12 years. On the other hand, solution 72 can deliver almost 862 kg with a travel time of about 3 years. In these figures, each continuous part of a trajectory represents a thrusting arc and each dashed part of a trajectory represents a coasting arc. It is interesting to note that only a small improvement in delivered mass occurs in the solutions between 73 and 72. To move to a somewhat improved delivered mass, a different strategy for the trajectory must be found. Near solution 72, an additional burn is added, causing the trajectories to have better delivered masses. Solution 36 can deliver a mass of 884.10 kg.

The scenario as in Fig. 15.19 is what we envisaged in discovering in a MOOP while suggesting the *ideal procedure* in Fig. 15.3. Once such a set of solutions with a good trade-off among objective values is obtained, one can then analyze them in order to choose a particular solution. For example, in this problem context, whether the wait of an extra year to be able to carry an additional 180 kg of payload is worthwhile or not would lead a decision-maker to choose between solutions 44 and 73. Without the knowledge of such a wide variety of optimal solutions, the decision-making could be difficult. Although one can set a relative weight to each objective and optimize the resulting aggregate objective function, the decision-maker will always wonder what solution would have been derived if a slightly different weight vector had been used. The ideal multi-objective optimization technique allows for a flexible and a pragmatic procedure for analyzing a well-diversified set of solutions before choosing a particular solution.

15.6.2 A Cantilever Plate Design

A rectangular plate ($1.2 \times 2\,\mathrm{m}^2$) is fixed at one end and a 100 kN load is applied to the center element of the opposite end. The following other parameters are chosen:

- Plate thickness: 20 mm
- Yield strength: 150 MPa
- Young's modulus: 200 GPa
- Poisson ratio: 0.25.

The rectangular plate is divided into a number of grids and the presence or absence of each grid becomes a Boolean decision variable. NSGA-II is applied for 100 generations with a population size of 54 and crossover probability of 0.95. In order to increase the quality of the obtained solutions, we use an incremental grid-tuning technique. The NSGA-II and the first local search procedure are run with a coarse grid structure (6×10 or 60 elements). After the first local search procedure, each grid is divided into four equal-sized grids, thereby having a 12×20 or 240 elements. The new smaller elements inherit its parent's status of being present or absent. After the second local search is over, the elements are divided again, thereby making 24×40 or 960 elements. In all cases, an automatic mesh-generating finite element method is used to analyze the developed structure.

Figure 15.20 shows the obtained front with eight solutions: the trade-off between weight and deflection is clear. Figure 15.21 shows the shape of these eight solutions. The solutions are arranged according to increasing weight from left to right and top to bottom. Thus, the minimum-weight solution is the top-left solution and the minimum-deflection solution is the bottom-right solution.

One striking difference between single-objective optimization and multi-objective optimization is the cardinality of the solution set. In the latter, multiple solutions are the outcome and each solution is theoretically an optimal solution corresponding to a particular trade-off among the objectives. Thus, all such trade-off solutions found by an EMO are high-performing near-optimal solutions. It is intuitive to realize that these solutions will possess some common properties that qualify them to be near

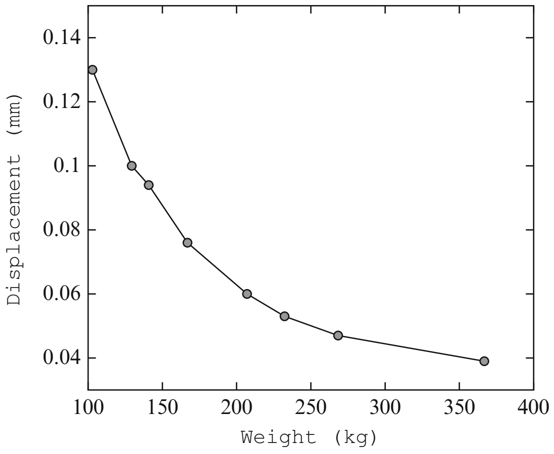

Fig. 15.20 Obtained front with eight clustered solutions shown for the cantilever plate design problem

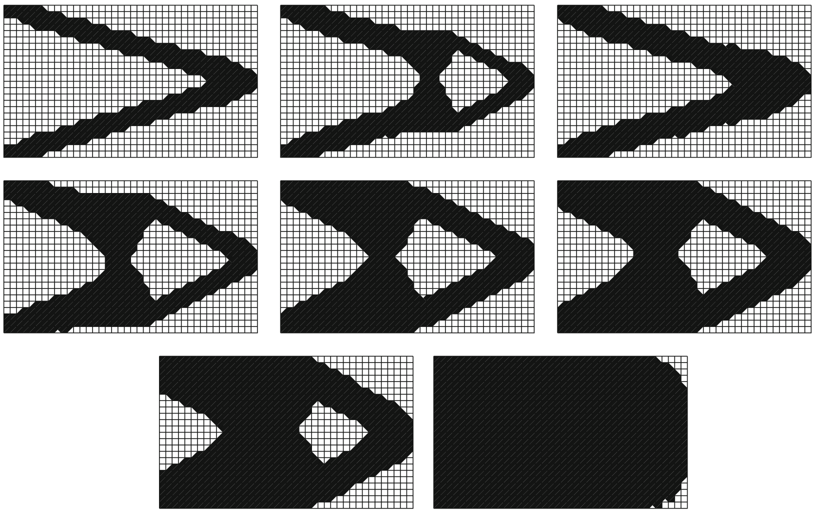

Fig. 15.21 Eight trade-off solutions of the cantilever plate design problem

Pareto-optimal. The author first proposed an analysis procedure to decipher hidden relationships common to EMO-obtained trade-off solutions in 1993 (Deb 2003) and later (Deb and Srinivasan 2006) termed the task as *innovization* procedure—revealing innovation through optimization. Such useful properties are expected to exist in practical problems, as they follow certain scientific and engineering principles at the core. In the recent past, the author and his student have devised several automated innovization procedures that take EMO solutions and churn out multiple hidden mathematical relationships of certain structure through an optimization task (Bandaru and Deb 2010, 2011a,b).

We consider the canteliver plate design problem to illustrate the usefulness of the innovization task. The obtained nine solutions are manually analyzed and the following interesting insights are revealed as properties of those solutions:

1. First, all nine solutions seem to be symmetric about the middle row of the plate. Since the loading and support are symmetrically placed around the middle row, the resulting optimum solution is also likely to be symmetric. Although this information is not explicitly coded in the hybrid NSGA-II procedure, this emerges as one of the features in all optimal solutions. Although in this problem, it is difficult to know the true Pareto-optimal solutions, the symmetry achieved in these solutions is an indication of their proximity to the true Pareto-optimal solutions.
2. The minimum-weight solution simply extends two arms from the extreme support nodes to reach to the element carrying the load. Since a straight line is the shortest way to join two points, this solution can be easily conceived as one close to the minimum-weight feasible solution.
3. Thereafter, to have a reduction in deflection, the weight has to be increased. This is where the hybrid procedure discovers an innovation. For a particular sacrifice in the weight, the procedure finds that the maximum reduction in deflection occurs when the two arms are connected by a *stiffener*. This is an engineering trick often used to design a stiff structure. Once again, no such information was explicitly coded in the hybrid procedure. By merely making elements on or off, the procedure has resulted in a *design innovation*.
4. Interestingly, the third solution is a thickened version of the minimum-weight solution. By making the arms thicker, the deflection can be increased maximally for a fixed change in the weight from the previous solution. Although not intuitive, this thick-arm solution is not an immediate trade-off solution to the minimum-weight solution. Although the deflection of this solution is smaller compared to the second solution, the stiffened solution is a good compromise between the thin- and thick-armed solutions.
5. Thereafter, any increase in the thickness of the two-armed solution turns out to be a suboptimal proposition and the stiffened solution is rediscovered instead. From the support to the stiffener, the arms are now thicker than before, providing better stiffness than before.
6. In the remaining solutions, the stiffener and arms get wider and wider, finally leading to the complete plate with rounded corners. This solution is, no doubt, close to the true minimum-deflection solution.

The transition from a simple thin two-armed cantilever plate having a minimum-weight solution to a complete plate with edges rounded off having a minimum-deflection solution proceeds by discovering a vertical stiffener connecting the two arms and then by widening the arms and then by gradually thickening the stiffener. The symmetric feature of the solutions about the middle row of the plate has emerged to be a *common* property of all obtained solutions. Such information about the trade-off solutions is very useful to a designer. Importantly, it is not obvious how such vital design information can be obtained by any other means and in a single simulation run.

15.7 Tricks of the Trade

Here, we discuss how to develop an ideal multi-objective optimization algorithm in a step-by-step manner. Since they can be developed by using classical or evolutionary optimization methods, we discuss each of these in turn.

15.7.1 Classical Multi-objective Optimization

We assume here that the user knows a classical optimization method to optimize a single-objective optimization problem P with constraints ($\mathbf{x} \in S$) and can find a near-optimum solution $\mathbf{x}^*$. Let us assume that the user desires to find K different efficient solutions.

Step 1 Find individual optimum solutions $\mathbf{x}^{*i}$ for all objectives, $i = 1, 2, \ldots, M$.
Step 2 Choose K points $\varepsilon^{(k)}$ uniformly in the $(M-1)$-dimensional plane having objectives $i = 1$ to $i = M-1$ as coordinate directions.
Step 3 Solve each subproblem ($k = 1, 2, \ldots, K$) as follows:

$$\begin{aligned} &\text{Minimize } f_M(\mathbf{x}) \\ &\text{subject to } f_i(\mathbf{x}) \le \varepsilon_i^{(k)}, \quad i = 1, 2, \ldots, (M-1) \\ &\qquad\qquad \mathbf{x} \in S. \end{aligned} \tag{15.7}$$

Call the optimal solution $\mathbf{x}^{*(k)}$ and corresponding objective vector $\mathbf{f}^{*(k)}$.
Step 4 Declare the non-dominated set, $F = \text{non-dominated}(\mathbf{f}^{*(1)}, \ldots, \mathbf{f}^{*(K)})$, as the set of efficient solutions.

If desired, the above ε-constraint method can be replaced by other conversion methods, such as the weighted-sum method or the Tchebyshev metric method (Deb 2001; Miettinen 1999; Chankong and Haimes 1983).

15.7.2 Evolutionary Multi-objective Optimization (EMO)

The bottleneck of the above method is Step 3, in which a single-objective minimizer needs to be applied K times (where K is the number of desired efficient solutions). Here, we discuss an evolutionary search principle to find a set of efficient solutions simultaneously in a synergistic manner. It must be kept in mind that the main aim in an ideal multi-objective optimization is to (i) converge close to the true Pareto-optimal front and (ii) maintain a good diversity among them. Thus, an EMO method must use specific operations to achieve each of the above goals. Usually, an emphasis on non-dominated solutions is performed to achieve the first goal and a niching operation is performed to achieve the second goal. In addition, an elite-preserving operation is used to speed up convergence.

Again, we assume that the user is familiar with a particular population-based evolutionary algorithm, in which in each generation one or more new offspring are created by means of recombination and mutation operators. We describe here a generic archive-based EMO strategy.

Step 1 A population P and an empty archive A are initialized. The non-dominated solutions of P are copied in A. Steps 2 and 3 are iterated till a termination criterion is satisfied.

Step 2 A set of λ new offspring solutions are created using P and A.

Step 3 Every new offspring solution is checked for its inclusion in A by using a archive-acceptance criterion C_A and for its inclusion in P by using a population-acceptance criterion C_P. If an offspring is not to be included to either P or A, it is deleted.

Step 4 Before termination, the non-dominated set of the archive A is declared as the efficient set.

In Step 2, random solutions from the combined set $P \cup A$ can be used to create an offspring solution or some solution from P and some other solutions from A can be used to create the offspring solution. Different archive-acceptance and population-acceptance criteria can be used. Here, we propose one criterion each. Readers can find another implementation elsewhere (Deb et al. 2003b).

15.7.2.1 Archive Acceptance Criterion $C_A(c,A)$

The archive A has a maximum size K, but at any iteration it may not have all K members. This criterion required domination check and a niching operator which computes the niching of the archive with respect to a particular solution. For example, the crowding distance metric for a solution i in a subpopulation (suggested in Sect. 15.4.3) measures the objective-wise distance between two neighboring solutions of solution i in the subpopulation. The larger the crowding distance, the less crowded is the solution and the higher is probability of its existence in the subpopulation.

The offspring c is accepted in the archive A if any of the following conditions is true:

1. Offspring c dominates any archive member A. In this case, delete those archive members and include c in A.
2. Offspring c is non-dominated with all archive members and the archive is not *full* (that is, $|A| < K$). In this case, c is simply added to A.
3. Offspring c is non-dominated with all archive members, the archive is full, and crowding distance of c is larger than that of an archive member. In this case, that archive member is deleted and c is included in A.

15.7.2.2 Population Acceptance Criterion $C_P(c,P)$

If the offspring is a good solution compared to the current archive, it will be included in A by the above criterion. The inclusion of the offspring in the population must be made mainly from the point of view of keeping diversity in the population. Any of the following criteria can be adopted to accept c in P:

1. Offspring c replaces the most old (in terms of its time of inclusion in the population) population member.
2. Offspring c replaces a random population member.
3. Offspring c replaces the least-used (as a parent) population member.
4. Offspring c introduces more diversity of the population compared to an existing population member. Here the crowding distance or an entropy-based metric can be used to compute the extent of diversity.
5. Offspring c replaces a population member *similar* (in terms of its phenotype or genotype) to itself.
6. Offspring c replaces a population member dominated by c.

It is worth the effort to investigate which of the above criteria works the best in standard test problems, but the maintenance of diversity in the population and search for widespread non-dominated solutions in the archive are two activities which should allow the combined algorithm to reach the true efficient frontier quickly and efficiently.

In the following, we suggest some important post-optimality studies which are equally important to the optimality study and are often ignored in EMO studies.

15.7.3 Post-optimality Studies

It should be well understood that an EMO method (no matter whether it is the above one or any of the existing ones) does not have a guaranteed convergence properties; nor do they have any guaranteed proof for finding a well-diversed set of solutions. Thus, there is an onus on the part of EMO researchers/practitioners to perform a number of post-optimality studies to ensure (or build confidence about) convergence and achievement of diversity in obtained solutions. Here, we make some suggestions.

1. Use a hybrid EMO–local search method. From each of the obtained EMO solution, perform a single-objective search by optimizing a combined objective function—see Chap. 9.6 in Deb (2001) for more details. This will cause the EMO solutions to reach near to the true efficient frontier.
2. Obtain individual optimum solutions and compare the obtained EMO solutions with the individual optima on a plot or by means of a table. Such a visual comparison will indicate the extent of convergence as well as the extent of diversity in the obtained solutions.

3. Perform a number of ε-constraint studies for different values of the ε-vector, given in Eq. (15.7) and obtain efficient solutions. Compare these solutions with the obtained EMO solutions to get further visual confirmation of the extent of convergence and diversity of obtained EMO solutions.
4. Finally, it is advisable to also plot the initial population on the same objective space showing the efficient solutions, as this will depict the extent of optimization performed by the EMO local search approach.

For such a post-optimality study, refer to any of the application studies performed by the author (Deb and Jain 2003; Deb et al. 2004b; Deb and Tiwari 2004; Deb et al. 2004a).

For practical problems, we also suggest performing an *innovization* study after multiple trade-off solutions are found to reveal useful properties that are common to them. As shown in Sect. 15.6.2 and in other studies (Deb and Srinivasan 2006; Bandaru and Deb 2011b), such a task elicits useful knowledge that is usually more valuable than just the discovery of a set of trade-off solutions.

15.7.4 Evaluating a Multi-objective Optimization Algorithm

When a new algorithm is suggested to find a set of Pareto-optimal solutions in a MOOP, the algorithm has to be evaluated by applying them on standard test problems and compared with existing state-of-the-art EMO algorithms applied to the identical test problems. Here, we suggest a few guidelines in this direction.

1. Test problems with 20–50 variables must be included in the test set.
2. Test problems with three or more objectives must be included in the test set. For scalable test problems, readers may refer to WFG (Huband et al. 2005) and DTLZ (Deb et al. 2005) test problems.
3. Test problems must include some non-linear constraints, making some portion of the unconstrained Pareto-optimal front infeasible. For a set of constrained test problems, see the CTP problems (Deb 2001) or DTLZ test problems.
4. Standard EMO algorithms such as NSGA-II, SPEA2, PESA, ε-MOEA, and others must have to be used for comparison purposes. See the section *Sources of Additional Information* for some freely downloadable codes of these algorithms.
5. A proper criterion for the comparison must be chosen. Often, the algorithms are compared based on the fixed number of evaluations. They can also be compared based on some other criterion (Deb 2001).
6. Besides static performance metrics which are applied to the final solution set, *running metrics* (Deb and Jain 2002) may also be computed, plotted with generation number, and compared among two algorithms. The running metrics provide a dynamic (generation-wise) evaluation of the algorithm, rather than what had happened at the end of a simulation run.

15.8 Research Challenges

With the growing interest in the field of multi-objective optimization, particularly using evolutionary algorithms, there exist a number of research directions:

EMO and decision making: EMO methodologies have amply shown that multiple and well-spread Pareto-optimal solutions can be obtained in a single simulation run for a few objectives (four or less). We discuss solving *many-objective* problems later in this section. However, finding a set of trade-off solutions is only a part of the whole story. In practice, one needs to choose only a single preferred solution. Such a task requires one to use a multiple-criterion decision-making (MCDM) technique (Miettinen 1999; Belton and Stewart 2002; Chankong and Haimes 1983; Collette and Siarry 2004; Tzeng and Huang 2011). The combination of EMO and MCDM can, in principle, be used in three possible ways:

1. *A priori* approach, in which preference information can be determined before any optimization task is performed, like in scalarization methods, such as weighted-sum, ε-constraint, and other methods (Chankong and Haimes 1983). As discussed above, determining a preference information without any knowledge of trade-off solutions becomes a difficult task. However, if a priori information is used to find a preferred region of the Pareto-optimal front, instead of a single preferred solution, MCDM techniques can be used with an EMO to find a predefined focussed region (Deb et al. 2006; Deb and Kumar 2007a,b).
2. *A posteriori* approach, in which a set of trade-off solutions is first found and then a MCDM technique is used to anlayse the solutions to choose a single preferred solution (Deb 2001; Coverstone-Carroll et al. 2000). Although the decision-making becomes relatively meaningful when a set of trade-off solutions are available, the approach becomes difficult to apply in problems having five or more objectives, as finding a set of trade-off solutions for a large set of objectives is still a difficult task.
3. *Interactive* approach, in which preference information is used during the EMO process iteratively, so that the combined approach is directed towards the preferred part of the Pareto-optimal region. A couple of such hybrid methods have been suggested recently (Deb et al. 2010; Branke et al. 2009).

In one of the interactive studies (Deb et al. 2010), after every 10 or 20 generations, a few (four or five) clustered non-dominated solutions are presented to the decision-maker. Using MCDM techniques, the decision-maker then provides partial or complete preference information by performing pair-wise comparisons of these solutions. These information are then used to build a mathematical utility function honoring the decision-maker's preference information. For the next 10 or 20 generations the EMO modifies its domination principle with the developed utility function so as to focus its search towards the preferred part of the search space. These approaches are practical and promise to handle many-objective problems using a combined EMO and MCDM approach.

Handling many objectives: So far, most studies using EMO strategies have been restricted to two- or three-objective problems. In practice, there exist a considerable number of problems in which 10 or 15 objectives are commonplace. Existing domination-based EMO approaches have difficulties in solving such large-dimensional problems due to the following reasons:

- A large fraction of the population becomes non-dominated to each other for a large number of objectives, as there are many ways a solution can become non-dominated. In an EMO approach emphasizing all non-dominated solutions, such a method does not keep many population slots free for new solutions, thereby slowing the search.
- A diversity preservation procedure becomes computationally expensive to determine the extent of crowding of solutions.
- An exponentially large number of solutions are required to represent a large-dimensional Pareto-optimal front, thereby requiring an exponentially large population (or an archive) to store trade-off solutions.
- Comparison of two sets of trade-off solutions for set-based EMO approaches (Zitzler et al. 2010; Zitzler and Künzli 2004) becomes a difficult task for a large-dimensional problem. Estimation of set-based metrics, such as hypervolume, becomes computationally expensive.
- Although not specific to EMO algorithms, visualization of a large-dimensional dataset becomes difficult.

However, recent studies on many objective optimization problems (Zhang and Li 2007; Knowles and Corne 2007; López and Coello Coello 2009; Corne and Knowles 2007; Hughes 2005; Ishibuchi et al. 2008; Deb and Jain 2012) have shown that computationally tractable EMO algorithms can be developed by using fixed search directions or fixed reference points to handle large-dimensional problems.

Non-evolutionary multi-objective optimization: EMO methods include principles of genetic algorithms, evolution strategy, genetic programming, particle swarm optimization, differential evolution and others. But other non-traditional optimization techniques such as ant colony optimization, tabu search and simulated annealing can also be used for solving MOOPs. Although there has been some research and application in this direction (Hansen 1997; Khor et al. 2001; Balicki and Kitowski 2001; Bandyopadhyay et al. 2008; McMullen 2001; Gravel et al. 2002; Kumral 2003; Parks and Suppapitnarm 1999; Chattopadhyay and Seeley 1994), more rigorous studies are called for, and such techniques can also be suitably used to find multiple Pareto-optimal solutions.

Performance metrics: For M objectives, the Pareto-optimal region will correspond to at most an M-dimensional surface. To compare two or more algorithms, it is then necessary to compare M-dimensional data sets which are partially ordered. It is not possible to have only one performance metric to compare such multi-dimensional data sets in an unbiased manner. A study has shown that at least M performance metrics are necessary to properly compare such data sets (Zitzler et al. 2003). An alternative pragmatic suggestion was to compare the data sets from a purely *functional* point of view of (i) measuring the extent of convergence to the front and (ii) measuring the extent of diversity in the obtained solutions (Deb 2001). It then becomes a challenge to develop performance metrics for both functional goals for problems having any number of objectives. The hypervolume metric (Zitzler and Thiele 1999) has received a lot of attention among the EMO researchers, due to its ability to provide a combined estimate of convergence and diversity of a set of solutions. However, the computation of hypervolume for more than three or four objective problems is time-consuming. Researchers have devised approximate procedures for estimating hypervolume in higher dimensions (Bradstreet et al. 2008; While et al. 2006; Bader et al. 2010).

Test problem design: When new algorithms are designed, they need to be evaluated on test problems for which the desired Pareto-optimal solutions are known. Moreover, the test problems must be such that they are controllable to test an algorithm's ability to overcome a particular problem difficulty. Although there exist a number of such test problems (Deb 2001; Deb et al. 2005), more such problems providing different kinds of difficulties must be developed. Care should be taken to make sure that the test problems are scalable to any number of objectives and decision variables, so that systematic evaluation of an algorithm can be performed. Recent test problems (Zhang et al. 2008; Huband et al. 2005) are some efforts in this direction.

Parallel EMO methodologies: With the availability of parallel or distributed processors, it may be wise to find the complete Pareto-optimal front in a distributed manner. A study (Deb et al. 2003a) has suggested such a procedure based on a guided-domination concept, in which one processor focuses on finding only a portion of the Pareto-optimal front. With intermediate cross-talks between the processors, the procedure has shown that the complete Pareto-optimal front can be discovered by concatenating the solutions from a number of processors. Since each processor works on a particular region in the search space and processors communicate between themselves, a faster and parallel search is expected from such an implementation. Other similar parallelization techniques must be attempted and evaluated. Parallel EMO algorithms are also required to be developed for GPU-based computing platforms to take advantage of the recent enhancement of GPU technology. Some efforts in this direction are by Sharma and Collet (2010) and Wong (2009).

Multi-objectivization using EMO principle: Over the past few years and since the development of EMO methodologies, they have been also used to help solve a number of other optimization problems, such as (i) in reducing bloating problems commonly found in genetic programming applications (Bleuler et al. 2001), (ii) in goal programming problems (Deb 1999), (iii) in maintaining diversity in a single-objective EA (Jensen 2003a), (iv) single-objective constraint-handling problems (Coello 2000; Surry et al. 1995), (v) solving constrained optimization problems (Deb and Datta 2010), (vi) solving multimodal problems (Deb and Saha 2012), and others. Because of the use of additional objectives signifying a desired effect to be achieved, the search procedure becomes more flexible. More such problems, which reportedly perform poorly due to some fixed or rigid solution procedures, must be tried using a multi-objective approach. A recent edited book (Knowles et al. 2008) presents many such recently proposed multi-objectivization studies.

EMO for redundant objectives: Many practical problems may have a large number of objectives, but the Pareto-optimal front of the problem may be lower-dimensional. In such problems, certain objectives get correlated to each other as the solutions approach the Pareto-optimal front. In such methods, the redundancy in objectives are determined by various means—through a principal component analysis (PCA) (Deb and Saxena 2006), nonlinear PCA analysis (Saxena et al. 2013), and other means (Brockhoff and Zitzler 2006, 2007)—as the algorithms progress and the redundant objectives are eliminated to reduce the cardinality of the objectives. In problems having 50 objectives, the PCA-based NSGA-II procedure (Deb and Saxena 2006) was shown to reduce 48 correlated objectives successively with generations. Further such studies with better computational efficiency are needed for application to real-world problems.

Theoretical developments: One aspect for which EMO can be criticized is the lukewarm interest among its researchers to practice much theory related to their working principles or convergence behaviors. Apart from a few studies (Rudolph 1998; Rudolph and Agapie 2000), this area still remains a fertile field for theoretically oriented researchers to dive into and suggest algorithms with a good theoretical basis. Algorithms with a time complexity analysis on certain problems have been started (Giel 2003; Laumanns et al. 2002b, 2004) and research in this area should grow more popular in trying to devise problem–algorithm combinations with an estimated computational time for finding the complete Pareto-optimal set.

EMO on dynamic problems: Dynamic optimization involves objectives, constraints or problem parameters which change over time (Branke 2001). This means that as an algorithm is approaching the optimum of the current problem, the problem definition changes and now the algorithm must solve a new problem. Often, in such dynamic optimization problems, an algorithm is usually not expected to find the optimum, instead it is expected to track the changing optimum with iteration. A study (Deb et al. 2007) proposed the following procedure for dynamic optimization involving single or multiple objectives. Let $\mathcal{P}(t)$ be a problem which changes with time t (from $t = 0$ to $t = T$). Despite the continual change in the problem, we assume that the problem is fixed for a time period τ, which is not known a priori and the aim of the (offline) dynamic optimization study is to identify a suitable value of τ for an accurate as well as computationally faster approach. For this purpose, an optimization algorithm with τ as a fixed time period is run from $t = 0$ to $t = T$ with the problem assumed fixed for every τ time period. A measure $\Gamma(\tau)$ determines the performance of the algorithm and is compared with a pre-specified and expected value Γ_L. If $\Gamma(\tau) \geq \Gamma_L$, for the entire time domain of the execution of the procedure, we declare τ to be a permissible length of stasis. Then, we try with a reduced value of τ and check if a smaller length of statis is also acceptable. If not, we increase τ to allow the optimization problem to remain static for a longer time so that the chosen algorithm can now have more iterations (time) to perform better. Such a procedure will eventually come up with a time period τ^* which would be the smallest time of statis allowed for the optimization algorithm to work based on the chosen performance requirement. Based on this study, a number of test problems and a hydro-thermal power dispatch problem have been tackled (Deb et al. 2007).

In the case of dynamic multi-objective problem-solving tasks, there is an additional difficulty which is worth mentioning here. Not only does an EMO algorithm needs to find or track the changing Pareto-optimal fronts, but in a real-world implementation, it must also make an immediate decision about which solution to implement from the current front before the problem changes to a new one. Decision-making analysis is considered to be time-consuming, involving execution of analysis tools, higher-level considerations, and sometimes group discussions. If dynamic EMO is to be applied in practice, *automated* procedures for making decisions must be developed. Although it is not clear how to generalize such an automated decision-making procedure in different problems, problem-specific tools are possible and certainly a worthwhile and fertile area for research.

Real-world applications: Although the usefulness of EMO and classical multi-objective optimization methods are increasingly being demonstrated by solving real-world problems (Coello and Lamont 2004), more complex and innovative applications would not only demonstrate the widespread applicability of these methods but also may open up new directions for research.

15.9 Conclusions

For the past two decades, the usual practice of treating MOOPs by scalarizing them into a single objective and optimizing it has been seriously questioned. The presence of multiple objectives results in a number of Pareto-optimal solutions, instead of a single optimum solution. In this tutorial, we have discussed the use of an ideal multi-objective optimization procedure which attempts to find a well-distributed set of Pareto-optimal solutions first. It has been argued that choosing a particular solution as a post-optimal event is a more convenient and pragmatic approach than finding an optimal solution for a particular weighted function of the objectives. Besides introducing the multi-objective optimization concepts, this tutorial also has also presented two commonly used MOEAs.

Besides finding the multiple Pareto-optimal solutions, the suggested ideal multi-objective optimization procedure has another unique advantage. Once a set of Pareto-optimal solutions are found, they can be analyzed. The principle behind the *transition* from the optimum of one objective to that of other objectives can be investigated as a post-optimality analysis. Since all such solutions are optimum with respect to certain trade-off between objectives, the transition should reveal interesting knowledge on an optimal process of sacrifice of one objective to get a gain in other objectives.

The field of MOEAs has now matured. Nevertheless, there exist a number of interesting and important research topics which must be investigated before their full potential is unearthed. This tutorial has suggested a number of salient research topics to motivate newcomers to pay further attention to this growing field of importance.

Sources of Additional Information

Here, we outline some dedicated literature in the area of evolutionary multi-objective optimization and decision-making.

Books in Print

- C. A. C. Coello, D. A. VanVeldhuizen, and G. Lamont (2002). *Evolutionary algorithms for solving multi-objective problems*. Kluwer, Boston—a good reference book with a good citation of most EMO studies up to 2001.
- A. Osyczka (2002). *Evolutionary algorithms for single and multicriteria design optimization*. Physica-Verlag, Heidelberg—a book describing single and multi-objective EAs with lots of engineering applications.
- K. Deb (2001). *Multi-objective optimization using evolutionary algorithms*. Wiley, Chichester (2nd edn, with exercise problems)—a comprehensive book introducing the EMO field and describing major EMO methodologies and some research directions.
- K. Miettinen (1999). *Nonlinear multiobjective optimization*. Kluwer, Boston—a good book describing classical multi-objective optimization methods and a extensive discussion on interactive methods.
- M. Ehrgott (2000). *Multicriteria optimization*. Springer, Berlin—a good book on the theory of multi-objective optimization.

Conference Proceedings

The following six conference proceedings spanning from 2001 to 2013 are most useful on the theory, algorithms, and application of EMO.

- Purshouse et al., eds (2013). *Evolutionary Multi-Criterion Optimization (EMO-13) Conference Proceedings*, LNCS 7811. Springer, Berlin.
- Takahashi et al., eds (2011). *Evolutionary Multi-Criterion Optimization (EMO-11) Conference Proceedings*, LNCS 6576. Springer, Berlin.
- Ehrgott et al., eds (2009). *Evolutionary Multi-Criterion Optimization (EMO-09) Conference Proceedings*, LNCS 5467. Springer, Berlin.

- Obayashi et al., eds (2007). *Evolutionary Multi-Criterion Optimization (EMO-07) Conference Proceedings*, LNCS 4403. Springer, Berlin.
- Coello et al., eds (2005). *Evolutionary Multi-Criterion Optimization (EMO-05) Conference Proceedings*, LNCS 3410. Springer, Berlin.
- Fonseca et al., eds (2003). *Evolutionary Multi-Criterion Optimization (EMO-03) Conference Proceedings*. LNCS 2632. Springer, Berlin.
- Zitzler et al., eds (2001). *Evolutionary Multi-Criterion Optimization (EMO-01) Conference Proceedings*. LNCS 1993. Springer, Berlin.

Additionally,

- GECCO (Springer LNCS) and CEC (IEEE Press) annual conference proceedings feature numerous research papers on EMO theory, implementation, and applications.
- MCDM conference proceedings (Springer) publish theory, implementation, and application papers in the area of classical multi-objective optimization.

Mailing Lists

- emo-list@ualg.pt (EMO methodologies)
- http://lists.jyu.fi/mailman/listinfo/mcdm-discussion (MCDM related queries)

Public-Domain Source Codes

- NSGA-II in C: http://www.iitk.ac.in/kangal/soft.htm
- SPEA2 in C++: http://www.tik.ee.ethz.ch/~zitzler
- MOEA/D in C++: http://dces.essex.ac.uk/staff/zhang/webofmoead.htm
- Other codes: http://www.lania.mx/~ccoello/EMOO/
- MCDM softwares: http://www.mit.jyu.fi/MCDM/soft.html
- JMetal software: http://jmetal.sourceforge.net/

References

Babu B, Jehan ML (2003) Differential evolution for multi-objective optimization. In: Proceedings of the CEC'2003, Canberra, vol 4. IEEE, Piscataway, pp 2696–2703

Bader J, Deb K, Zitzler E (2010) Faster hypervolume-based search using Monte Carlo sampling. In: Proceedings of the MCDM 2008, Auckland. LNEMS 634. Springer, Heidelberg, pp 313–326

Bagchi T (1999) Multiobjective scheduling by genetic algorithms. Kluwer, Boston

Balicki J, Kitowski Z (2001) Multicriteria evolutionary algorithm with tabu search for task assignment. In: Proceedings of the EMO-01, Zurich, pp 373–384

Bandaru S, Deb K (2010) Automated discovery of vital knowledge from pareto-optimal solutions: first results from engineering design. In: Proceedings of the WCCI-2010, Barcelona. IEEE, Piscataway

Bandaru S, Deb K (2011a) Automated innovization for simultaneous discovery of multiple rules in bi-objective problems. In: Proceedings of the EMO-2011, Ouro Preto. Springer, Heidelberg, pp 1–15

Bandaru S, Deb K (2011b) Towards automating the discovery of certain innovative design principles through a clustering based optimization technique. Eng Optim 43:911–941

Bandyopadhyay S, Saha S, Maulik U, Deb K (2008) A simulated annealing-based multiobjective optimization algorithm: Amosa. IEEE Trans Evol Comput 12:269–283

Belton V, Stewart TJ (2002) Multiple criteria decision analysis: an integrated approach. Kluwer, Boston

Bleuler S, Brack M, Zitzler E (2001) Multiobjective genetic programming: reducing bloat using SPEA2. In: Proceedings of the CEC-2001, Seoul, pp 536–543

Bradstreet L, While L, Barone L (2008) A fast incremental hypervolume algorithm. IEEE Trans Evol Comput 12:714–723

Branke J (2001) Evolutionary optimization in dynamic environments. Springer, Heidelberg

Branke J, Greco S, Slowinski R, Zielniewicz P (2009) Interactive evolutionary multiobjective optimization using robust ordinal regression. In: Proceedings of the EMO-09, Nantes. Springer, Berlin, pp 554–568

Brockhoff D, Zitzler E (2006) Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization. In: PPSN IX, Reykjavik. LNCS 4193, pp 533–542

Brockhoff D, Zitzler E (2007) Dimensionality reduction in multiobjective optimization: the minimum objective subset problem. In: Waldmann KH, Stocker UM (eds) OR proceedings 2006, Karlsruhe, Germany. Springer, Berlin, pp 423–429

Chankong V, Haimes YY (1983) Multiobjective decision making theory and methodology. North-Holland, New York

Chattopadhyay A, Seeley C (1994) A simulated annealing technique for multiobjective optimization of intelligent structures. Smart Mater Struct 3:98–106

Coello CAC (2000) Treating objectives as constraints for single objective optimization. Eng Optim 32:275–308

Coello CAC (2003) http://www.lania.mx/~ccoello/EMOO/

Coello CAC, Lamont GB (2004) Applications of multi-objective evolutionary algorithms. World Scientific, Singapore

Coello CAC, Lechuga MS (2002) MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the CEC 2002, vol 2. IEEE, Piscataway, Honolulu, USA, pp. 1051–1056

Coello CAC, Toscano G (2000) A micro-genetic algorithm for multi-objective optimization. Technical report Lania-RI-2000–06, Laboratoria Nacional de Informatica Avanzada, Xalapa, Veracruz

Coello CAC, Van Veldhuizen DA, Lamont G (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer, Boston

Coello CAC, Aguirre AH, Zitzler E (eds) (2005) Evolutionary multi-criterion optimization (EMO-2005). LNCS 3410. Springer, Berlin

Collette Y, Siarry P (2004) Multiobjective optimization: principles and case studies. Springer, Berlin

Cormen TH, Leiserson CE, Rivest RL (1990) Introduction to algorithms. Prentice-Hall, New Delhi

Corne DW, Knowles JD (2007) Techniques for highly multiobjective optimization: some nondominated points are better than others. In: Proceedings of the GECCO-07, London. ACM, New York, pp 773–780

Corne DW, Knowles JD, Oates M (2000) The Pareto envelope-based selection algorithm for multiobjective optimization. In: Proceedings of the PPSN-VI, Paris, pp 839–848

Coverstone-Carroll V, Hartmann JW, Mason WJ (2000) Optimal multi-objective low-thurst spacecraft trajectories. Comput Methods Appl Mech Eng 186:387–402

Deb K (1995) Optimization for engineering design: algorithms and examples. Prentice-Hall, New Delhi

Deb K (1999) Solving goal programming problems using multi-objective genetic algorithms. In: Proceedings of the CEC, Washington, pp 77–84

Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester

Deb K (2003) Unveiling innovative design principles by means of multiple conflicting objectives. Eng Optim 35:445–470

Deb K, Datta R (2010) A fast and accurate solution of constrained optimization problems using a hybrid bi-objective and penalty function approach. In: Proceedings of the IEEE WCCI 2010, Barcelona, pp 165–172

Deb K, Jain S (2002) Running performance metrics for evolutionary multi-objective optimization. In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (SEAL-02), Singapore, pp 13–20

Deb K, Jain S (2003) Multi-speed gearbox design using multi-objective evolutionary algorithms. ASME Trans Mech Des 125:609–619

Deb K, Jain H (2012) Handling many-objective problems using an improved NSGA-II procedure. In: Proceedings of the CEC 2012, Brisbane

Deb K, Kumar A (2007a) Interactive evolutionary multi-objective optimization and decision-making using reference direction method. In: Proceedings of the GECCO 2007, London. ACM, New York, pp 781–788

Deb K, Kumar A (2007b) Light beam search based multi-objective optimization using evolutionary algorithms. In: Proceedings of the CEC-07, Singapore, pp 2125–2132

Deb K, Saha A (2012) Multimodal optimization using a bi-objective evolutionary algorithms. Evol Comput J 20:27–62

Deb K, Saxena D (2006) Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: Proceedings of the WCCI 2006, Vancouver, pp 3352–3360

Deb K, Srinivasan A (2006) Innovization: innovating design principles through optimization. In: Proceedings of the GECCO-2006, Seattle. ACM, New York, pp 1629–1636

Deb K, Tiwari S (2004) Multi-objective optimization of a leg mechanism using genetic algorithms. Technical report KanGAL 2004005, Kanpur Genetic Algorithms Laboratory (KanGAL), IIT, Kanpur

Deb K, Agrawal S, Pratap A, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197

Deb K, Zope P, Jain A (2003a) Distributed computing of pareto-optimal solutions using multi-objective evolutionary algorithms. In: Proceedings of the EMO-03, Faro. LNCS 2632, pp 535–549

Deb K, Mohan M, Mishra S (2003b) Towards a quick computation of well-spread pareto-optimal solutions. In: Proceedings of the EMO-03, Faro. LNCS 2632, pp 222–236

Deb K, Jain P, Gupta N, Maji H (2004a) Multi-objective placement of electronic components using evolutionary algorithms. IEEE Trans Compon Packag Technol 27:480–492

Deb K, Mitra K, Dewri R, Majumdar S (2004b) Towards a better understanding of the epoxy polymerization process using multi-objective evolutionary computation. Chem Eng Sci 59:4261–4277

Deb K, Thiele L, Laumanns M, Zitzler E (2005) Scalable test problems for evolutionary multi-objective optimization. In: Abraham A et al (eds) Evolutionary multiobjective optimization. Springer, London, pp 105–145

Deb K, Sundar J, Uday N, Chaudhuri S (2006) Reference point based multi-objective optimization using evolutionary algorithms. Int J Comput Intell Res (IJCIR) 2:273–286

Deb K, Rao UB, Karthik S (2007) Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling bi-objective optimization problems. In: Proceedings of the EMO-2007, Matsushima

Deb K, Sinha A, Korhonen P, Wallenius J (2010) An interactive evolutionary multi-objective optimization method based on progressively approximated value functions. IEEE Trans Evol Comput 14:723–739

Ehrgott M (2000) Multicriteria optimization. Springer, Berlin

Ehrgott M, Fonseca CM, Gandibleux X, Hao JK, Sevaux M (eds) (2009) Proceedings of the EMO-2009, Nantes. LNCS 5467. Springer, Heidelberg

Fonseca C, Fleming P, Zitzler E, Deb K, Thiele L (eds) (2003) Proceedings of the EMO-2003, Faro. LNCS 2632. Springer, Heidelberg

Giel O (2003) Expected runtimes of a simple multi-objective evolutionary algorithm. In: Proceedings of the CEC-2003, Canberra. IEEE, Piscatway, pp 1918–1925

Goh CK, Tan KC (2009) Evolutionary multi-objective optimization in uncertain environments: issues and algorithms. Springer, Berlin

Goldberg DE (1989) Genetic algorithms for search, optimization, and machine learning. Addison-Wesley, Reading

Gravel M, Price WL, Gagné C (2002) Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. Eur J Oper Res 143:218–229

Haimes YY, Lasdon LS, Wismer DA (1971) On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Trans Syst Man Cybern 1:296–297

Hansen MP (1997) Tabu search in multiobjective optimization: MOTS. Paper presented at MCDM'97, University of Cape Town

Huband S, Barone L, While L, Hingston P (2005) A scalable multi-objective test problem toolkit. In: Proceedings of the EMO-2005, Guanajuato. Springer, Berlin

Hughes EJ (2005) Evolutionary many-objective optimization: many once or one many? In: Proceedings of the CEC-2005, Edinburgh, pp 222–227

Ishibuchi H, Tsukamoto N, Nojima Y (2008) Evolutionary many-objective optimization: a short review. In: Proceedings of the CEC-2008, Hong Kong, pp 2424–2431

Jensen MT (2003a) Guiding single-objective optimization using multi-objective methods. In: Raidl G et al (eds) Applications of evolutionary computing. Evoworkshops 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM, Essex. LNCS 2611. Springer, Berlin, pp 199–210

Jensen MT (2003b) Reducing the run-time complexity of multiobjective EAs. IEEE Trans Evol Comput 7:503–515

Khor EF, Tan KC, Lee TH (2001) Tabu-based exploratory evolutionary algorithm for effective multi-objective optimization. In: Proceedings of the EMO-01, Zurich, pp 344–358

Knowles JD, Corne DW (2000) Approximating the non-dominated front using the Pareto archived evolution strategy. Evol Comput J 8:149–172

Knowles J, Corne D (2007) Quantifying the effects of objective space dimension in evolutionary multiobjective optimization. In: Proceedings of the EMO-2007, Matsushima. LNCS 4403, pp 757–771

Knowles JD, Corne DW, Deb K (2008) Multiobjective problem solving from nature. Natural computing series. Springer, Berlin

Kumral M (2003) Application of chance-constrained programming based on multi-objective simulated annealing to solve a mineral blending problem. Eng Optim 35:661–673

Kung HT, Luccio F, Preparata FP (1975) On finding the maxima of a set of vectors. J Assoc Comput Mach 22:469–476

Laumanns M, Thiele L, Deb K, Zitzler E (2002a) Combining convergence and diversity in evolutionary multi-objective optimization. Evol Comput 10:263–282

Laumanns M, Thiele L, Zitzler E, Welzl E, Deb K (2002b) Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In: Proceedings of the PPSN-VII, Granada, pp 44–53

Laumanns M, Thiele L, Zitzler E (2004) Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. IEEE Trans Evol Comput 8:170–182

López JA, Coello Coello CA (2009) Some techniques to deal with many-objective problems. In: Proceedings of the 11th annual conference companion on genetic and evolutionary computation, Montreal. ACM, New York, pp 2693–2696

Loughlin DH, Ranjithan S (1997) The neighborhood constraint method: a multiobjective optimization technique. In: Proceedings of the 7th international conference on genetic algorithms, East Lansing, pp 666–673

McMullen PR (2001) An ant colony optimization approach to addessing a JIT sequencing problem with multiple objectives. Artif Intell Eng 15:309–317

Miettinen K (1999) Nonlinear multiobjective optimization. Kluwer, Boston

Mostaghim S, Teich J (2003) Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: Proceedings of the 2003 IEEE symposium on swarm intelligence, Indianapolis. IEEE, Piscataway, pp 26–33

Obayashi S, Deb K, Poloni C, Hiroyasu T, Murata T (eds) (2007) Proceedings of the EMO-2007, Matsushima. LNCS 4403. Springer, Berlin

Osyczka A (2002) Evolutionary algorithms for single and multicriteria design optimization. Physica-Verlag, Heidelberg

Parks G, Suppapitnarm A (1999) Multiobjective optimization of PWR reload core designs using simulated annealing. In: Aragonès JM (eds) Mathematics and computation, reactor physics and environmental analysis in nuclear applications, vol 2. Senda Editorial, Madrid, pp 1435–1444

Rudolph G (1998) Evolutionary search for minimal elements in partially ordered finite sets. In: Proceedings of the 7th annual conference on evolutionary programming, San Diego. Springer, Berlin, pp 345–353

Rudolph G, Agapie A (2000) Convergence properties of some multi-objective evolutionary algorithms. In: Proceedings of the CEC 2000, San Diego, pp 1010–1016

Saxena D, Duro JA, Tiwari A, Deb K, Zhang Q (2013) Objective reduction in many-objective optimization: linear and nonlinear algorithms. IEEE Trans Evol Comput 17(1):77–99

Sharma D, Collet P (2010) GPGPU compatible archive based stochastic ranking evolutionary algorithm (G-ASREA) for multi-objective optimization. In: Proceedings of the PPSN-2010, Kraków. Springer, Berlin, pp 111–120

Srinivas N, Deb K (1994) Multi-objective function optimization using non-dominated sorting genetic algorithms. Evol Comput J 2:221–248

Surry PD, Radcliffe NJ, Boyd ID (1995) A multi-objective approach to constrained optimization of gas supply networks: the COMOGA method. In: Evolutionary computing. AISB workshop, Sheffield. Springer, Berlin, pp 166–180

Takahashi RHC, Deb K, Wanner EF, Greco S (2011) Proceedings of the EMO-2011, Ouro Preto. LNCS 6576. Springer, Berlin

Tzeng GH, Huang J-J (2011) Multiple attribute decision making: methods and applications. CRC, Boca Raton

Veldhuizen DV, Lamont GB (2000) Multiobjective evolutionary algorithms: analyzing the state-of-the-art. Evol Comput J 8:125–148

While L, Hingston P, Barone L, Huband S (2006) A faster algorithm for calculating hypervolume. IEEE Trans Evol Comput 10:29–38

Wong ML (2009) Parallel multi-objective evolutionary algorithms on graphics processing units. In: Proceedings of the GECCO-2009, Montreal, pp 2515–2522

Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11:712–731

Zhang Q, Zhou A, Zhao SZ, Suganthan PN, Liu W, Tiwari S (2008) Multiobjective optimization test instances for the CEC-2009 special session and competition. Technical report, Nanyang Technological University, Singapore

Zitzler E (1999) Evolutionary agorithms for multiobjective optimization: methods and applications. PhD thesis, Swiss Federal Institute of Technology ETH, Zürich

Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: Proceedings of the PPSN VIII, Birmingham. LNCS 3242. Springer, Berlin, pp 832–842

Zitzler E, Thiele L (1998) An evolutionary algorithm for multiobjective optimization: the strength Pareto approach. Technical report 43, Computer Engineering and Networks Laboratory, Switzerland

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 3:257–271

Zitzler E, Deb K, Thiele L, Coello CAC, Corne DW (2001a) Proceedings of the EMO-2001, Zurich. LNCS 1993. Springer, Berlin

Zitzler E, Laumanns M, Thiele L (2001b) SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou KC, Tsahalis DT, Périaux J, Papailiou KD, Fogarty T (eds) Evolutionary methods for design optimization and control with applications to industrial problems, Athens. International Center for Numerical Methods in Engineering (CIMNE), pp 95–100

Zitzler E, Thiele L, Laumanns M, Fonseca CM, da Fonseca VG (2003) Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evol Comput 7:117–132

Zitzler E, Thiele L, Bader J (2010) On set-based multiobjective optimization. IEEE Trans Evol Comput 14:58–79