1 Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design

KALYANMOY DEB[†]
Kanpur Genetic Algorithms Laboratory (KanGAL)
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
E-mail: deb@iitk.ernet.in

1.1 INTRODUCTION

Many real-world engineering design or decision making problems involve simultaneous optimization of multiple objectives. The principle of multi-criterion optimization is different from that in a single-objective optimization. In single-objective optimization, the goal is to find the *best* design solution, which corresponds to the minimum or maximum value of the objective function [9, 34]. On the contrary, in a multi-criterion optimization with *conflicting* objectives, there is no single optimal solution. The interaction among different objectives gives rise to a set of compromised solutions, largely known as the Pareto-optimal solutions [40, 2]. Since none of these Pareto-optimal solutions can be identified as better than others without any further consideration, the goal in a multi-criterion optimization is to find as many Pareto-optimal solutions as possible. Once such solutions are found, it usually requires a higher-level decision-making with other considerations to choose one of them for implementation. Here, we address the first task of finding multiple Pareto-optimal solutions. There are two objectives in a multi-criterion optimization: (i) find solutions close to the true Pareto-optimal solutions and (ii) find solutions that are widely different from each other. The first task is desired to satisfy optimality conditions in the obtained solutions. The second task is desired to have no bias towards any particular objective function.

[†]Currently at the Computer Science Department, University of Dortmund, Germany

In dealing with multi-criterion optimization problems, classical search and optimization methods are not efficient, simply because (i) most of them cannot find multiple solutions in a single run, thereby requiring them to be applied as many times as the number of desired Pareto-optimal solutions, (ii) multiple application of these methods do not guarantee finding widely different Pareto-optimal solutions, and (iii) most of them cannot efficiently handle problems with discrete variables and problems having multiple optimal solutions [9]. On the contrary, the studies on evolutionary search algorithms, over the past few years, have shown that these methods can be efficiently used to eliminate most of the above difficulties of classical methods [38, 19, 26]. Since they use a population of solutions in their search, multiple Pareto-optimal solutions can, in principle, be found in one single run. The use of diversity-preserving mechanisms can be added to the evolutionary search algorithms to find widely different Pareto-optimal solutions.

In this paper, we briefly outline the principles of multi-objective optimization. Thereafter, we discuss why classical search and optimization methods are not adequate for multi-criterion optimization by discussing the working of two popular methods. We then outline several evolutionary methods for handling multi-criterion optimization problems. Of them, we discuss one implementation (non-dominated sorting GA or NSGA [38]) in somewhat greater details. Thereafter, we demonstrate the working of the evolutionary methods by applying NSGA to three test problems having constraints and discontinuous Pareto-optimal region. We also show the efficacy of evolutionary algorithms in engineering design problems by solving a welded beam design problem. The results show that evolutionary methods can find widely different yet near-Pareto-optimal solutions in such problems. Based on the above studies, this paper also suggests a number of immediate future studies which would make this emerging field more mature and applicable in practice.

1.2 PRINCIPLES OF MULTI-CRITERION OPTIMIZATION

The principles of multi-criterion optimization are different from that in a single-objective optimization. The main goal in a single-objective optimization is to find the global optimal solution, resulting in the optimal value for the single objective function. However, in a multi-criterion optimization problem, there are more than one objective function, each of which may have a *different* individual optimal solution. If there is sufficient difference in the optimal solutions corresponding to different objectives, the objective functions are often known as *conflicting* to each other. Multi-criterion optimization with such conflicting objective functions gives rise to a set of optimal solutions, instead of one optimal solution. The reason for the optimality of many solutions is that no one can be considered to be better than any other with respect to all objective functions. These optimal solutions have a special name—Pareto-optimal solutions. Let us illustrate this aspect with a hypothetical example problem shown in Figure 1.1. The figure considers two objectives—cost and

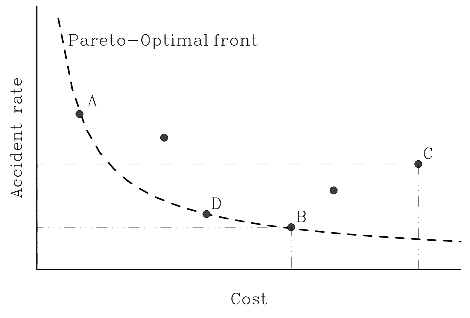

Fig. 1.1 The concept of Pareto-optimal solutions is illustrated.

accident rate—both of which are to be minimized. The point A represents a solution which incurs a near-minimal cost, but is highly accident-prone. On the other hand, the point B represents a solution which is costly, but is near least accident-prone. If both objectives (cost and accident rate) are important goals of design, one cannot really say whether solution A is better than solution B, or vice versa. One solution is better than other in one objective, but is worse in the other. In fact, there exist many such solutions (like solution D) which also belongs to the Pareto-optimal set and one cannot conclude about an absolute hierarchy of solutions A, B, D, or any other solution in the set. All these solutions (in the front marked by the dashed line) are known as Pareto-optimal solutions.

Looking at the figure, we also observe that there exists non-Pareto-optimal solutions, like the point C. If we compare solution C with solution A, we again are in a fix and cannot say whether one is better than the other in both objectives. Does this mean that solution C is also a member of the Pareto-optimal set. The answer is no. This is because there exists another solution D in the search space, which is better than solution C in *both* objectives. That is why solutions like C are known as *dominated* solutions or *inferior* solutions.

It is now clear that the concept of optimality in multi-criterion optimization deals with a number (or a set) of solutions, instead of one solution. Based on the above discussions, we first define conditions for a solution to become dominated with respect to another solution and then present conditions for a set of solutions to become a Pareto-optimal set.

For a problem having more than one objective function (say, f_j, $j = 1,\ldots,M$ and $M > 1$), any two solutions $x^{(1)}$ and $x^{(2)}$ can have one of two possibilities—one dominates the other or none dominates the other. A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$, if both the following conditions are true:

1. The solution $x^{(1)}$ is no worse (say the operator $\prec$ denotes worse and $\succ$ denotes better) than $x^{(2)}$ in all objectives, or $f_j(x^{(1)}) \not\prec f_j(x^{(2)})$ for all $j = 1, 2, \ldots, M$ objectives.

2. The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective, or $f_{\bar{j}}(x^{(1)}) \succ f_{\bar{j}}(x^{(2)})$ for at least one $\bar{j} \in \{1, 2, \ldots, M\}$.

If any of the above condition is violated, the solution $x^{(1)}$ does not dominate the solution $x^{(2)}$. If $x^{(1)}$ dominates the solution $x^{(2)}$, it is also customary to write $x^{(2)}$ is dominated by $x^{(1)}$, or $x^{(1)}$ is non-dominated by $x^{(2)}$, or, simply, among the two solutions, $x^{(1)}$ is the non-dominated solution.

The following definitions ensure whether a set of solutions belong to a local or global Pareto-optimal set, similar to the definitions of local and global optimal solutions in single-objective optimization problems:

Local Pareto-optimal Set: If for every member x in a set $\underline{P}$, there exist no solution y satisfying $||y - x||_\infty \leq \epsilon$, where ϵ is a small positive number (in principle, y is obtained by perturbing x in a small neighborhood), which dominates any member in the set $\underline{P}$, then the solutions belonging to the set $\underline{P}$ constitute a local Pareto-optimal set.

Global Pareto-optimal Set: If there exists no solution in the search space which dominates any member in the set $\bar{P}$, then the solutions belonging to the set $\bar{P}$ constitute a global Pareto-optimal set.

We would like to highlight here that there exists a difference between a non-dominated set and a Pareto-optimal set. A non-dominated set is defined in the context of a sample of the search space. In a sample of search points, solutions that are not dominated (according to the above definition) by any other solutions in the sample space are non-dominated solutions. A Pareto-optimal set is a non-dominated set, when the sample is the entire search space.

From the above discussions, we observe that there are primarily two goals that a multi-criterion optimization algorithm must achieve:

1. Guide the search towards the global Pareto-optimal region, and

2. Maintain population diversity in the Pareto-optimal front.

The first task is a natural goal of any optimization algorithm. The second task is unique to multi-criterion optimization[1]. Since no one solution in the

[1] In multi-modal optimization problems, often, the goal is also to find multiple global optimal solutions simultaneously [14, 22].

Pareto-optimal set can be said to be better than the other, what an algorithm can do best is to find as many different Pareto-optimal solutions as possible.

We now review a couple of popular classical search and optimization methods briefly and discuss why there is a need for better algorithms for multi-criterion optimization.

1.3 CLASSICAL METHODS

Here, we shall discuss two popular classical optimization methods used for solving multi-criterion optimization problems.

1.3.1 Weighted Sum Method

Multiple objective functions are combined into one overall objective function, F, as follows:

$$\begin{array}{ll} \text{Minimize} & F = \sum_{j=1}^{M} w_j f_j(\vec{x}), \\ & \vec{x} \in \mathcal{F}, \end{array} \tag{1.1}$$

where w_j is the weight used for the j-th objective function $f_j(\vec{x})$. Usually, non-zero fractional weights are used so that sum of all weights $\sum_{j=1}^{M} w_j$ is equal to one. All Pareto-optimal solutions must lie in the feasible region $\mathcal{F}$. The procedure is simple. Choose a random weight vector and optimize the single-objective function F to get an optimal solution. Hopefully, the obtained optimal solution belongs to the set of the desired Pareto-optimal set. In order to get different Pareto-optimal solutions, choose different random vectors and optimize the resulting F in each case. We illustrate the working of this method in a hypothetical problem shown in Figure 1.2. The figure shows the feasible search space in the function space, having two objectives (which are to be minimized). Each point inside the feasible region represents a solution ($\vec{x}$) having two objective function values (such as cost and accident rate). Fixing a weight vector and optimizing equation 1.1 means finding a hyper-plane (a line for two objective functions) with a fixed orientation in the function space. The optimal solution is the point where a hyperplane having this fixed orientation has a common tangent with the feasible search space. We show this solution as the point A in the figure, where the line with intercepts at the f_1 and f_2 axes in proportions of w_2 and w_1, respectively, is tangent to the feasible region. One can now imagine finding other solutions such as B or C by choosing a different weight vector and finding the corresponding common tangent point again. A collection of such solutions (A, B, C, and others) constitute the Pareto-optimal set (shown by a thick line). Although such a simple strategy is intuitively a computationally expensive method, there is a major difficulty with such a method.

This weighted sum method cannot be used to find Pareto-optimal solutions in multi-criterion optimization problems having a non-convex Pareto-optimal

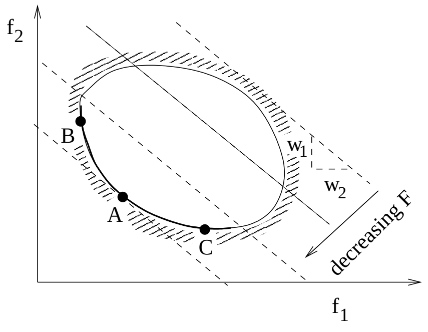

Fig. 1.2 A convex Pareto-optimal front is shown

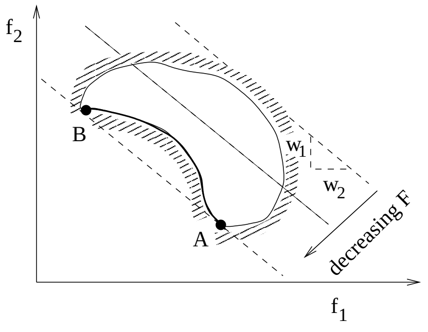

Fig. 1.3 A non-convex Pareto-optimal front is shown

front. In Figure 1.2, the Pareto-optimal front is convex. But, one can also think of multi-criterion optimization problems having a non-convex Pareto-optimal front (Figure 1.3). In this figure, fixing a different weight vector and finding the tangents closest to the origin do not give rise to finding different Pareto-optimal solutions. For every weight vector, only the solutions A or B will be found, and all other Pareto-optimal solutions within A and B cannot be found.

1.3.2 The ϵ-perturbation method

In order to remedy the above difficulty, a single-objective optimization problem is constructed in which all but one objectives are used as constraints and only one is used as the objective function:

$$\begin{array}{rl} \text{Minimize} & f_k(\vec{x}), \\ \text{Subject to} & f_j(\vec{x}) \le \epsilon_j \quad \forall j \neq k, \\ & \vec{x} \in \mathcal{F}. \end{array} \tag{1.2}$$

To find one Pareto-optimal solution, a suitable value of ϵ_j is chosen for the j-th objective function (where $j \neq k$). Thereafter, the above single-objective constraint optimization problem is solved to find the solution A. This procedure is continued with different values of ϵ_j to find different Pareto-optimal solutions. Figure 1.4 shows that this method can find non-convex Pareto-optimal solutions. However, there also exists a difficulty with this method. A knowledge of an appropriate range of ϵ_j values for each objective function is required to be known a priori.

Although there exists a few other methods such as goal programming, min-max method, and others [40], all methods require some kind of problem knowledge. The most profound difficulty with all these methods is that all need to be applied many times, hopefully finding one different Pareto-optimal

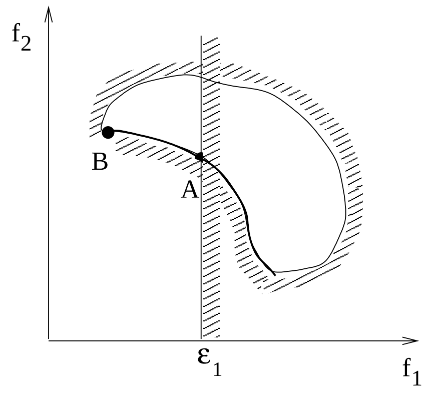

Fig. 1.4 The ϵ-perturbation method is illustrated. The first objective function is used as a constraint $f_1(\vec{x}) \leq \epsilon_1$.

solution each time. This makes the methods unattractive and this is one of the reasons why multi-criterion optimization problems are mostly avoided in practice. Based on the above discussions, we summarize the difficulties with the classical optimization methods:

1. An algorithm is needed to be applied many times to find multiple Pareto-optimal solutions,

2. Most algorithms demand some knowledge about the problem being solved,

3. Some algorithms are sensitive to the shape of the Pareto-optimal front,

4. The spread of Pareto-optimal solutions depends on efficiency of the single-objective optimizer,

5. In problems involving uncertainties or stochasticities, classical methods are not reliable,

6. Since classical single-objective optimizers are not efficient in handling discrete search space problems [9, 13], they will also not be efficient for multi-criterion optimization problems having discrete search space.

The first difficulty can be eliminated if an appropriate population-based search algorithm is used and special operators are used to emphasize maintenance of multiple Pareto-optimal solutions in the population. However, all the above difficulties can be handled by using an evolutionary search algorithm. In the following, we briefly describe a number of evolutionary-based implementations and describe one such method in details.

1.4 EVOLUTIONARY METHODS

As early as in 1967, Rosenberg suggested, but did not simulate, a genetic search method for finding the chemistry of a population of single-celled organisms with multiple properties or objectives [35]. However, the first practical implementation was suggested by David Schaffer in the year 1984 [36]. Thereafter, no significant study was performed for almost a decade, except a revolutionary 10-line sketch of a new non-dominated sorting procedure outlined in David Goldberg's book [20]. The book came out in the year 1989. Getting a clue for an efficient multi-objective optimization technique, many researchers developed different versions of multi-objective optimization algorithms [38, 19, 26] based on their interpretations of the 10-line sketch. The idea was so sound and appropriate, that almost any such implementation has been successful in many test problems. The publication of the above-mentioned algorithms showed the superiority of evolutionary multi-criterion optimization techniques over classical methods, and since then there has been no looking back. Many researchers have modified the above-mentioned approaches and developed their own versions. Many researchers have also applied these techniques to more complex test problems and to real-world engineering design problems. Till to date, most of the successful evolutionary implementations for multi-criterion optimization rely on the concept of non-domination. Although there are other concepts that can be used to develop a search algorithm [28], the concept of non-domination is simple to use and understand. However, a recent study [7] has shown that search algorithms based on non-domination need not always lead to the true Pareto-optimal front. The algorithm can get stuck to a non-dominated front which is different from the true Pareto-optimal front. This exception is in the details of the implementational issues and we would not like to belabor this here, simply because in most test problems tried so far most of the search algorithms based on non-domination concept has found the true Pareto-optimal front.

With the development of many new algorithms, many researchers have attempted to summarize the studies in the field from different perspectives [18, 25, 42, 3]. These reviews list many different techniques of multi-criterion optimization that exist to date.

We now present a brief summary of a few salient evolutionary multi-criterion optimization algorithms.

1.4.1 Schaffer's VEGA

Schaffer [36] modified the simple tripartite genetic algorithm by performing independent selection cycles according to each objective. He modified the public-domain GENESIS software by creating a loop around the traditional selection procedure so that the selection method is repeated for each individual objective to fill up a portion of the mating pool. Then the entire population is thoroughly shuffled to apply crossover and mutation operators. This is performed to achieve the mating of individuals of different subpopulation groups.

The algorithm worked efficiently for some generations but in some cases suffered from its bias towards some individuals or regions. The independent selection of specialists resulted in speciation in the population. The outcome of this effect is the convergence of the entire population towards the individual optimum regions after a large number of generations. From a designer's point of view, it is not desirable to have any bias towards such middling individuals, rather it is of interest to find as many non-dominated solutions as possible. Schaffer tried to minimize this speciation by developing two heuristics—the non-dominated selection heuristic (a wealth redistribution scheme), and the mate selection heuristic (a cross breeding scheme) [36, 37]. In the non-dominated selection heuristic, dominated individuals are penalized by subtracting a small fixed penalty from their expected number of copies during selection. Then the total penalty for dominated individuals was divided among the non-dominated individuals and was added to their expected number of copies during selection. But this algorithm failed when the population has very few non-dominated individuals, resulting in a large fitness value for those few non-dominated points, eventually leading to a high selection pressure. The mate selection heuristic was intended to promote the cross breeding of specialists from different subgroups. This was implemented by selecting an individual, as a mate to a randomly selected individual, which has the maximum Euclidean distance in the performance space from its mate. But it failed too to prevent the participation of poorer individuals in the mate selection. This is because of random selection of the first mate and the possibility of a large Euclidean distance between a champion and a mediocre. Schaffer concluded that the random mate selection is far superior than this heuristic.

1.4.2 Fonseca amd Fleming's multi-objective GA

Fonesca and Fleming [19] implemented Goldberg's suggestion in a different way. In this study, the multi-objective optimization GA (MOGA) uses a non-dominated sorting procedure. In MOGA, the whole population is checked and all non-dominated individuals are assigned rank '1'. Other individuals are ranked by checking the non-dominance of them with respect to the rest of the population in the following way. For an individual solution, the number of solutions that strictly dominate it in the population is first found. Thereafter, the rank of that individual is assigned to be one more than that number. Therefore, at the end of this ranking procedure, there may exist many solutions having the same rank. The selection procedure then uses these ranks to select or delete blocks of points to form the mating pool. As discussed elsewhere [21], this type of blocked fitness assignment is likely to produce a large selection pressure which might cause premature convergence. MOGA also uses a niche-formation method to distribute the population over the Pareto-optimal region. But instead of performing sharing on the parameter values, they have used sharing on objective function values. Even though this maintain diversity in the objective function values, this may not maintains diversity in the parameter set, a matter which is important for a decision maker. Moreover, MOGA may not be able to find multiple solutions in problems where different Pareto-optimal points correspond to the same objective function value [7]. However, the ranking of individuals according to their non-dominance in the population is an important aspect of this work.

1.4.3 Horn, Nafploitis, and Goldberg's Niched Pareto GA

Horn, Nafploitis, and Goldberg [26] used a *Pareto domination tournaments* instead of non-dominated sorting and ranking selection method in solving multi-objective optimization problems. In this method, a *comparison set* comprising of a specific number (t_{dom}) of individuals is picked at random from the population at the beginning of each selection process. Two random individuals are picked from the population for selecting a winner in a tournament selection according to the following procedure. Both individuals are compared with the members of the comparison set for domination with respect to objective functions. If one of them is non-dominated and the other is dominated, then the non-dominated point is selected. On the other hand, if both are either non-dominated or dominated, a niche count is found for each individual in the entire population. The niche count is calculated by simply counting the number of points in the population within a certain distance (σ_{share}) from an individual. The individual with least niche count is selected. The effect of multiple objectives is taken into the non-dominance calculation. Since this non-dominance is computed by comparing an individual with a randomly chosen population set of size t_{dom}, the success of this algorithm highly depends on the parameter t_{dom}. If a proper size is not chosen, true non-dominated (Pareto-optimal) points may not be found. If a small t_{dom} is chosen, this may result in a few non-dominated points in the population. Instead, if a large t_{dom} is chosen, premature convergence may result. This aspect is also observed by the authors. They have presented some empirical results with various t_{dom} values. Nevertheless, the concept of niche formation among the non-dominated points is an important aspect of this work.

1.4.4 Zitzler and Theile's strength Pareto approach (SPEA)

Zitzler and Theile [44] have recently suggested an elitist multi-criterion EA with the concept of non-domination. They suggested maintaining an external population at every generation storing a set of non-dominated solutions discovered so far beginning from the initial population. This external population participates in genetic operations. The fitness of each individual in the current population and in the external population is decided based on the number of dominated solutions. Specifically, the following procedure is adopted. A combined population with the external and the current population is first constructed. All non-dominated solutions in the combined population are assigned a fitness based on the number of solutions they dominate. To maintain diversity and in the context of minimizing the fitness function, they assigned more fitness to a non-dominated solution having more dominated solutions in the combined population. On the other hand, more fitness is also assigned to solutions dominated by more solutions in the combined population. Care is taken to assign no non-dominated solution a fitness worse than that of the best dominated solution. This assignment of fitness makes sure that the search is directed towards the non-dominated solutions and simultaneously diversity among dominated and non-dominated solutions are maintained. On knapsack problems, they have reported better results than any other method used for comparison in that study. However, such comparisons of algorithms is not appropriate, simply because SPEA approach uses a inherent elitism mechanism of using best non-dominated solutions discovered up to the current generation, whereas other algorithms do not use any such mechanism. Nevertheless, an interesting aspect of that study is that it shows the importance of introducing elitism in evolutionary multi-criterion optimization. Similar effect of elitism in multi-criterion optimization was also observed elsewhere [30].

1.4.5 Srinivas and Deb's non-dominated sorting genetic algorithm (NSGA)

Srinivas and Deb [38] have implemented Goldberg's idea most directly. The idea behind NSGA is that a ranking selection method is used to emphasize current non-dominated points and a niching method is used to maintain diversity in the population. We describe the NSGA procedure in somewhat more details.

NSGA varies from a simple genetic algorithm only in the way the selection operator in used. The crossover and mutation operators remain as usual. Before the selection is performed, the population is first ranked on the basis of an individual's non-domination level, which is found by the following procedure, and then a fitness is assigned to each population member.

1.4.5.1 Fitness assignment: Consider a set of N population members, each having M (> 1) objective function values. The following procedure can be used to find the non-dominated set of solutions:

Step 0: Begin with $i = 1$.

Step 1: For all $j = 1, \ldots, N$ and $j \neq i$, compare solutions $x^{(i)}$ and $x^{(j)}$ for domination using two conditions (page iv) for all M objectives.

Step 2: If for any j, $x^{(i)}$ is dominated by $x^{(j)}$, mark $x^{(i)}$ as 'dominated'.

Step 3: If all solutions (that is, when $i = N$ is reached) in the set are considered, Go to Step 4, else increment i by one and Go to Step 1.

Step 4: All solutions that are not marked 'dominated' are non-dominated solutions.

All these non-dominated solutions are assumed to constitute the first non-dominated front in the population and assigned a large dummy fitness value (we assign a fitness N). The same fitness value is assigned to give an equal reproductive potential to all these non-dominated individuals. In order to maintain diversity in the population, these non-dominated solutions are then *shared* with their dummy fitness values. Sharing methods are discussed elsewhere in details [11, 22]; we give a brief description of the sharing procedure in the next subsection. Sharing is achieved by dividing the dummy fitness value of an individual by a quantity (called the niche count) proportional to the number of individuals around it. This procedure causes multiple optimal points to co-exist in the population. The worst shared fitness value in the solutions of the first non-dominated front is noted for further use.

After sharing, these non-dominated individuals are ignored temporarily to process the rest of population members. The above step-by-step procedure is used to find the second level of non-dominated solutions in the population. Once they are identified, a dummy fitness value which is a little smaller than the worst shared fitness value observed in solutions of first non-dominated set is assigned. Thereafter, the sharing procedure is performed among the solutions of second non-domination level and shared fitness values are found as before. This process is continued till all population members are assigned a shared fitness value.

The population is then reproduced with the shared fitness values. A stochastic remainder proportionate selection [20] is used in this study. Since individuals in the first front have better fitness values than solutions of any other front, they always get more copies than the rest of population. This was intended to search for non-dominated regions, which will finally lead to the Pareto-optimal front. This results in quick convergence of the population towards non-dominated regions and sharing procedure helps to distribute it over this region.

Another aspect of this method is that practically any number of objectives can be used. Both minimization and maximization problems can also be handled by this algorithm. The only place a change is required for the above two cases is the way the non-dominated points are identified (according to the definition outlined on page iv).

1.4.5.2 Sharing procedure: Given a set of n_k solutions in the k-th non-dominated front each having a dummy fitness value f_k, the sharing procedure is performed in the following way for each solution $i = 1, 2, \ldots, n_k$:

Step 1: Compute a normalized Euclidean distance measure with another solution j in the k-th non-dominated front, as follows:

$$d_{ij} = \sqrt{\sum_{p=1}^{P} \left(\frac{x_p^{(i)} - x_p^{(j)}}{x_p^u - x_p^l} \right)^2},$$

where P is the number of variables in the problem. The parameters x_p^u and x_p^l are the upper and lower bounds of variable x_p.

Step 2: This distance d_{ij} is compared with a pre-specified parameter σ_{share} and the following *sharing function* value is computed [22]:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{\text{share}}} \right)^2, & \text{if } d_{ij} \leq \sigma_{\text{share}}, \\ 0, & \text{otherwise.} \end{cases}$$

Step 3: Increment j. If $j \leq n_k$, go to Step 1 and calculate $Sh(d_{ij})$. If $j > n_k$, calculate niche count for i-th solution as follows:

$$m_i = \sum_{j=1}^{n_k} Sh(d_{ij}).$$

Step 4: Degrade the dummy fitness f_k of i-th solution in the k-th non-domination front to calculate the shared fitness, f_i', as follows:

$$f_i' = \frac{f_k}{m_i}.$$

This procedure is continued for all $i = 1, 2, \ldots, n_k$ and a corresponding f_i' is found. Thereafter, the smallest value $f_k^{\min}$ of all f_i' in the k-th non-dominated front is found for further processing. The dummy fitness of the next non-dominated front is assigned to be $f_{k+1} = f_k^{\min} - \epsilon_k$, where ϵ_k is a small positive number.

The above sharing procedure requires a pre-specified parameter σ_{share}, which can be calculated as follows [11]:

$$\sigma_{\text{share}} \approx \frac{0.5}{\sqrt[P]{q}}, \tag{1.3}$$

where q is the desired number of distinct Pareto-optimal solutions. Although the calculation of σ_{share} depends on this parameter q, it has been been shown elsewhere [7, 38] that the use of above equation with $q \approx 10$ works in many test problems. Moreover, the performance of NSGAs is not very sensitive to this parameter near σ_{share} values calculated using $q \approx 10$.

It may be mentioned here that the above sharing procedure can also be implemented with a distance measure defined in the gene space. That is, instead of the Euclidean distance measure as given above, d_{ij} can also be calculated using the number of bit differences (or the Hamming distance) between solutions i and j [11]. Such distance measures will be useful in problems where the problem is defined in the gene space [7]. The calculation of σ_{share} is different in this case and interested readers may refer to the original study for more details [11].

1.5 PROOF-OF-PRINCIPLE RESULTS

In this section, we apply NSGA to three test problems: (i) a single-variable simple problem [36], (ii) a two-variable constrained problem [2], and (iii) a two-variable problem having discontinuous Pareto-optimal region [7]. In all simulations, we use binary-coded genetic algorithms [20, 24] with a single-point crossover operator with probability one. Mutation probability is kept zero in order to observe the effectiveness of NSGA alone. The parameters are held constant across all runs. Unbiased initial population is generated by randomly spreading solutions over the entire variable space in consideration.

1.5.1 Problem F1

This is a single-variable problem which has been widely used in the multi-criterion optimization literature [36, 38]:

$$\begin{array}{ll} \text{Minimize} & f_1(x) = x^2, \\ \text{Minimize} & f_2(x) = (x-2)^2. \end{array} \tag{1.4}$$

Initial range for the design variable used in simulations is $(-100.0, 100.0)$. The Pareto-optimal solutions lie in $x \in [0, 2]$. The variable is coded using binary strings of size 30. We use a population size[2] of 100. The parameter σ_{share} is set to be 0.0002 here. This value induces about N or 100 niches in the Pareto-optimal region. Figure 1.5 shows the population distribution in the initial generation. The figure shows that the population is widely distributed. With the above parameter settings, it is expected to have only one (out of 100) population member in the Pareto-optimal region. Thereafter, we show the population of 100 members after 10 generations in Figure 1.6. The figure shows how quickly the NSGA has managed to get the population from function values of about four order of magnitudes to one order of magnitude.

[2] Although a much smaller population size will also be able to find Pareto-optimal solutions, this rather large population size is chosen for a specific purpose. This will allow creation of around 100 niches in the Pareto-optimal region, demonstrating that a large number of different niches can be found and maintained in the population using the sharing procedure.

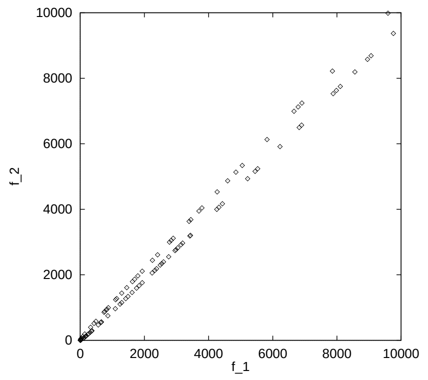

Fig. 1.5 Initial population for function F1.

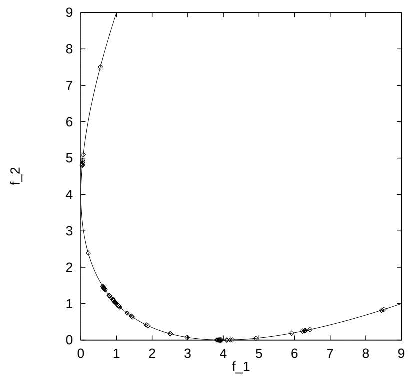

Fig. 1.6 Population at generation 10 for function F1.

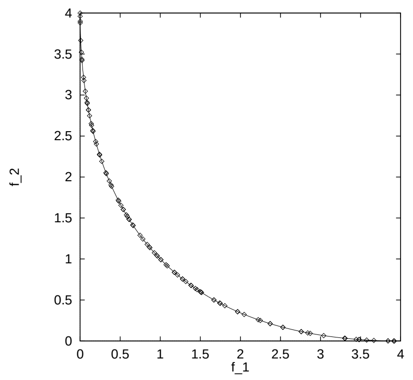

Fig. 1.7 Population at generation 100 for function F1.

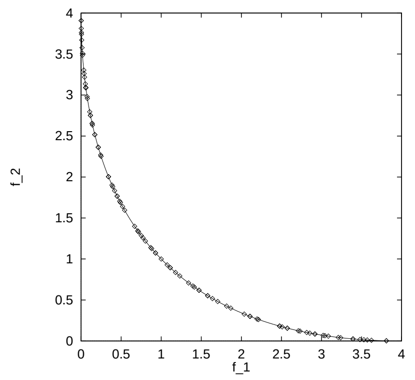

Fig. 1.8 Population at generation 500 for function F1.

However, most of the population members are now inside the Pareto-optimal region. Figure 1.7 shows the population history at generation 100. All population members are now inside the Pareto-optimal region and notice how the population gets uniformly distributed over the Pareto-optimal region. Finally, Figure 1.8 shows the population at generation 500. NSGAs are run so long just to show that they have the capabilities of maintaining a sustainable wide-spread population over the entire Pareto-optimal region.

1.5.2 Problem F2

Next, we consider a two-variable problem with constraints [2, 15]:

$$\begin{array}{rl} \text{Minimize} & f_1(\vec{x}) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2, \\ \text{Minimize} & f_2(\vec{x}) = 9x_1 - (x_2 - 1)^2, \\ \text{Subject to} & g_1(\vec{x}) \equiv 225 - x_1^2 - x_2^2 \geq 0, \\ & g_2(\vec{x}) \equiv 3x_2 - x_1 - 10 \geq 0. \end{array} \tag{1.5}$$

Both variables are initialized in the range $[-20.0, 20.0]$. Both variables are coded using binary strings of size 15. We handle constraints by first normalizing them and then using the bracket-operator penalty function [9] with a penalty parameter 10^3. It was shown in an earlier study [38], that resulting Pareto-optimal region in this problem lies where both functions f_1 and f_2 are tangent to each other. This happens for $x_1 = -2.5$ in the unconstrained problem. The addition of two constraints makes only a portion of the search space feasible. Thus, the resulting Pareto-optimal region is as follows:

$$x_1 = -2.5, \quad 2.5 \leq x_2 \leq 14.79.$$

Figure 1.9 shows the initial population of 100 solutions in the x_1-x_2 space. The figure shows that although some of the solutions are feasible, most solutions are infeasible in the initial population. Figure 1.10 shows the population at generation 10. This figure shows that most population members are now in the feasible region. At generation 20, Figure 1.11 shows that most population members are close to the true Pareto-optimal front. Finally, Figure 1.12 shows that even up to 200 generations the population is able to maintain solutions in the true Pareto-optimal region.

1.5.3 Problem F3

We construct a multi-objective problem having a Pareto-optimal front which is discontinuous [7]:

$$\begin{array}{rl} \text{Minimize} & f_1(\vec{x}) = x_1, \\ \text{Minimize} & f_2(\vec{x}) = (1 + 10x_2)\left(1 - \left(\frac{x_1}{1+10x_2}\right)^2 - \frac{x_1}{1+10x_2}\sin(8\pi x_1)\right). \end{array} \tag{1.6}$$

Both x_1 and x_2 varies in the interval $[0, 1]$ and are coded in binary strings of size 15 each. The discontinuity in the Pareto-optimal region comes due to the periodicity in function f_2. This function tests an algorithm's ability to maintain subpopulations in all discontinuous Pareto-optimal regions.

Figure 1.13 shows the 50,000 random solutions in f_1-f_2 space. The discontinuous Pareto-optimal regions are also shown by showing solid lines. When NSGAs (population size of 200 and σ_{share} of 0.1) are tried on this problem, the resulting population at generation 300 is shown in Figure 1.14. The plot

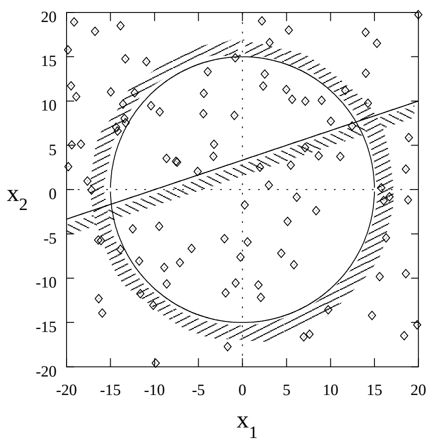

Fig. 1.9 Initial population for function F2.

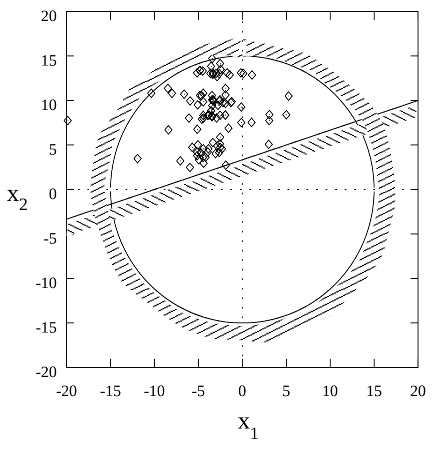

Fig. 1.10 Population at generation 10 for function F2.

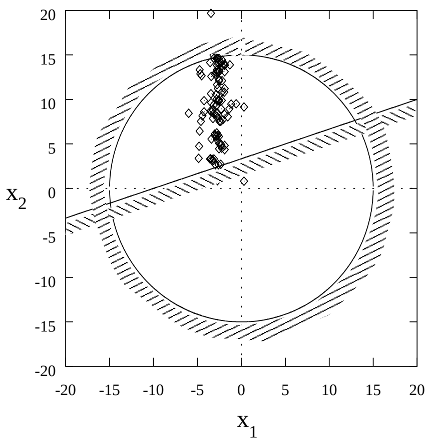

Fig. 1.11 Population at generation 20 for function F2.

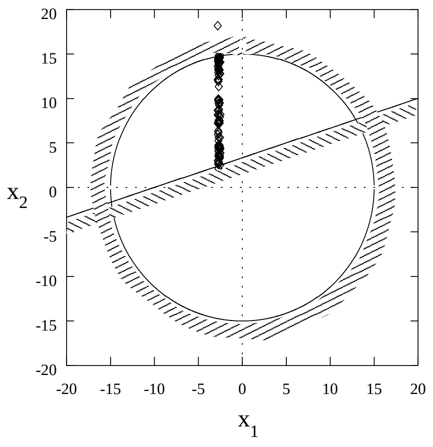

Fig. 1.12 Population at generation 200 for function F2.

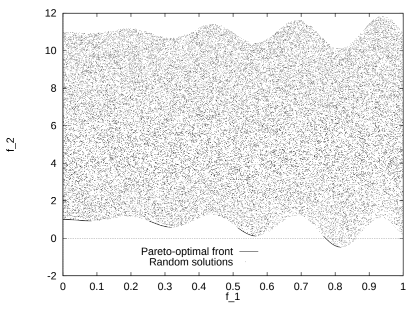

Fig. 1.13 50,000 random solutions are shown on a f_1-f_2 plot of a multi-objective problem having discontinuous Pareto-optimal front.

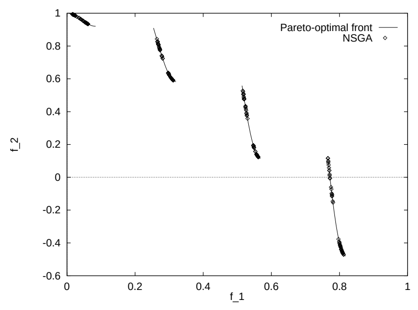

Fig. 1.14 The population at generation 300 for a NSGA run is shown to have found solutions in all four discontinuous Pareto-optimal regions.

shows that if reasonable GA parameter values are chosen, NSGAs can find Pareto-optimal solutions in discontinuous regions. A population size of 200 is deliberately used to have a wide distribution of solutions in all discontinuous regions. A little thought will reveal that Pareto-optimal region comprises of solutions $0 \leq x_1 \leq 1$ and $x_2 = 0$. It is interesting to note how NSGAs avoid creating the non-Pareto-optimal solutions, although the corresponding x_2 value is same (and equal to zero) as that of the Pareto-optimal solutions.

1.6 AN ENGINEERING DESIGN

This problem has been well studied in the context of single-objective optimization [34]. A beam needs to be welded on another beam and must carry a certain load F (Figure 1.15). In the context of single-objective optimal

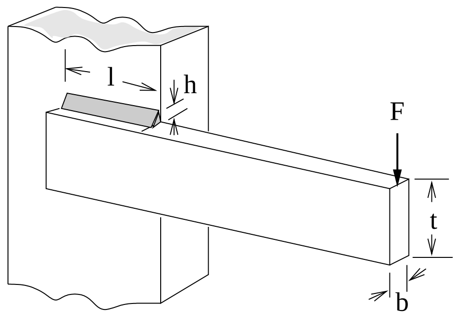

Fig. 1.15 The welded beam design problem. Minimizations of cost and end deflection are two objectives.

design, it is desired to find four design parameters (thickness of the beam, b, width of the beam t, length of weld ℓ, and weld thickness h) for which the cost of the beam is minimum. The overhang portion of the beam has a length of 14 inch and $F = 6,000$ lb force is applied at the end of the beam. It is intuitive that an optimal design for cost will make all four design variables to take small values. When the beam dimensions are small, it is likely that the deflection at the end of the beam is going to be large. In the parlance of mechanics of materials, this means that the rigidity of the beam is smaller for smaller dimensions of the beam. In mechanical design activities, optimal design for maximum rigidity is also common. Again, a little thought will reveal that a design for maximum rigidity of the above beam will make all four design dimensions to take large dimensions. Thus, the design solutions for minimum cost and maximum rigidity (or minimum end deflection) are conflicting to each other. In other words, a design that is near-optimal from cost consideration is not near-optimal from rigidity consideration and vice versa. This kind of conflicting objective functions leads to Pareto-optimal solutions. In the following, we present the mathematical formulation of the two-objective optimization problem of minimizing cost and the end deflection [15, 8]:

$$\begin{array}{rl} \text{Minimize} & f_1(\vec{x}) = 1.10471h^2\ell + 0.04811tb(14.0+\ell), \\ \text{Minimize} & f_2(\vec{x}) = \delta(\vec{x}), \\ \text{Subject to} & g_1(\vec{x}) \equiv 13,600 - \tau(\vec{x}) \geq 0, \\ & g_2(\vec{x}) \equiv 30,000 - \sigma(\vec{x}) \geq 0, \\ & g_3(\vec{x}) \equiv b - h \geq 0, \\ & g_4(\vec{x}) \equiv P_c(\vec{x}) - 6,000 \geq 0. \end{array} \tag{1.7}$$

The deflection term $\delta(\vec{x})$ is given as follows:

$$\delta(\vec{x}) = \frac{2.1952}{t^3 b}.$$

There are four constraints. The first constraint makes sure that the shear stress developed at the support location of the beam is smaller than the allowable shear strength of the material (13,600 psi). The second constraint makes sure that normal stress developed at the support location of the beam is smaller than the allowable yield strength of the material (30,000 psi). The third constraint makes sure that thickness of the beam is not smaller than the weld thickness from a practical standpoint. The fourth constraint makes sure that the allowable buckling load (along t direction) of the beam is more than the applied load F. A violation of any of the above four constraints will make the design unacceptable. The stress and buckling terms are given as follows [34]:

$$\tau(\vec{x}) \quad = \quad \sqrt{(\tau')^2 + (\tau'')^2 + (\ell\tau'\tau'')/\sqrt{0.25(\ell^2 + (h+t)^2)}},$$

$$\begin{aligned}
\tau' &= \frac{6,000}{\sqrt{2}h\ell}, \\
\tau'' &= \frac{6,000(14+0.5\ell)\sqrt{0.25(\ell^2+(h+t)^2)}}{2\left\{0.707h\ell(\ell^2/12+0.25(h+t)^2)\right\}}, \\
\sigma(\vec{x}) &= \frac{504,000}{t^2 b}, \\
P_c(\vec{x}) &= 64,746.022(1-0.0282346t)tb^3.
\end{aligned}$$

The variables are initialized in the following range: $0.125 \leq h, b \leq 5.0$ and $0.1 \leq \ell, t \leq 10.0$. Constraints are handled using the bracket-operator penalty function [9]. Penalty parameters of 100 and 0.1 are used for the first and second objective functions, respectively. We use real-parameter GAs with simulated binary crossover (SBX) operator [10] to solve this problem. Unlike in the binary-coded GAs, variables are used directly and a crossover operator that creates two real-valued children solutions from two parent solutions is used. For details of this crossover implementation, refer to original studies [10, 15, 13]. In order to investigate the search space, we plot about 230,000 random feasible solutions in f_1-f_2 space in Figure 1.16. The figure clearly shows the Pareto-optimal front near the origin. It can be seen that the Pareto-optimal solutions have a maximum end deflection value of around 0.01 inch, beyond which the an increase in end deflection also causes an increase in cost. It is also interesting to observe that a large density of solutions lie near the Pareto-optimal region, rather than away from it. An initial population of 100 solutions are shown in the figure. We use a σ_{share} of 0.281 (refer to equation 1.3 with $P = 4$ and $q = 10$). Figure 1.17 shows the populations at different generations. The figure clearly shows that NSGA progresses towards the Pareto-optimal front with generation. Finally, Figure 1.18 shows that the population after 500 generations has truly come near the Pareto-optimal front. Note here that the deflection axis is now plotted for a reduced range compared to that in Figure 1.16. These figures demonstrate the efficacy of NSGAs in converging close to the Pareto-optimal front with a wide variety of solutions.

Before we end the discussion on this problem, we would like to highlight an important by-product of multi-criterion optimization. If we investigate the two extreme solutions (the smallest cost solution and the smallest deflection solution), we observe an interesting property among the two solutions (Table 1.1). It is remarkable to note that both optimal solutions have identical values for three of four variables (h, ℓ, and t). The only way the solutions differ is in variable b. As long as the first three variables are set to the above values, any value of b (within lower and upper limits satisfying all constraints) would produce a Pareto-optimal or a near-Pareto-optimal solution. For a smaller value of b, the resulting design is a closer to a smaller cost design. This information is crisp and very useful for designers and is not apparent how else such an information could be found. It is not clear whether such a property exists

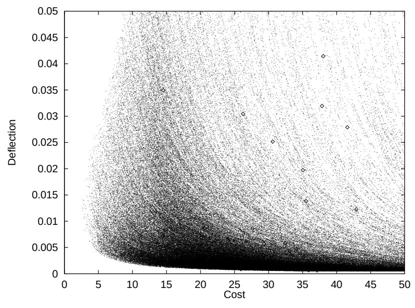

Fig. 1.16 About 230,000 random feasible solutions are shown in a function space plot.

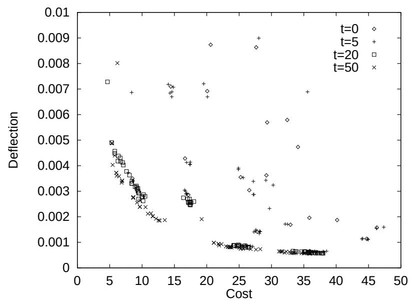

Fig. 1.17 Population history for a few generations are shown.

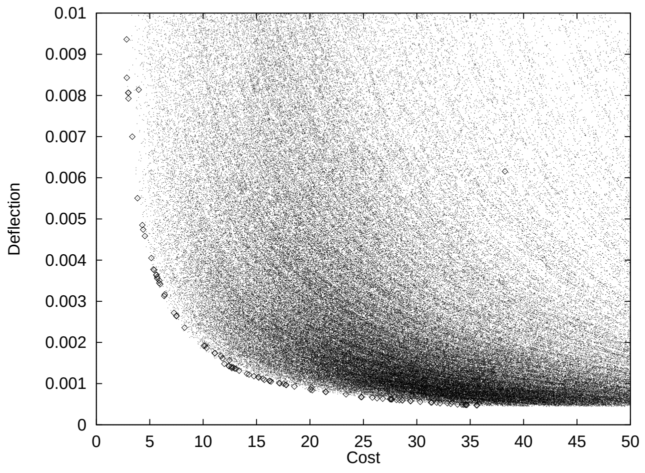

Fig. 1.18 Population at generation 500 shows that a wide range of Pareto-optimal solutions are found.

in Pareto-optimal solutions of other multi-criterion optimal design problems, but if there exists such a property, it is obvious that any designer would like to know it at any cost. Evolutionary approaches to multi-criterion optimization problems allow such a study to be carried out in other engineering design problems as well.

Table 1.1 Two extreme Pareto-optimal solutions are shown.

Cost	Defl.	h	ℓ	t	b
3.944	0.005	0.422	2.465	9.990	0.439
20.677	0.001	0.422	2.465	9.990	2.558

1.7 FUTURE DIRECTIONS FOR RESEARCH

As mentioned earlier, evolutionary multi-criterion optimization techniques are getting growing interests among researchers and practitioners. As the interests grow, there also exists a need of a directed research and application. In the following, we describe some of the immediate research directions that may help develop better algorithms for multi-criterion optimization.

1. Develop constrained test problems for multi-objective optimization
2. Comparison of existing multi-objective GA implementations
3. Understand dynamics of GA populations with generations
4. Scalability of multi-objective GAs with number of objectives
5. Convergence to Pareto-optimal front
6. Define appropriate multi-objective GA parameters (such as elitism)
7. Metrics for comparing two populations
8. Hybrid multi-objective GAs
9. Real-world applications
10. Multi-objective scheduling and other optimization problems

Because of page restrictions, we only discuss here the first issue by presenting a number of test problems. Interested readers may refer to [7] for details on other issues.

With the development of multi-objective optimization algorithms, there is an obvious need of a good set of test problems, which will test and compare different multi-criterion optimization algorithms with each other. Many test problems that are used in the literature are listed in [42]. However, it is not clear what aspect of an algorithm is tested by these test problems. Recently, a systematic procedure of constructing test problems have been suggested [7]. In that study, test problems are constructed from single-objective optimization problems to test two main features a multi-criterion optimization technique must have: (i) convergence ability to the true Pareto-optimal front and (ii) ability to find diverse Pareto-optimal solutions. To test these two features, three different functions (g, f_1, and h) are used to construct a two-objective optimization problem:

$$\begin{aligned}&\text{Minimize} \quad f_1(\vec{x}) = f_1(x_1, x_2, \ldots, x_m),\\ &\text{Minimize} \quad f_2(\vec{x}) = g(x_{m+1}, \ldots, x_N) h(f_1(x_1, \ldots, x_m), g(x_{m+1}, \ldots, x_N)).\end{aligned} \tag{1.8}$$

Since there are no variables which are common to both functions g and f_1, the Pareto-optimal solutions take those values of x_{m+1} to x_N variables which correspond to the global optimum of g. Since the function f_1 is also to be minimized, Pareto-optimal solutions take all possible values of x_1 to x_m variables. The above construction procedure allows the function g to control the search space lateral to the Pareto-optimal front. Thus, just by choosing a multi-modal or a *deceptive* g function [12], a multi-modal or deceptive two-objective optimization problem can be constructed. The function f_1 controls the search space along the Pareto-optimal front. Thus, a non-linear function of f_1 can construct a two-objective optimization problem having bias against some regions in the Pareto-optimal region. Finally, the function h controls the shape of the Pareto-optimal front. By choosing a convex, concave, or periodic function for h, a two-objective optimization problem having convex, concave, or discontinuous Pareto-optimal region can be constructed. In this study, we have already shown one such function (problem F3) which has a discontinuous Pareto-optimal region. This construction method allows each of two important features of a multi-criterion optimization algorithm to be tested in a controlled manner. In the following, we present a few test problems which can be used to test multi-criterion optimization algorithms.

Multi-modal multi-objective problem: Construct the problem in equation 1.8 using following functions:

$$\begin{aligned} f_1(x_1) &= x_1,\\ g(x_2, \ldots, x_N) &= 1 + 10(N-1) + \sum_{i=2}^{N} x_i^2 - 10\cos(2\pi x_i),\\ h(f_1, g) &= \begin{cases} 1 - \left(\frac{f_1}{g}\right)^{0.5}, & \text{if } f_1 \le g,\\ 0, & \text{otherwise.}\end{cases}\end{aligned}$$

The first variable lies in $x_1 \in [0, 1]$ and all others in $[-30, 30]$. Global Pareto-optimal solutions are as follows: $0 \le x_1 \le 1$ and $x_i = 0.0$ for $i = 2, \ldots, N$. A $N = 10$ or more is recommended. Multi-criterion optimization face difficulty with this problem because of the existence of $(61^{N-1} - 1)$ other local Pareto-optimal solutions each corresponding to an integer value of x_2 to x_N variables.

Deceptive multi-objective problem: The function is defined in *unitation*[3] u of N substrings of different lengths ℓ_i $(i = 1, \dots, N)$:

$$\begin{aligned} f_1 &= 1 + u(\ell_1), \\ f_2 &= \frac{\sum_{i=2}^{N} g(u(\ell_i))}{1 + u(\ell_1)}, \end{aligned}$$

where $g(u(\ell_i)) = \begin{cases} 2 + u(\ell_i), & \text{if } u(\ell_i) < \ell_i, \\ 1, & \text{if } u(\ell_i) = \ell_i. \end{cases}$

The global Pareto-optimal front has the following solutions: $0 \le u(\ell_1) \le \ell_1$ and $u(\ell_i) = \ell_i$ for all other $i = 2, \dots, N$. Note that this function is defined in gene the space. This problem is deceptive because the subfunction $g(u(\ell_i))$ shows that the function values of unitation $u = 0$ to $(\ell_i - 1)$ directs the search towards the deceptive attractor $u = 0$, whereas the solution $u = \ell_i$ is the true minimum of subfunction g.

Biased Pareto-optimal front: Construct the problem using equation 1.8 and following functions:

$$\begin{aligned} f_1(x_1) &= 1 - \exp(-4x_1)\sin^6(5\pi x_1), \\ g(x_2, \dots, x_N) &= g_{\min} + (g_{\max} - g_{\min}) \left(\frac{\sum_{i=2}^{N} x_i - \sum_{i=2}^{N} x_i^{\min}}{\sum_{i=2}^{N} x_i^{\max} - \sum_{i=2}^{N} x_i^{\min}} \right)^{\gamma}, \\ h(f_1, g) &= \begin{cases} 1 - \left(\frac{f_1}{g}\right)^2, & \text{if } f_1 \le g, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

Use $\gamma = 0.25$, $g_{\min} = 1$, and $g_{\max} = 10$. All variables vary in $[0, 1]$, thus $x_i^{\min} = 0$ and $x_i^{\max} = 1$. The true Pareto-optimal front has following values of variables: $0 \le x_1 \le 1$ and $x_i = x_i^{\min} = 0$ for $i = 2, \dots, N$. Multi-criterion optimization problems may face difficulty in this problem because of the non-linearity in f_1 function. There are two aspects of this problem: (i) There are more solutions having f_1 close to one than solutions having f_1 close to zero, and (ii) there will be a tendency for algorithms to find x_1 values within $[0, 0.2]$ than the other space ($x_1 \in [0.2, 1]$), although all solutions $x_1 \in [0, 1]$ are Pareto-optimal solutions. Thus, this problem tests an algorithm's ability to spread solutions over the entire Pareto-optimal region, although the problem introduces a bias against certain portion of the search space.

[3]Unitation is the number of 1s in the substring.

Concave Pareto-optimal front: Equation 1.8 with following functions can be used to construct this problem:

$$\begin{aligned} f_1(x_1) &= x_1, \\ g(x_2,\ldots,x_N) &= 1 + 10\frac{\sum_{i=2}^{N} x_i}{N-1}, \\ h(f_1,g) &= \begin{cases} 1 - \left(\frac{f_1}{g}\right)^{\alpha}, & \text{if } f_1 \leq g, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

All variables lie in $[0,1]$. The true Pareto-optimal front has following variable values: $0 \leq x_1 \leq 1$ and $x_i = 0$ for $i = 2,\ldots,N$. By setting $\alpha = 0.5$, we obtain a problem having convex Pareto-optimal front (like in Figure 1.2) and by setting $\alpha = 2$, we shall obtain a problem having non-convex Pareto-optimal front (line in Figure 1.3). Thus, multi-objective algorithms can be tested for the effect of non-convexity in the Pareto-optimal front just by changing the α parameter.

Discontinuous Pareto-optimal front: We construct the problem using equation 1.8 and the following functions:

$$\begin{aligned} f_1(x_1) &= x_1, \\ g(x_2,\ldots,x_N) &= 1 + 10\frac{\sum_{i=2}^{N} x_i}{N-1}, \\ h(f_1,g) &= 1 - \left(\frac{f_1}{g}\right)^{0.25} - \frac{f_1}{g}\sin(10\pi f_1). \end{aligned}$$

All variables vary in $[0,1]$ and true Pareto-optimal solutions are constituted with $x_i = 0$ for $i = 2,\ldots,N$ and discontinuous values of x_1 in the range $[0,1]$. This problem tests an algorithm's ability to find Pareto-optimal solutions which are discontinuously spread in the search space.

More complex test problems can be created by combining more than one aspects described above. Difficulties can also be increased by introducing parameter interactions by first rotating the variables by a transformation matrix and then using above equations. For more details, refer to the original study [7].

In Table 1.2, we outline some of the engineering design problems where an evolutionary multi-criterion optimization algorithm has been successfully applied.

Table 1.2 Applications of evolutionary multi-criterion optimization algorithms.

Researcher(s)	Year	Application area
P. Hajela and C.-Y. Lin	1992	Multi-criterion structure design [23]
J. W. Eheart, S. E. Cieniawski and S. Ranjithan	1993	Ground-water quality monitoring system [16]
C. M. Fonseca and P. J. Fleming	1993	Gas turbine engine design [17]
A. D. Belegundu et al.	1994	Laminated ceramic composites [1]
T. J. Stanley and T. Mudge	1995	Microprocessor chip design [39]
D. S. Todd and P. Sen	1997	Containership loading design [41]
D. S. Weile, E. Michielssen, and D. E. Goldberg	1996	Broad-band microwave absorber design [43]
A. G. Cunha, P. Oliviera, and J. A. Covas	1997	Extruder screw design [4, 6]
D. H. Loughlin and S. Ranjithan	1997	Air pollution management [27]
C. Poloni et al.	1997	Aerodynamic shape design [33, 32]
E. Zitzler and L. Thiele	1998	Synthesis of digital hardware-software multi-processor system [45]
G. T. Parks and I. Miller	1998	Pressurized water reactor reload design [31]
S. Obayashi, S. Takahashi, and Y. Takeguchi	1998	Aircraft wing planform shape design [30]
K. Mitra, K. Deb, and S. K. Gupta	1998	Dynamic optimization of an industrial nylon 6 semibatch reactor [29]
D. Cvetkovic and I. Parmee	1998	Airframe design [5]

1.8 SUMMARY

In this paper, we have discussed evolutionary algorithms for multi-criterion optimization. By reviewing a couple of popular classical algorithms, it has been argued that there is a need for more efficient search algorithms for multi-criterion optimization. A number of evolutionary algorithm implementations have been outlined and one particular implementation (called non-dominated sorting GA (NSGA)) has been discussed in details. The efficacy of NSGA has been demonstrated by showing simulation results on three test problems and on one engineering design problem. The results show that evolutionary algorithms are effective tools for doing multi-criterion optimization in that multiple Pareto-optimal solutions can be found in one simulation run. The study also suggests a number of immediate future studies including suggesting a few test problems for multi-criterion optimization, which should help develop better algorithms and maintain a focused research direction for the development of this fast growing field.

Acknowledgments

The author acknowledges the support provided by Alexander von Humboldt Foundation, Germany during the course of this study.

REFERENCES

1. Belegundu, A. D., Murthy, D. V., Salagame, R. R., and Constants, E. W. (1994). Multiobjective optimization of laminated ceramic composites using genetic algorithms. *Proceedings of the Fifth AIAA/USAF/NASA Symposium on Multidisciplinary Analysis and Optimization*, 1015–1022.

2. Chankong, V., and Haimes, Y. Y. (1983). *Multiobjective decision making theory and methodology*. New York:North-Holland.

3. Coello, C. A. C. (1998). A comprehensive survey of evolutionary based multiobjective optimization techniques. *Unpublished document.*

4. Cunha, A. G., Oliveira, P., and Covas, J. A. (1997). Use of genetic algorithms in multicriteria optimization to solve industrial problems. *Proceedings of the Seventh International Conference on Genetic Algorithms.* 682–688.

5. Cvetkovic, D. and Parmee, I. (1998). Evolutionary design and multi-objective optimisation. *Proceedings of the Sixth European Congress on Intelligent Techniques and Soft Computing (EUFIT)*, 397–401.

6. Covas, J. A., Cunha, A. G., and Oliveira, P. (in press). Optimization of single screw extrusion: Theoretical and experimental results. *International Journal of Forming Processes.*

7. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Technical Report No: CI-49/98. Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany.

8. Deb, K. (in press). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering.*

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

10. Deb, K. and Agrawal, R. B. (1995) Simulated binary crossover for continuous search space. *Complex Systems, 9* 115–148.

11. Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 42–50).

12. Deb, K. and Goldberg, D. E. (1994). Sufficient conditions for arbitrary binary functions. *Annals of Mathematics and Artificial Intelligence, 10,* 385–408.

13. Deb, K. and Goyal, M. (1998). A robust optimization procedure for mechanical component design based on genetic adaptive search. *Transactions of the ASME: Journal of Mechanical Design, 120*(2), 162–164.

14. Deb, K., Horn, J., and Goldberg, D. E. (1993). Multi-Modal deceptive functions. *Complex Systems, 7,* 131–153.

15. Deb, K. and Kumar, A. (1995). Real-coded genetic algorithms with simulated binary crossover: Studies on multi-modal and multi-objective problems. *Complex Systems, 9*(6), 431–454.

16. Eheart, J. W., Cieniawski, S. E., and Ranjithan, S. (1993). Genetic-algorithm-based design of groundwater quality monitoring system. *WRC Research Report No. 218.* Urbana: Department of Civil Engineering, The University of Illinois at Urbana-Champaign.

17. Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms–Part II: Application example. *IEEE Transactions on Systems, Man, and Cybernetics: Part A: Systems and Humans.* 38–47.

18. Fonseca, C. M. and Fleming, P. J. (1995). An overview of evolutionary algorithms in multi-objective optimization. *Evolutionary Computation, 3*(1). 1–16.

19. Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multi-objective optimization: Formulation, discussion, and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms.* 416–423.

20. Goldberg, D. E. (1989). *Genetic algorithms for search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

21. Goldberg, D. E. and Deb, K. (1991). A comparison of selection schemes used in genetic algorithms, *Foundations of Genetic Algorithms*, 69–93.

22. Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the First*

International Conference on Genetic Algorithms and Their Applications. 41–49.

23. Hajela, P. and Lin, C.-Y. (1992). Genetic search strategies in multicriterion optimal design. *Structural Optimization, 4.* 99–107.

24. Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, Michigan: MIT Press.

25. Horn, J. (1997). Multicriterion decision making. In Eds. (T. Bäck et al.) *Handbook of Evolutionary Computation.*

26. Horn, J. and Nafploitis, N., and Goldberg, D. E. (1994). A niched Pareto genetic algorithm for multi-objective optimization. *Proceedings of the First IEEE Conference on Evolutionary Computation.* 82–87.

27. Loughlin, D. H. and Ranjithan, S. (1997). The neighborhood constraint-method: A genetic algorithm-based multiobjective optimization technique. *Proceedings of the Seventh International Conference on Genetic algorithms*, 666–673.

28. Laumanns, M., Rudolph, G., and Schwefel, H.-P. (1998). A spatial predator-prey approach to multi-objective optimization: A preliminary study. *Proceedings of the Parallel Problem Solving from Nature, V.* 241–249.

29. Mitra, K., Deb, K., and Gupta, S. K. (1998). Multiobjective dynamic optimization of an industrial Nylon 6 semibatch reactor using genetic algorithms. *Journal of Applied Polymer Science, 69*(1), 69–87.

30. Obayashi, S., Takahashi, S., and Takeguchi, Y. (1998). Niching and elitist models for MOGAs. *Parallel Problem Solving from nature, V*, 260–269.

31. Parks, G. T. and Miller, I. (1998). Selective breeding in a multi-objective genetic algorithm. *Proceedings of the Parallel Problem Solving from Nature, V*, 250–259.

32. Poloni, C. (1997). Hybrid GA for multi-objective aerodynamic shape optimization. In G. Winter, J. Periaux, M. Galan, and P. Cuesta (Eds.), *Genetic algorithms in engineering and computer science.* Chichester: Wiley (Pages 397–414).

33. Poloni, C., Giurgevich, A., Onesti, L., and Pediroda, V. (in press). Hybridisation of multiobjective genetic algorithm, neural networks and classical optimiser for complex design problems in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering.*

34. Reklaitis, G. V., Ravindran, A. and Ragsdell, K. M. (1983). *Engineering optimization methods and applications.* New York: Wiley.

35. Rosenberg, R. S. (1967). *Simulation of genetic populations with biochemical properties.* PhD dissertation. University of Michigan.

36. Schaffer, J. D. (1984). Some experiments in machine learning using vector evaluated genetic algorithms. (Doctoral Dissertation). Nashville, TN: Vanderbilt University.

37. Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Proceedings of the First International Conference on Genetic Algorithms*, 93–100.

38. Srinivas, N. and Deb, K. (1994). Multi-Objective function optimization using non-dominated sorting genetic algorithms, *Evolutionary Computation, 2*(3), 221–248.

39. Stanley, T. J. and Mudge, T. (1995). A parallel genetic algorithm for multiobjective microprocessor design. *Proceedings of the Sixth International Conference on Genetic algorithms*, 597–604.

40. Steuer, R. E. (1986). *Multiple criteria optimization: Theory, computation, and application.* New York: Wiley.

41. Todd, D. S. and Sen, P. (1997). A multiple criteria genetic algorithm for containership loading. *Proceedings of the Seventh International Conference on Genetic algorithms*, 674–681.

42. van Veldhuizen, D. and Lamont, G. B. (1998). Multiobjective evolutionary algorithm research: A history and analysis. *Report Number TR-98-03.* Wright-Patterson AFB, Ohio: Department of Electrical and Computer Engineering, Air Force Institute of Technology.

43. Weile, D. S., Michielssen, E., and Goldberg, D. E. (1996). Genetic algorithm design of Pareto-optimal broad band microwave absorbers. *IEEE Transactions on Electromagnetic Compatibility, 38*(4).

44. Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—A comparative case study. *Parallel Problem Solving from Nature, V*, 292–301.

45. Zitzler, E. and Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. *Technical Report No. 43 (May 1998).* Zürich: Computer Engineering and Networks Laboratory, Switzerland.