Multi-Objective Evolutionary Algorithms: Introducing Bias Among Pareto-Optimal Solutions

Kalyanmoy Deb
Kanpur Genetic Algorithms Laboratory (KanGAL)
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
deb@iitk.ac.in

KanGAL Report No. 99002

Abstract

Since the beginning of Nineties, research and application of multi-objective evolutionary algorithms (MOEAs) have found increasing attention. This is mainly due to the ability of evolutionary algorithms to find multiple Pareto-optimal solutions in one single simulation run. In this paper, we present an overview of the multi-objective evolutionary algorithms and then discuss a particular algorithm in details. Although MOEAs can find multiple Pareto-optimal solutions, often, users need to impose a particular order of priority to objectives. In this paper, we present a few classical techniques to identify a preferred or a compromised solution, and finally suggest a biased sharing technique which can be used during the optimization phase to find a biased distribution of Pareto-optimal solutions in the region of interest. The results are encouraging and suggest further application of the proposed strategy to more complex multi-objective optimization problems.

1 Introduction

Many real-world optimization problems are naturally posed as multi-objective optimization problems. However, due to the lack of efficient multi-objective optimization algorithms, they have been suitably converted into single-objective optimization problems and solved. The basic difficulty arises due to the nature of the optimality conditions for multiple objectives. In the presence of multiple and conflicting objectives, the resulting optimization problem gives rise to a set of optimal solutions, instead of one optimal solution. Multiple optimal solutions exist because no one solution can be optimal for multiple conflicting objectives. Let us illustrate this concept through an example. If cost and reliability are two objectives in a design optimization, it is clear that a minimum cost solution is usually not maximally reliable and a maximally reliable solution is not often the cheapest. In such a scenario, none of these two extreme solutions (the cheapest and the most reliable solutions) can be declared as an absolute optimum corresponding to both objectives of design. In the parlance of multi-criterion decision-making, both these solutions are optimal in some sense or they are Pareto-optimal [33]. In fact, there exists many other solutions in the search space which are also Pareto-optimal. Since none of these solutions can be said to be an 'absolute optimum', the onus on the part of the user is then to first find as many such solutions as possible. Once multiple such solutions are found, usually, a higher-level decision-making strategies is adopted to choose a one solution from the set of obtained Pareto-optimal solutions.

In this paper, we describe the principle of multi-objective optimization and then discuss a number of evolutionary algorithms. Since evolutionary algorithms deal with a population of solutions [15], it is logical that they can be used to find multiple Pareto-optimal solutions in one single simulation run [3, 11, 18, 24,

26, 35]. We describe one such algorithm—Non-dominated sorting GA or NSGA [32]—in somewhat greater details. We present simulation results of NSGA on two problems.

Although most research on multi-objective evolutionary algorithms have concentrated their efforts in developing new and efficient search algorithms for finding solutions closer to the Pareto-optimal front and having a wider spread [14, 20, 32, 38], none of them concentrates on devising strategies to select one 'compromised' solution from the Pareto-optimal set. In this paper, we borrow a number of multi-objective optimization techniques from the classical field and suggest a number of post-optimization and optimization-level methods for choosing a Pareto-optimal solution or a biased Pareto-optimal region. Specifically, we suggest a biased sharing NSGA to find a biased distribution of solutions in the Pareto-optimal region. Instead of artificially forming a weighted sum of objective functions and finding the optimum of it, this method uses user's order of preference for objectives to find a biased distribution of Pareto-optimal solutions. On a couple of two-objective optimization problems, it is shown that the proposed biased sharing NSGA can find a biased distribution of solutions according to the weight vector setting. This procedure allows more solutions to be found in the region of user's preference. Although more studies are needed, the results of this paper show the efficacy of the proposed method and take the multi-objective evolutionary algorithms one step closer in making them usable in practice.

2 Multi-Objective 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. 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 [6]. The figure considers two objectives—cost and accident rate—both of which are to be minimized. The

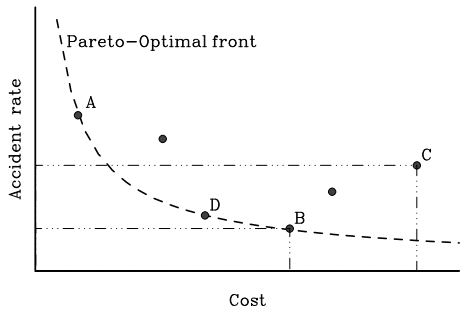

Figure 1: The concept of Pareto-optimal solutions is illustrated.

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 without any further information. 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 $x^{(2)}$ (having P decision variables each) 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 [33]:

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 solution 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 try to 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. Since no one solution in the Pareto-optimal set can be said to be better than the other, what an algorithm can do best to find as many different Pareto-optimal solutions as possible.

The classical ways of tackling multi-objective optimization problems is straightforward: Convert multiple objectives into one objective. There exists a number of conversion methods [1, 23, 31]—weighted sum approach, ϵ-perturbation method, Tchybeshev method, min-max method, goal programming method and others. Since multiple objectives are converted into one objective, the resulting solution to the single-objective optimization problem is usually subjective to the parameter settings chosen by the user. Moreover, since usually a classical optimization method is used, only one solution (hopefully a Pareto-optimal solution) can be found in one simulation run. Thus, in order to find multiple Pareto-optimal solutions, the chosen optimization algorithm must have to be used a number of times. Furthermore, the classical methods have been found to be sensitive to the convexity and continuity of the Pareto-optimal region.

3 Evolutionary Techniques

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 [28]. However, the first practical implementation was suggested by David Schaffer in the year 1984 [29]. 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 [15]. 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 [32, 14, 20] 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 [22], the concept of non-domination is simple to use and understand. However, a recent study [4] 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 [5, 13, 19, 34, 2]. These reviews list many different techniques of multi-criterion optimization that exist to date.

A web site maintained by Carlos A. Coello Coello (`http://www.lania.mx/~ccoello/EMOO/EMOObib.html`) shows that there are at least 300 research papers written till to date. A year-wise count of those papers is plotted in Figure 2, which shows an exponential growth in interest in the field over the past few years.

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

3.1 Schaffer's VEGA

Schaffer [29] 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.

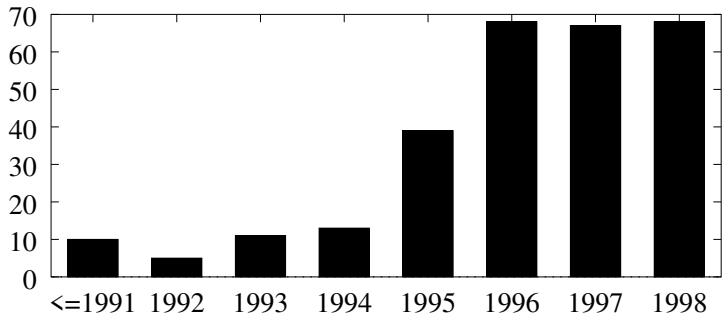

Figure 2: Growth in number of research papers in the field of multi-objective evolutionary algorithms.

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) [29, 30]. 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.

3.2 Fonseca amd Fleming's multi-objective GA

Fonesca and Fleming [14] 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 [16], 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 [4]. However, the ranking of individuals according to their non-dominance in the population is an important aspect of this work.

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

Horn, Nafploitis, and Goldberg [20] used *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 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 ($\sigma_{\rm share}$) 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.

3.4 Zitzler and Thiele's strength Pareto approach (SPEA)

Zitzler and Thiele [38] 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 [25].

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

Using the concept of sharing functions, Srinivas and Deb [32] 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 sharing function method is used to maintain diversity in the population. Since we proposed this method, 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, two procedures are performed serially. First, the population is ranked on the basis of an individual's non-domination level and

then sharing function method is used to assign fitness to each individual. We describe both these mechanisms in the following subsections.

3.5.1 Classifying a population according to non-domination

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 3) 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. In order to find the solutions belonging to the second level of non-domination, we temporarily disregard the solutions of the first level of non-domination and follow the above procedure. The resulting non-dominated solutions are the solutions of the second level of non-domination. These procedure is continued till all solutions are classified into a level of non-domination. It is important to realize that the number of different non-domination levels could vary between one to N. Figure 3 shows how the above procedure can be used to identify four different levels of non-domination with 5 solutions. In this example problem, the objective function f_1 is to be maximized and the objective function f_2 is to be minimized. Thus, the result

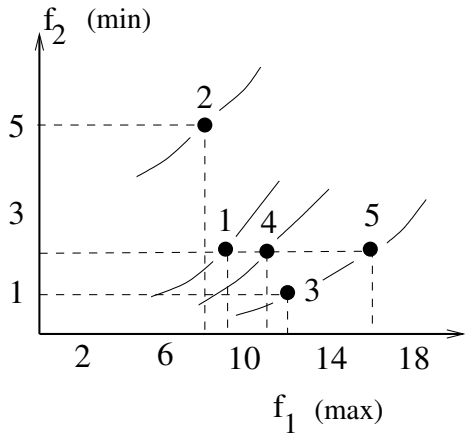

Figure 3: Classification of a population of solutions into different non-domination levels.

of classification is as follows: ((3,5), (4), (1), (2)), meaning that solutions 3 and 5 belong to the first level of non-domination, solution 4 belongs to the second level of non-domination, solution 1 belongs to the third level of non-domination, and solution 2 belongs to the fourth level of non-domination. It is important to note that this procedure has a computational complexity of $O(N^2)$.

3.5.2 Fitness assignment

In NSGA, a fitness is assigned to each individual according to its non-domination level. An individual in a higher level gets lower fitness. This is done in order to maintain a selection pressure for choosing solutions

from the lower levels of non-domination. Since solutions in lower levels of non-domination are better, a selection mechanism that selects individuals with higher fitness provides a search direction towards the Pareto-optimal region.

Setting a search direction towards the Pareto-optimal region allows one of the two tasks of multi-objective optimization. Providing diversity among current non-dominated solutions is also important in order to get a good distribution of solutions in the Pareto-optimal front.

Thus, the fitness assignment is performed in two stages. First, all solutions in a particular non-domination level (or front) are assigned an identical dummy fitness. Thereafter, based on the crowding of solutions in the front, lonely solutions are emphasized by using a sharing function strategy. We discuss both these strategies in the following.

First, all solutions in the first non-dominated front are assigned a fitness equal to the population size. This becomes the maximum fitness that any solution can have in any population. Based on the sharing strategy, if a solution has many neighboring solutions in the same front, its dummy fitness is reduced by a factor and a shared fitness is computed. The factor depends on the number and proximity of neighboring solutions. We shall describe the calculation procedure of this factor (usually known as the *niche count*) in a little while. Once all solutions in the first front are assigned their shared fitness values, the smallest shared fitness value is determined.

Thereafter, the individuals in the second non-domination level are all assigned a dummy fitness equal to a number smaller than the smallest shared fitness of the previous front. This makes sure that no solution in the second front has a shared fitness better than that of any solution in the first front. This maintains a pressure for the solutions to lead towards the Pareto-optimal region. The sharing method is again used among the individuals of second front and shared fitness of each individual is found. This procedure is continued till all individuals are assigned a shared fitness.

Thereafter, a stochastic remainder roulette-wheel selection [15] is used to select N individuals. Single-point crossover and bit-wise mutation are used as search operators.

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.

Sharing function method: 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 [8] 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 [17]:

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

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 [8]:

$$\sigma_{\text{share}} \approx \frac{0.5}{\sqrt[P]{q}}, \tag{1}$$

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 [4, 32] 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 $q \approx 10$. Although in all simulations here σ_{share} is kept constant, they can be varied with generation according to some pre-specified rules [12].

We illustrate the working of NSGA by hand-calculating one iteration of fitness assignment procedure on a simple test problem:

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

We randomly pick six solutions as shown in column 2 of Table 1. The first task is to classify the population

Table 1: A sample population and assignment of fitness.

Sl. No.	x	f_1	f_2	Front	Dummy fitness	Shared fitness
1	-1.5	2.25	12.25	2	3.00	3.00
2	0.7	0.49	1.69	1	6.00	6.00
3	4.2	17.64	4.84	2	3.00	3.00
4	2.0	4.00	0.00	1	6.00	3.43
5	1.75	3.06	0.062	1	6.00	3.43
6	-3.0	9.00	25.00	3	2.00	2.00

into different non-domination classes. By using the above-mentioned procedure, we identify the following classes in ascending order of non-domination level: ((2,4,5), (1,3), (6)). This means that solutions 2, 4 and 5 belong to the first class of non-domination and there are three non-domination classes. This fact is also evident from Figure 4. Since solutions 2, 4, and 5 fall on the Pareto-optimal region, they clearly belong to the first class of non-domination. Thereafter, solutions 1 and 3 belong to the second class of non-domination and finally the solution 6 belongs to the third class of non-domination. It is important to note that this figure is plotted with the objective function values.

Columns 3 and 4 show the two objective function value for each of six solutions. the fourth column shows the non-dominated front, each solution belongs to. The next task is to assign a dummy fitness to each solution. First, a fitness of 6 (equal to the population size) is assigned to solutions 2, 4 and 5. Thereafter, sharing function method is used to find niche count of each of these three solutions. Using the distance metric, we observe the following among these three solutions:

$$d(2,4) = 1.3, \quad d(2,5) = 1.05, \quad d(4,5) = 0.25.$$

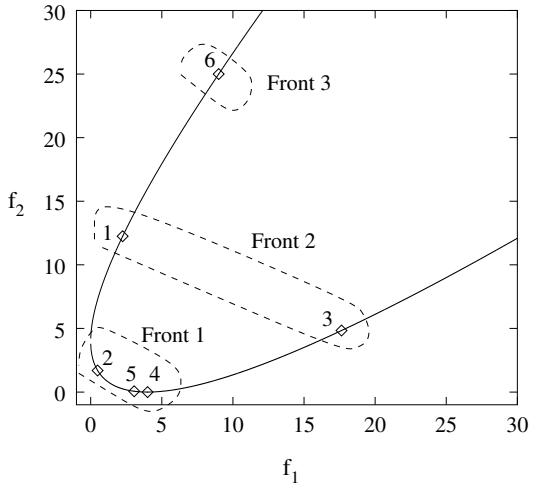

Figure 4: Six solutions are assigned a fitness in NSGA.

Using $\sigma_{\text{share}} = 0.5$, the sharing function values are calculated as follows:

$$Sh(2,4) = 0, \quad Sh(2,5) = 0, \quad Sh(4,5) = 0.75.$$

It is clear that $Sh(2,2) = Sh(4,4) = Sh(5,5) = 1$, since the distance between any solution and the solution itself is zero. Now the niche count of each solution is as follows:

$$m_2 = 1, \quad m_4 = 1.75, \quad m_5 = 1.75.$$

Thereafter, we divide the dummy fitness value of each solution by the corresponding niche count to compute the shared fitness value, as follows:

$$f_2' = 6/1 = 6, \quad f_4' = 6/1.75 = 3.43, \quad f_5' = 6/1.75 = 3.43.$$

It is important to note that since solutions 4 and 5 are closely spaced, their shared fitness is lower compared to the lone solution 2.

Once all solutions in the first front is evaluated for a shared fitness, the minimum shared fitness is computed as 3.43. Then, all solutions in the second front are assigned a dummy fitness equal to 3 (a number smaller than 3.43). Once again sharing procedure is adopted with $\sigma_{\text{share}} = 0.5$. Since solutions 1 and 3 are away from each other, their shared fitness is the same as the dummy fitness. Next, a dummy fitness equal to 2 (a number smaller than 3) is assigned to the only member (solution 6) in the third front. This completes the fitness assignment procedure. The stochastic remainder roulette-wheel selection is then performed with these shared fitness values (last column of Table 1).

4 Test Results

We present simulation results on two problems: (i) a single-variable two-objective problem and (ii) an engineering design problem.

4.1 Problem P1

We choose the above single-variable, two-objective problem given in equation 2. 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 of 100. The parameter σ_{share} is set to be

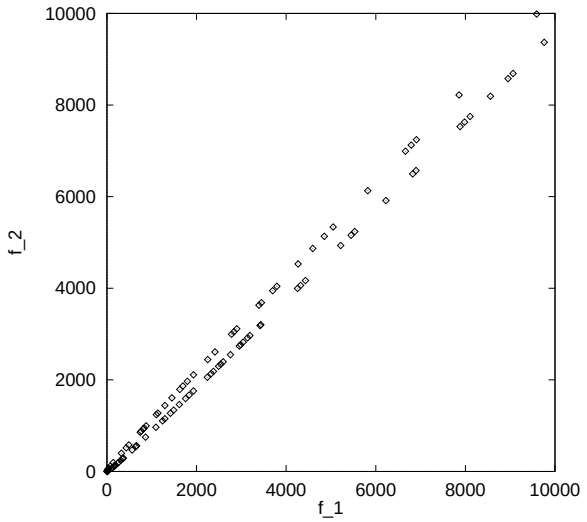

Figure 5: Initial population for function F1.

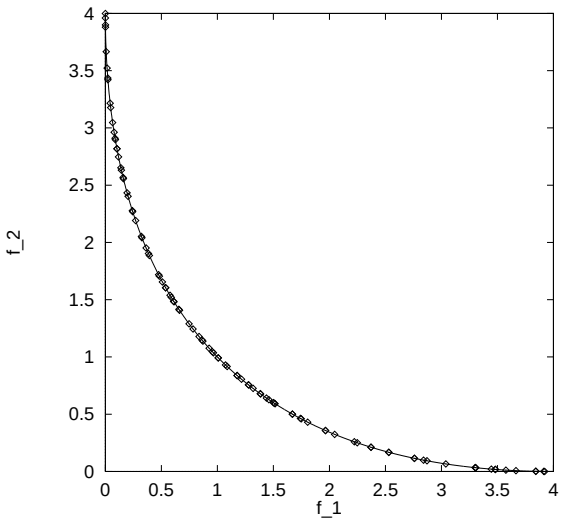

Figure 6: Population at generation 100 for function F1.

0.0002 here. This value induces about N or 100 niches in the Pareto-optimal region. Figure 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. Figure 6 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.

4.2 Problem P2: An engineering design problem

A beam needs to be welded on another beam and must carry a certain load F (Figure 7). It is desired to find

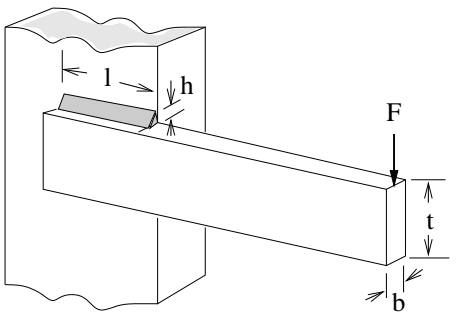

Figure 7: The welded beam design problem. Minimizations of cost and end deflection are two objectives.

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 and the deflection at the open end are 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. Again, a little thought will reveal that a design for minimum deflection at the end (or 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. 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 [10]:

$$\begin{array}{ll} \text{Minimize} & f_1(\vec{x}) = 1.10471h^2\ell + 0.04811tb(14.0+\ell), \\ \text{Minimize} & f_2(\vec{x}) = \delta(\vec{x}), \\ \text{Subject to} & g_1(\vec{x}) \equiv 13,600 - \tau(\vec{x}) \geq 0, \\ & g_2(\vec{x}) \equiv 30,000 - \sigma(\vec{x}) \geq 0, \\ & g_3(\vec{x}) \equiv b - h \geq 0, \\ & g_4(\vec{x}) \equiv P_c(\vec{x}) - 6,000 \geq 0. \end{array} \tag{3}$$

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

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

The expression for other terms can be found in the literature [27]. 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 [6]. 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 [7] 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 [7, 10, 9]. We use a $\sigma_{\rm share}$ of 0.281 (refer to equation 1 with $P = 4$ and $q = 10$). Figure 8 shows that the population after 500 generations (marked with diamonds) has truly come near the Pareto-optimal front. This plot demonstrates the efficacy of NSGAs in converging close to the Pareto-optimal front with a wide variety of solutions. The dots in the figure are randomly chosen feasible solutions.

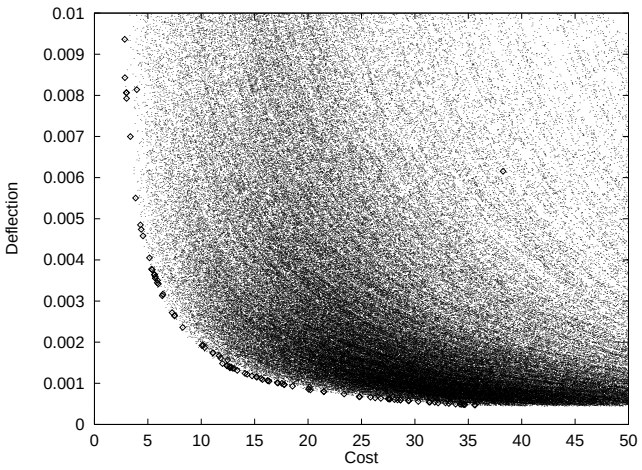

Figure 8: Population at generation 500 shows that a wide range of Pareto-optimal solutions are found for the welded beam design problem.

5 Introducing Bias

With last few years of research on multi-objective optimization using evolutionary algorithms, it is amply evident that evolutionary algorithms are capable of finding multiple Pareto-optimal solutions in a single simulation run. It is then natural to ask the question: 'How does one choose a particular solution from the obtained set of Pareto-optimal solutions?'. In the following, we first review a few techniques often followed

in the context of multi-criterion decision-making and then suggest a technique which can be used during the optimization phase.

5.1 Post-Optimal Techniques

Once the set of Pareto-optimal solutions is obtained, usually some higher-level decision making considerations (often societal or political) are used to pick a solution. Following methods are often used.

5.1.1 Compromise programming

The method of compromise programming or sometimes known as method of global criterion picks a solution from the Pareto-optimal set which is minimally located from a reference point [36, 37]. The user has to fix a distance metric $d()$ and a reference point $\vec{z}$ related to the problem. A couple of commonly-used metrics are presented below:

$$L_p\text{-metric:} \qquad d(\vec{f},\vec{z}) = \left(\sum_{k=1}^{M} |f_k(\vec{x}) - z_k|^p\right)^{1/p}, \tag{4}$$

$$\text{Tchebycheff metric:} \qquad d(\vec{f},\vec{z}) = \max_{k=1}^{M} \frac{|f_k(\vec{x}) - z_k|}{\max_{\vec{x}\in S} f_k(\vec{x}) - z_k}. \tag{5}$$

Here, S is the entire search space. The reference point is an *ideal* point, which is usually comprised of the individual best objective function values $\vec{z} = (f_1^*, f_2^*, \ldots, f_M^*)$. Since this solution is usually an infeasible solution, the user is interested in choosing a feasible solution (and Pareto-optimal) closest to this ideal solution. Figure 9 shows the reference point and the chosen solution in the Pareto-optimal set for a two-objective minimization problem. With two objectives, the L_2-metric becomes the Euclidean distance metric.

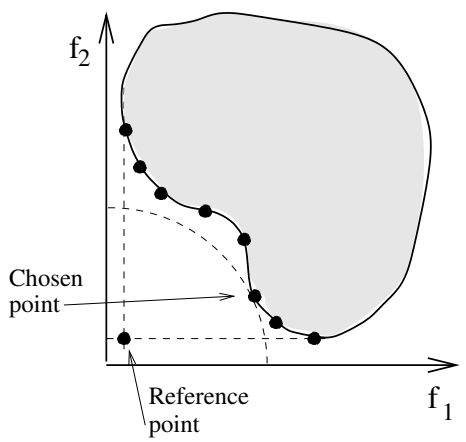

Figure 9: The reference point and the chosen optimal solution.

5.1.2 Marginal rate of return

It is the amount of improvement in one objective function which can be obtained by sacrificing an unit decrement in performance in any other objective function [23]. The solution having the maximum marginal rate of return is the one chosen by this method. Since pair-wise comparison have to made with all M objectives and for each Pareto-optimal solution, this method may be computationally expensive. Figure 10

shows the preferred 'knee' point, where the marginal rate of return is maximum among a set of obtained Pareto-optimal solutions.

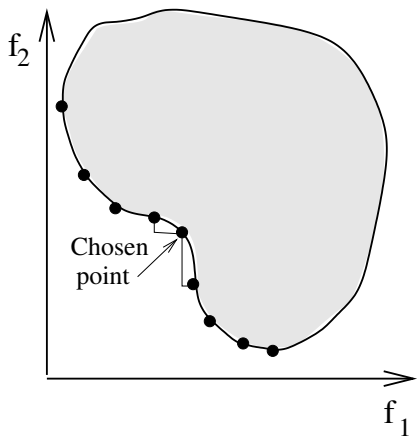

Figure 10: The chosen optimal solution having the maximum rate of return.

5.1.3 Weighted average

A simple strategy would be to choose a solution closer to the optimum solution corresponding to a particular user-specified weighted average of the objective functions. It is important to realize that this scheme is only applicable for identifying solutions in a convex Pareto-optimal region. The solutions inside the non-convex Pareto-optimal region cannot be found by this procedure. Figure 11 shows the chosen solution with a weighted scheme of $(w_1, w_2) = (0.85, 0.15)$. This procedure is different from the classical weighted

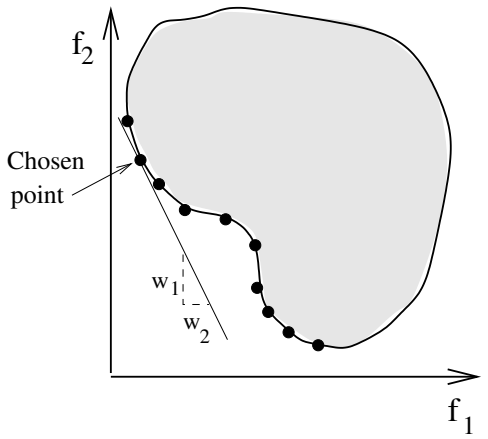

Figure 11: The chosen optimal solution with weighted average scheme.

average scheme in that here a solution is chosen after many Pareto-optimal solutions have been found. In the classical weighted average scheme, only one solution optimizing the weighted average of the objectives would be found. A different solution can be found with a different weight vector.

5.2 Optimization-level Techniques

In this section, we describe a couple of methods which can be used during the optimization phase to find biased solutions in the Pareto-optimal region.

5.2.1 Utility functions

Multiple objective functions can be used to form a utility function $U(\vec{f})$ [21]. The meaning of a utility function is that any two solutions having the same utility function value have the same user-preference to be chosen. This way multiple objectives are reduced to a single objective of maximizing the utility function. It is obvious that the construction of a utility function is problem-dependent and is highly subjective to the user. However, if an utility function can be constructed, a solution maximizing the utility function can be obtained.

5.2.2 Biased sharing approach

Here, we propose a sharing approach which uses a biased distance metric. In calculating the distance metric discussed in section 3.5.2 in the fitness-space sharing, the following normalized distance metric was suggested:

$$d(i,j) = \left(\sum_{k=1}^{M} \frac{(f_k^{(i)} - f_k^{(j)})^2}{(f_k^{\max} - f_k^{\min})^2} \right)^{\frac{1}{2}}. \tag{6}$$

This distance metric is nothing but the normalized Eucledian distance between two objective vectors. In the proposed biased sharing approach, an unequal weightage is given to each objective in computing the Eucledian distance. For example, if w_k ($\in (0,1)$) is the weight assigned to the k-th objective function, then normalized $\vec{w}'$ is calculated as follows:

$$w_k' = \frac{(1-w_k)}{\max_{k=1}^{M}(1-w_k)}, \tag{7}$$

and the modified distance metric is computed as follows:

$$d(i,j) = \left(\sum_{k=1}^{M} w_k' \frac{(f_k^{(i)} - f_k^{(j)})^2}{(f_k^{\max} - f_k^{\min})^2} \right)^{\frac{1}{2}}. \tag{8}$$

The fitness-based sharing can then be used with this distance metric. Although this new distance metric requires a new calculation procedure for σ_{share}, we have used the following formula:

$$\sigma_{\text{share}} = \frac{0.5}{10^{\frac{1}{M}}}. \tag{9}$$

This way the highest-priority objective function always gets a weightage (w) of one, whereas all others get a weightage between zero and one. It is interesting to note that if equal weights are assigned to each objective, equation 8 reduces to equation 6.

For convex Pareto-optimal regions, a higher weight (w) for an objective function will produce more dense solutions near the individual optimum. Figure 12 explains this fact. For a two-objective optimization problem, if an extreme case of $w_1 = 0$ and $w_2 = 1$ is used, the corresponding $\vec{w}'$ is as follows: $w_1' = 1$ and $w_2' = 0$. Since the effect of f_2 is absent in the calculating the distance metric, equal number of solutions are expected to be created in each equal partition of the f_1 search space. Thus, the density of Pareto-optimal solutions in the partition closer to the individual best f_1^* is) will be less. For non-convex Pareto-optimal region, w_k' can be calculated as $w_k' = w_k / \max_{i=1}^{M} w_k$.

In the following, we apply a modified NSGA with the above biased sharing approach to two multi-objective optimization problems (problems P1 and P2) having convex Pareto-optimal region. In both problems, we use a real-parameter GA [7] with simulated binary crossover and a polynomially distributed mutation operator. We use a population of size 100, crossover probability of 0.9, mutation probability of $1/P$, and two spread factors [7] for crossover and mutation $\eta_c = 30$ and $\eta_m = 500$, respectively. GAs are run till 500 generations are over.

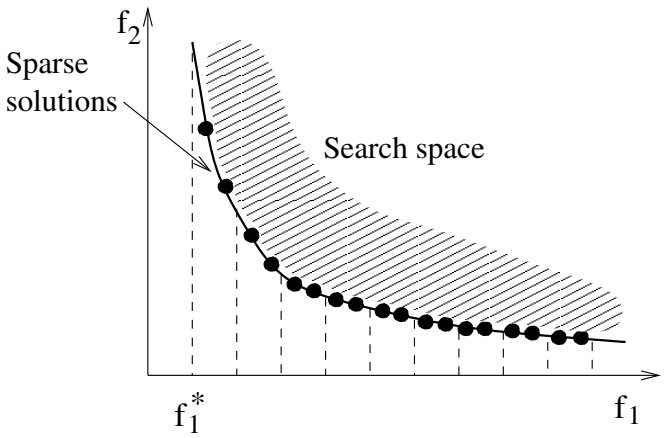

Figure 12: The density of Pareto-optimal solutions near the individual champion solution of f_1 is less with $w_1 = 0$ and $w_2 = 1$.

In Problem P1, the Pareto-optimal solutions correspond to $x \in (0, 2)$. We divide this region into 10 equal divisions and count the number of Pareto-optimal solutions found in each division under different weight vectors. Figure 13 shows the number of such individuals versus x (the number of solutions in (0,0.2) are shown at $x = 0.2$ in the figure). An average of 20 runs is plotted. The figure clearly shows that when weights $w_1 = w_2 = 0.5$ are used, almost a uniform distribution is observed. But when $w_1 = 0.1$ and $w_2 = 0.9$ are used, more solutions are found closer to $x = 2$ solution (the individual champion to the single-objective optimization problem: minimize $f_2 = (x - 2)^2$). But when $w_1 = 0.9$ and $w_2 = 0.1$ are used, an opposing trend emerged. This experiment shows how by changing the weight vector the density of solutions along the Pareto-optimal front can be changed. In order to investigate how the Pareto-optimal solutions are

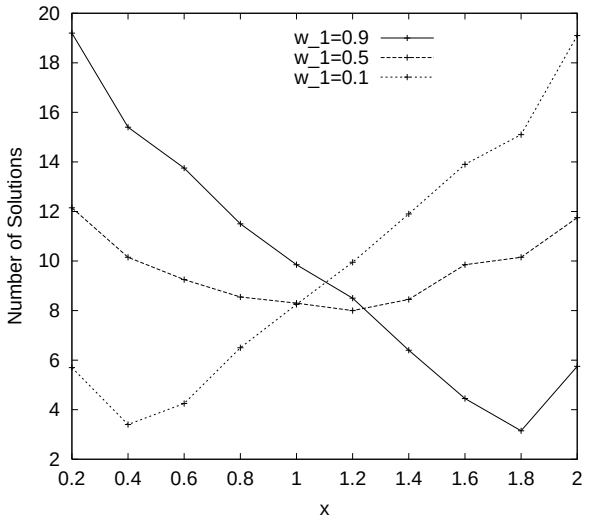

Figure 13: Number of solutions in each partition of the Pareto-optimal region under different weight vectors for problem P1.

distributed, we have also plotted the solutions of one run in Figures 14 and 15 for $w_1 = 0.9$ and $w_1 = 0.1$, respectively. The left figure with $w_1 = 0.9$ shows that more solutions are near f_1^* and the right figure with $w_1 = 0.9$ shows that more solutions are away from f_1^*.

Although the proposed approach does not provide a single compromised solution, instead finds a biased

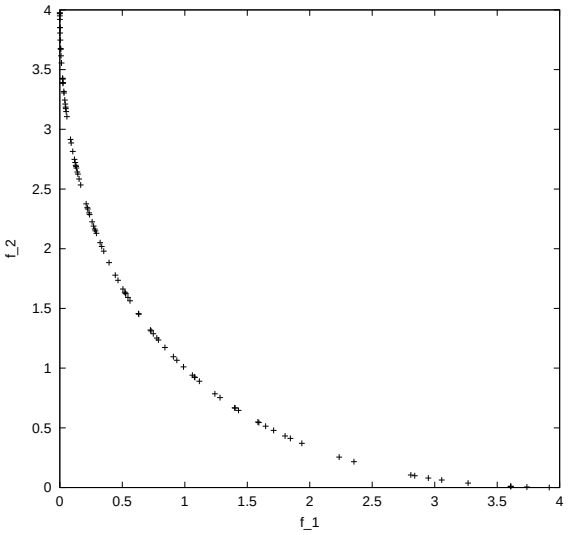

Figure 14: Biased Pareto-optimal solutions with $w_1 = 0.9$.

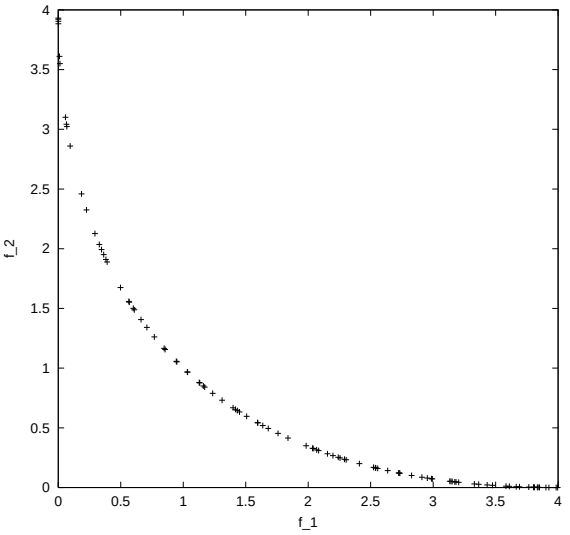

Figure 15: Biased Pareto-optimal solutions with $w_1 = 0.1$.

distribution of solutions, there is an advantage with the proposed method. If an user wants to have a bias towards a particular objective, this biased sharing approach produces more solutions towards the preferred region in the search space. Finding more dense solutions in the region of preference is a much better approach than predefining a weighted sum of objective and finding only one optimum solution. With a biased population, there exists many solutions in the desired region. One solution can then be chosen using previously described methods. In case preference of one objective over the other is not known, an equal weightage can be set for all objectives and a uniform distribution of solutions is expected to be found..

Next, we apply the above procedure for two different weight vectors to the welded beam design problem. Figures 16 and 17 show the spread of obtained Pareto-optimal solutions for $\vec{w} = (0.9, 0.1)$ and $\vec{w} = (0.1, 0.9)$, respectively. When the weightage for cost function is more (the left figure), more solutions near minimal cost solution are found. When a large weightage for the deflection function is introduced (the right figure), more solutions in the vicinity of the minimum deflection solution are obtained.

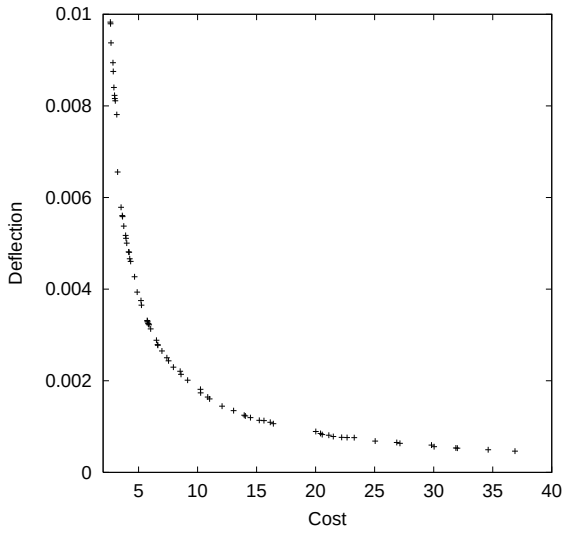

Figure 16: Biased Pareto-optimal solutions with $w_1 = 0.9$.

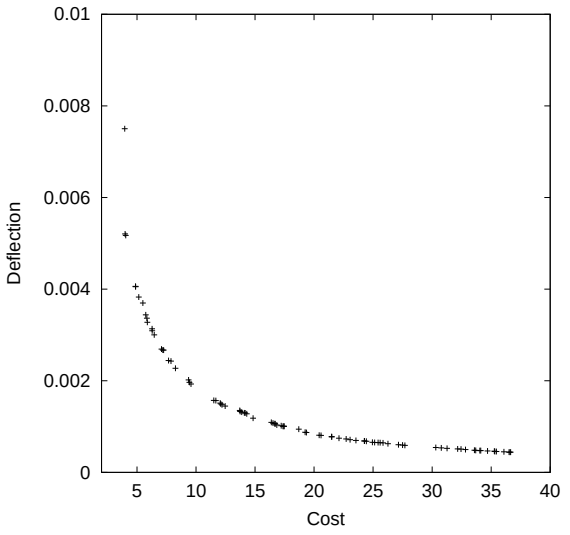

Figure 17: Biased Pareto-optimal solutions with $w_1 = 0.1$.

6 Conclusions

In this paper, we have discussed the principles of multi-objective optimization and have argued that the presence of multiple conflicting objective gives rise to multiple Pareto-optimal solutions, instead of one optimal solution. Therefore, there are two main goals in a multi-objective optimization problem: (i) convergence to the set of Pareto-optimal solutions, and (ii) maintenance of diversity among Pareto-optimal solutions. Since classical multi-objective optimization algorithms scalarize multiple objectives into a single objective, they are not efficient in satisfying the second goal described above. In the recent past, evolutionary algorithms have been efficiently used to satisfy both goals. This paper have reviewed a few salient algorithms, particularly the non-dominated sorting GA or NSGA.

A number of considerations have been presented for selecting one solution from the obtained Pareto-optimal set. A biased sharing NSGA has been proposed and its efficacy has been demonstrated by solving a couple of multi-objective optimization problems. With the popularity of multi-objective evolutionary algorithms, more studies are now needed to make them useful in practice. We believe that the biased sharing NSGA is one such algorithm which allows problem knowledge to be introduced into the search procedure. As a result, more solutions can be found in the desired Pareto-optimal region compared to other regions in the search space.

Acknowledgement

The author wishes to acknowledge the programming help of Shubhrajyoti Moitra and T. Meyarivan. The support from Department of Science and Technology (DST), Government of India during the course of this study is highly appreciated.

References

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

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

[3] 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.

[4] Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation Journal, 7*(3), 205–230.

[5] Deb, K. (1999). Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In K. Miettinen, M. Mäkelä, P. Neittaanmäki, and J. Périaux (Eds.) *Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN-99)*, (pp. 135–161).

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

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

[8] 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).

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

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

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

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

[17] 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.

[18] Hajela, P. and Lin, C.-Y. (1992). Genetic search strategies in multi-criterion optimal design. *Structural Optimization, 4*. 99–107.

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

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

[21] Keeney, R. L. and Raiffa, H. (1993). *Decisions with multiple objectives: Preferences and value tradeoffs*. Cambridge University Press.

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

[23] Miettinen, K. (1999). *Nonlinear multiobjective optimization*, Boston: Kluwer.

[24] 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.

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

[26] 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.

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

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

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

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

[31] Sen, P. and Yang, J.-B. (1998). *Multiple criteria decision support in engineering design*. London: Springer.

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

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

[34] 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.

[35] 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).

[36] Yu, P. L. (1973). A class of solutions for group decision problems. *Management Science, 19*(8), 936–946.

[37] Zeleny, M. (1973). Compromise programming. In J. L. Cochrane and M. Zeleny (eds.), *Multiple Criteria Decision Making*, Columbia: South Carolina: University of South Carolina Press, (pp. 262–301)

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