Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*

N. Srinivas and Kalyanmoy Deb
Department of Mechanical Engineering
Indian Institute of Technology
Kanpur, UP 208 016, INDIA
e-mail: deb@iitk.ernet.in

Abstract

In trying to solve multiobjective optimization problems, many traditional methods scalarize the objective vector into a single objective. In those cases, the obtained solution is highly sensitive to the weight vector used in the scalarization process and demands the user to have knowledge about the underlying problem. Moreover, in solving multiobjective problems, designers may be interested in a set of Pareto-optimal points, instead of a single point. Since genetic algorithms(GAs) work with a population of points, it seems natural to use GAs in multiobjective optimization problems to capture a number of solutions simultaneously. Although a vector evaluated GA (VEGA) has been implemented by Schaffer and has been tried to solve a number of multiobjective problems, the algorithm seems to have bias towards some regions. In this paper, we investigate Goldberg's notion of nondominated sorting in GAs along with a niche and speciation method to find multiple Pareto-optimal points simultaneously. The proof-of-principle results obtained on three problems used by Schaffer and others suggest that the proposed method can be extended to higher dimensional and more difficult multiobjective problems. A number of suggestions for extension and application of the algorithm is also discussed.

1 Introduction

Many real-world design or decision making problems involve simultaneous optimization of multiple objectives. In principle, multiobjective optimization is very different than the single-objective optimization. In single objective optimization, one attempts to obtain the *best* design or decision, which is usually the global minimum or the global maximum depending on the optimization problem is that of minimization or maximization. In the case of multiple objectives, there may not exist one solution which is best (global minimum or maximum) with respect to all objectives. In a typical multiobjective optimization problem, there exists a set of solutions which are superior to the rest of solutions in the search space when all objectives are considered but are inferior to other solutions in the space in one or more objectives. These solutions are known as *Pareto-optimal* solutions or *nondominated* solutions (Chankong and Haimes 1983; Hans 1988). The rest of the solutions are

*This paper has appeared in the Journal of *Evolutionary Computation*, Vol. 2, No. 3, pages 221–248.

known as dominated solutions. Since none of the solutions in the nondominated set is *absolutely* better than any other, any one of them is an acceptable solution. The choice of one solution over the other requires problem knowledge and a number of problem-related factors. Thus, one solution chosen by a designer may not be acceptable to another designer or in a changed environment. Therefore, in multiobjective optimization problems, it may be useful to have a knowledge about alternative Pareto-optimal solutions.

One way to solve multiobjective problems is to scalarize the vector of objectives into one objective by averaging the objectives with a weight vector. This process allows a simpler optimization algorithm to be used, but the obtained solution largely depends on the weight vector used in the scalarization process. Moreover, if available, a decision maker may be interested in knowing alternate solutions. Since genetic algorithms (GAs) work with a population of points, a number of Pareto-optimal solutions may be captured using GAs. An early GA application on multiobjective optimization by Schaffer (1984) opened a new avenue of research in this field. Though his algorithm, VEGA, gave encouraging results, it suffered from biasness towards some Pareto-optimal solutions. A new algorithm, Nondominated Sorting Genetic Algorithm (NSGA), is presented in this paper based on Goldberg's suggestion (Goldberg 1989). This algorithm eliminates the bias in VEGA and thereby distributes the population over the entire Pareto-optimal regions. Although there exist two other implementations (Fonesca and Fleming 1993; Horn, Nafpliotis, and Goldberg 1994) based on this idea, NSGA is different from their working principles, as explained below.

In the remainder of the paper, we briefly describe difficulties of using three common classical methods to solve multiobjective optimization problems. A brief introduction to Schaffer's VEGA and its problems are outlined. Thereafter, the nondominated sorting GA is described and applied to three two-objective test problems. Simulation results show that NSGA performs better than VEGA on these problems. A number of extensions to this work is also suggested.

2 Multiobjective Optimization Problem

A general multiobjective optimization problem consists of a number of objectives and is associated with a number of inequality and equality constraints. Mathematically, the problem can be written as follows (Rao 1991):

$$\begin{array}{rll} \text{Minimize/Maximize} & f_i(\mathbf{x}) & i = 1, 2, \ldots, N \\ \text{Subject to} & & \\ & g_j(\mathbf{x}) \leq 0 & j = 1, 2, \ldots, J \\ & h_k(\mathbf{x}) = 0 & k = 1, 2, \ldots, K \end{array} \tag{1}$$

The parameter $\mathbf{x}$ is a p dimensional vector having p design or decision variables. Solutions to a multiobjective optimization problem are mathematically expressed in terms of nondominated or superior points. In a minimization problem, a vector $\mathbf{x}^{(1)}$ is partially less than another vector $\mathbf{x}^{(2)}$, ($\mathbf{x}^{(1)} \prec \mathbf{x}^{(2)}$), when no value of $\mathbf{x}^{(2)}$ is less than $\mathbf{x}^{(1)}$ and at least one value of $\mathbf{x}^{(2)}$ is strictly greater than $\mathbf{x}^{(1)}$. If $\mathbf{x}^{(1)}$ is partially less than $\mathbf{x}^{(2)}$, we say that the solution $\mathbf{x}^{(1)}$ *dominates* $\mathbf{x}^{(2)}$ or the solution $\mathbf{x}^{(2)}$ is *inferior* to $\mathbf{x}^{(1)}$ (Tamura and Miura 1979). Any member of such vectors which is not dominated by any other member is said to be *nondominated* or *non-inferior*. Similarly if the objective is to maximize a function we define a dominated point if the corresponding component is not greater than that of a nondominated point. The optimal solutions to a multiobjective optimization problem are nondominated solutions. They are also known as *Pareto-optimal* solutions. The concept of Pareto-optimality is further illustrated in the next section by presenting an example problem. Mathematically, an optimization algorithm should be terminated if any one of the Pareto optimal solution is obtained. But in practice, since there could be a number of Pareto-optimal solutions and the suitability of one solution depends on a number of factors including designer's choice, problem environment, finding the entire set of Pareto-optimal solutions may be desired. In the following section, we describe a number of classical approaches to the solution of multiobjective optimization problems and discuss their difficulties by illustrating a simple two-objective optimization problem.

3 Classical Methods

A common difficulty with multiobjective optimization is the appearance of an *objective conflict* (Hans 1988)—none of the feasible solutions allow simultaneous optimal solutions for all objectives. In other words, individual optimal solutions of each objective are usually different. Thus a mathematically most favorable Pareto-optimum is that solution which offers least objective conflict. Such solutions can be viewed as points in the search space which are optimally placed from individual optimum of each objective. But such solutions may not satisfy a decision-maker because he/she may want a solution that satisfies some associated priorities of the objectives. To find such points all classical methods scalarize the objective vector into one objective. Many classical algorithms for non-linear vector optimization techniques define a substitute problem, reducing the vector optimization to a scalar optimization problem. Using such a substitute, a compromise solution is found subjected to specified constraints.

In the following subsections, three commonly-used methods—method of objective weighting, method of distance functions, and method of min-max formulation—are discussed.

3.1 Method of Objective Weighting

This is probably the simplest of all classical techniques. Multiple objective functions are combined into one overall objective function, Z, as follows:

$$\begin{array}{rcl} Z & = & \sum_{i=1}^{\mathbf{N}} w_i f_i(\mathbf{x}), \\ \text{where} \quad \mathbf{x} & \in & \mathbf{X}, \qquad \mathbf{X}\text{– the feasible region;} \end{array} \tag{2}$$

the weights w_i are fractional numbers $(0 \leq w_i \leq 1)$, and all weights are summed up to one, or $\sum_{i=1}^{\mathbf{N}} w_i = 1$. In this method, the optimal solution is controlled by the weight vector $\mathbf{w}$. It is clear from above equation that the preference of an objective can be changed by modifying the corresponding weight. Mathematically, a solution obtained with equal weights to all objectives may offer least objective conflict, but as a real-world situation demands a satisfying solution, priority must be induced in the formulation. In most cases, each objective is first optimized and all objective function values are computed at each individual optimum solution. Thereafter, depending on the importance of objectives a suitable weight vector is chosen and the single-objective problem given in equation 2 is used to find the desired solution. The only advantage of using this technique is that the emphasis of one objective over the other can controlled and the obtained solution is a usually a Pareto-optimum solution.

3.2 Method of Distance Functions

In this method, the scalarization is achieved by using a demand-level vector $\bar{\mathbf{y}}$ which has to be specified by the decision maker. The single objective function derived from multiple objectives is as follows:

$$Z = \left[\sum_{i=1}^{\mathbf{N}} |f_i(\mathbf{x}) - \bar{y}_i|^r \right]^{1/r}, \qquad 1 \leq r < \infty, \tag{3}$$

where $\mathbf{x} \in \mathbf{X}$ (the feasible region). Usually an Euclidean metric $r = 2$ is chosen, with $\bar{\mathbf{y}}$ as individual optima of objectives (Hans 1988). It is important to note that the solution obtained by solving above equation depends on the chosen demand-level vector. Arbitrary selection of a demand level may be highly undesirable. This is because a wrong demand level will lead to a nonPareto-optimal solution. As the solution is not guaranteed, the decision maker must have a thorough knowledge of individual optima of each objective prior to the selection of demand level. In a way this method works as a goal programming technique imposing a goal vector, $\bar{\mathbf{y}}$ (demand level), on the given objectives. This method is similar to the method of objective weighting. The only difference is that in this method the goal for each objective function is required to be known whereas in the previous method the relative importance of each objective is required.

3.3 Min-Max Formulation

This method is different in principle than the above two methods. This method attempts to minimize the relative deviations of the single objective functions from individual optimum. That is, it tries to minimize the objective conflict. For a minimization problem, the corresponding min-max problem is formulated as follows:

$$\text{minimize } \mathcal{F}(\mathbf{x}) = \max\ [Z_j(\mathbf{x})], \qquad j = 1, 2, \ldots, \mathrm{N}, \tag{4}$$

where $\mathbf{x} \in \mathbf{X}$ (the feasible region) and $Z_j(\mathbf{x})$ is calculated for nonnegative target optimal value $\bar{f}_j > 0$ as follows:

$$Z_j(\mathbf{x}) = \frac{f_j - \bar{f}_j}{\bar{f}_j}, \quad j = 1, 2, \ldots, \mathrm{N}. \tag{5}$$

This method can yield best possible compromise solution when objectives with equal priority are required to be optimized. However, priority of each objective can be varied by introducing dimensionless weights in the formulation. This can also be modified as a goal programming technique by introducing a demand-level vector in the formulation.

3.4 Drawbacks of Classical Methods

In all above methods, multiple objectives are combined to form one objective by using some knowledge of the problem being solved. The optimization of the single objective may guarantee a Pareto-optimal solution but results in single point solution. In real world situations decision makers often need different alternatives in decision making. Moreover, if some of the objectives are noisy or have discontinuous variable space these methods may not work effectively. Some of these methods are also expensive as they require knowledge of individual optimum prior to vector optimization. The most profound drawback of these algorithms is their sensitivity towards weights or demand-levels. The decision maker must have a thorough knowledge of the priority of each objective before forming the single objective from a set of objectives. The solutions obtained largely depend on the underlying weight-vector or demand-level. Thus, for different situations, different weight-vectors need to be used and the same problem needs to be solved a number of times. We illustrate this aspect by considering a simple example.

A simple two-objective problem $F1$ of one variable is considered to illustrate the concept of multiple Pareto-optimality. This problem was used for the same purpose by Vincent and Grantham

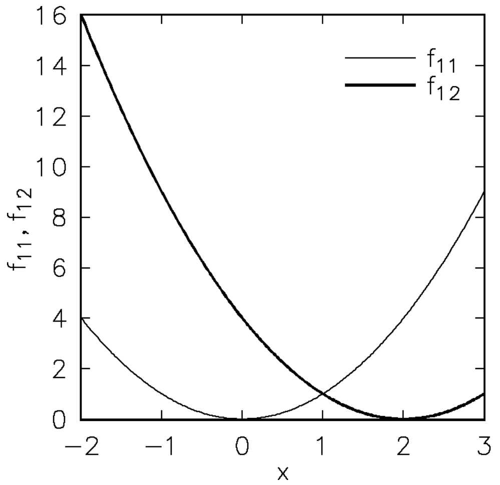


Figure 1: Functions f_{11} and f_{12} are plotted versus x.

Figure 2: The performance space of problem F1 is shown.

(1981) and subsequently by Schaffer (1984). The problem has two objectives and is shown in figure 1 and figure 2:

$$\begin{aligned} \text{Minimize} \quad f_{11} &= x^2 , \\ \text{Minimize} \quad f_{12} &= (x-2)^2 . \end{aligned} \tag{6}$$

From the plot showing the performance space, it is clear that the Pareto-optimal solutions constitute all x values varying from 0 to 2. The solution $x = 0$ is optimum with respect to f_{11} but not so good with respect to f_{12} and the solution $x = 2$ is optimum with respect to function f_{12} and not so good with respect to f_{11}. Any other point in between is a compromise or trade-off to the above two functions and is a Pareto-optimum point. But the solution $x = 3$, for example, is not a Pareto-optimum point since this point is not better than the solution $x = 2$ with respect to either objective.

Among the possible Pareto-optimal points, the decision maker may want to prefer one point over the other depending on the situation; but before taking any decision, he or she may want to know the other possible Pareto-optimal solutions. The traditional methods cannot find multiple Pareto-optimal solutions simultaneously. For example, with all the above methods and with equal priority to both functions having a weight vector (0.5,0.5), and demand-levels as individual optima, the obtained solution is $x^* = 1$. A weight vector $(1, 0)$ results in a scalarized objective as f_{11}. The solution obtained in this case is $x^* = 0$, which is optimum in f_{11} but not so good in f_{12}. Similarly the weight vector $(0, 1)$ produces the solution $x^* = 2$, which is the minimum point of f_{12}. Any point in the range $0 \leq x \leq 2$ may be a valid compromise and can be obtained with a particular choice of a weight vector. Thus, in order to obtain a particular solution, the decision maker has to know the corresponding weight vector, which is a difficult problem by itself. Another problem of using classical methods is that oftentimes some objectives may involve uncertainities. If the objective functions are not deterministic, the fixation of a weight vector or a demand-level may become even more difficult. This discussion suggests that the classical methods to handle multiobjective optimization problems are inadequate and inconvenient to use. A more realistic method would be one that can find multiple Pareto-optimal solutions simultaneously so that decision makers may be able to choose the most appropriate solution for the current situation. The knowledge of many Pareto-optimal solutions is also useful for later use, particularly when the current situation has changed and a new solution is required to be implemented. Since genetic algorithms deal with a population of points instead of one point, multiple Pareto-optimal solutions can be captured in the population in a single run. In the following section, we describe previous studies and current implementation of GAs to solve multiobjective optimization problems.

4 GA Implementation

As early as in 1967, Rosenberg suggested, but did not simulate, a genetic search to the simulation of the genetics and the chemistry of a population of single-celled organisms with multiple properties or objectives (Rosenberg 1967). The first practical algorithm, called Vector Evaluated Genetic Algorithm (VEGA), was developed by Schaffer in 1984 (Schaffer 1984). One of the problems with VEGA, as realized by Schaffer himself, is its bias towards some Pareto-optimal solutions. Later, Goldberg suggested a nondominated sorting procedure to overcome this weakness of VEGA (Goldberg 1989). Our algorithm, Nondominated Sorting Genetic Algorithm (NSGA), is developed based on this idea. There exists at least two other studies, different from our algorithm, based on Goldberg's idea. In the rest of this section, we discuss the merits and demerits of VEGA and NSGA, and the differences between NSGA and the two other recent implementations.

4.1 Schaffer's VEGA

Schaffer modified the simple tripartite genetic algorithm by performing independent selection cycles according to each objective. He modified Grefenstette's GENESIS program (Schaffer 1984) 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. Being a decision maker, we may not like to have any bias towards such middling individuals, rather we may want to find as many nondominated points as possible. Schaffer tried to minimize this speciation by developing two heuristics — the nondominated selection heuristic (a wealth redistribution scheme), and the mate selection heuristic (a cross breeding scheme) (Schaffer 1984). In the nondominated 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 nondominated individuals and was added to their expected number of copies during selection. But this algorithm failed when the population has very few nondominated individuals, resulting in a large fitness value for those few nondominated 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.

One method to minimize speciation is through a nondominated sorting procedure in conjunction with a sharing technique, as suggested by Goldberg (1989). Recently Fonesca and Fleming (1993) and Horn, Nafpliotis, and Goldberg (1994) implemented that suggestion, and successfully applied to some problems. These methods are briefly discussed later. But before that, we discuss our algorithm NSGA which is also developed based on Goldberg's suggestions.

4.2 Nondominated Sorting

The idea behind the nondominated sorting procedure is that a ranking selection method is used to emphasize good points and a niche method is used to maintain stable subpopulations of good points. Our algorithm is developed based on this concept. Since the algorithm is based on nondominated sorting procedure, we call this algorithm as the Nondominated Sorting Genetic Algorithm, NSGA.

4.2.1 Nondominated Sorting Genetic Algorithm (NSGA)

NSGA varies from simple genetic algorithm only in the way the selection operator works. The crossover and mutation operators remain as usual. Before the selection is performed, the population is ranked on the basis of an individual's nondomination described in section 2. The nondominated individuals present in the population are first identified from the current population. Then, all these individuals are assumed to constitute the first nondominated front in the population and assigned a large dummy fitness value. The same fitness value is assigned to give an equal reproductive potential to all these nondominated individuals. In order to maintain diversity in the population, these classified individuals are then *shared* with their dummy fitness values. Sharing methods are discussed elsewhere (Goldberg and Richardson 1987; Deb 1989). Sharing is achieved by performing selection operation using degraded fitness values which are obtained by dividing the original fitness value of an individual by a quantity proportional to the number of individuals around it. This causes multiple optimal points to co-exist in the population. After sharing, these nondominated individuals are ignored temporarily to process the rest of population in the same way to identify individuals for the second nondominated front. These new set of points are then assigned a new dummy fitness value which is kept smaller than the minimum shared dummy fitness of the previous front. This process is continued until the entire population is classified into several fronts.

The population is then reproduced according to the dummy fitness values. A stochastic remainder proportionate selection is used in this study. Since individuals in the first front have the maximum fitness value, they always get more copies than the rest of population. This was intended to search for nondominated regions or Pareto-optimal fronts. This results in quick convergence of the population towards nondominated regions and sharing helps to distribute it over this region. By emphasizing nondominated points, NSGA is actually processing the schemata representing Pareto-optimal regions. The efficiency of NSGA lies in the way multiple objectives are reduced to a dummy fitness function using nondominated sorting procedure. Another aspect of our method is that practically any number of objectives can be solved. Both minimization and maximization problems can also be handled by this algorithm. The only place a change is required for above two cases is the way the nondominated points are identified, as discussed in section 2.

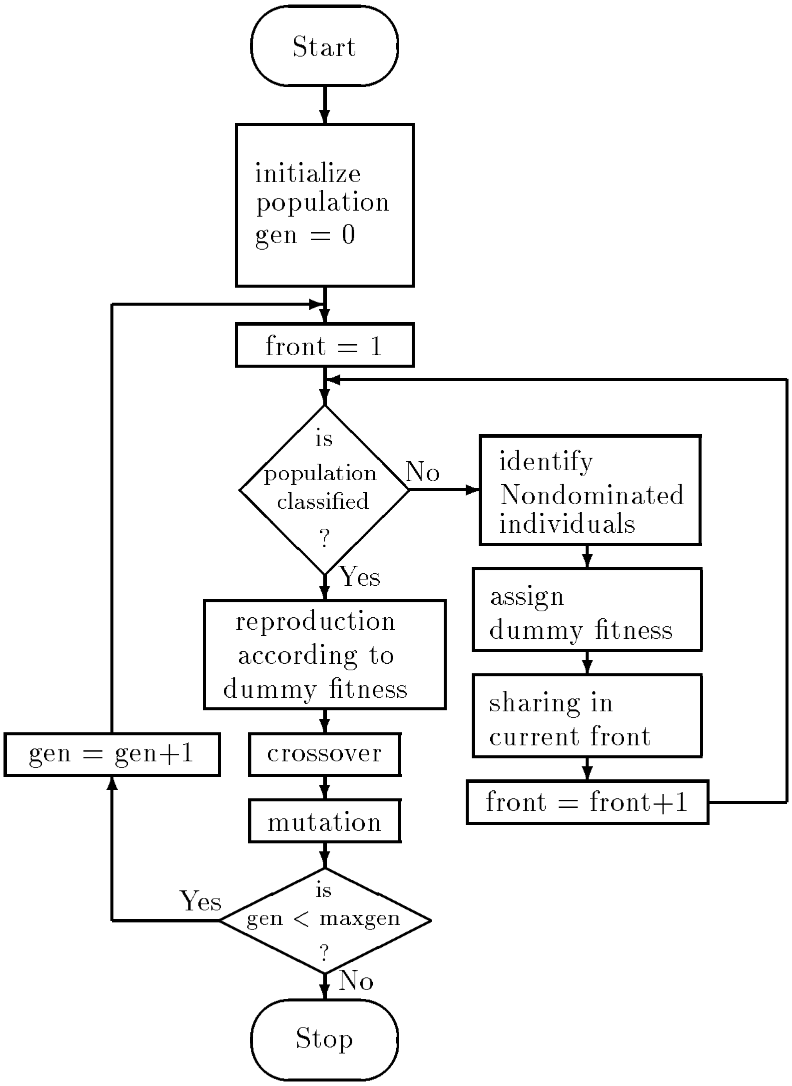
Figure 3 shows a flow chart of this algorithm. The algorithm is similar to a simple GA except the classification of nondominated fronts and the sharing operation. The sharing in each front is achieved by calculating a sharing function value between two individuals in the same front as follows:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{\text{share}}}\right)^2, & \text{if } d_{ij} < \sigma_{\text{share}}; \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

In the above equation, the parameter d_{ij} is the phenotypic distance between two individuals i and j in the current front and σ_{share} is the maximum phenotypic distance allowed between any two individuals to become members of a niche. Some guidelines to set these parameters appear elsewhere (Deb, 1989). A parameter *niche count* is calculated by adding the above sharing function values for all individuals in the current front. Finally, the shared fitness value of each individual is calculated by dividing its dummy fitness value with its niche count.

Fonesca and Fleming (1993) implemented Goldberg's suggestion in different way. In this study, the multiobjective optimization GA (MOGA) uses a similar sorting procedure presented in this paper. In MOGA, the whole population is checked and all nondominated individuals are assigned rank '1'. Other individuals are ranked by checking the nondominance of them with respect to the rest of the population in the following way. For an individual point, the number of points that strictly dominate the point 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 could be a number of points 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 (Goldberg and Deb, 1991), 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 maintains 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 (Srinivas, 1994). However, the ranking of the individuals according to their nondominance in the population is an important aspect of this work.

Horn, Nafpliotis, and Goldberg (1994) used a *Pareto domination tournaments* instead of non-dominated sorting and ranking selection method in solving multiobjective 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



æ

Figure 3: Flow chart of NSGA.

domination with respect to the objective functions. If one of them is nondominated and the other is dominated, then the nondominated point is selected. On the other hand, if both are either nondominated 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 nondominance calculation. Since this nondominance 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 nondominated (Pareto-optimal) points may not be found. If a small t_{dom} is chosen, this may result in a few nondominated 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 nondominated points is an important aspect of this work.

NSGA implements both aspects of Goldberg's suggestion in a better way. The ranking classification is performed according to the nondominance of the individuals in the population and a distribution of the nondominated points is maintained using a niche formation technique. Both these aspects cause the distinct nondominated points to be found in the population.

5 Simulation Results

In this section, we apply NSGA and VEGA on three test problems of which two were used by Schaffer (1984) and other one was solved by Chankong and Haimes (1983). In all simulations, the GA parameters used in the experiments are as follows:

Maximum generation	:	500
Population size	:	100
String length (binary code)	:	32
Probability of crossover	:	1.0
Probability of mutation	:	0.0

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 randomly spreading over entire variable space in consideration. To make the comparison fair, exactly the same initial population has been used in VEGA and NSGA. To confirm and recheck the solutions, each experiment is repeated five times with different initial populations and the average performance is presented in each case.

5.1 Problem F1

This problem is defined in section 3. Initial range for the design variable used in simulations is $(-10, 10)$, but the nondominated region is only $(0, 2)$. A comparison of population drift with NSGA and VEGA is shown in figures 4 through 11. These figures are drawn in performance space and σ_{share} parameter used is 0.1. As mentioned earlier, the initial population (at generation 0) is exactly same for both VEGA and NSGA. At generation 10, both methods show the convergence of the population towards the nondominated region. At generation 100, the difference in distribution is clear, and at generation 500, VEGA's population converged to only three sub-regions which are close to individual optima. This figure also shows the ability of NSGA in distributing the population uniformly and maintaining it till generation 500.

In order to study the distribution pattern better, the nondominated search space $(0, 2)$ is divided into 10 equal sub-regions. Since the population size is 100, we can expect to have about 10 individuals in each sub-region. Figures 12 and 13 show plots drawn with number of individuals in each sub-region and generation number. In case of VEGA (figure 12), after some generations, some of the sub-regions do not have any representation at all. These sub-regions represent middling individuals. Observations based on a number of simulation results reveal that atmost three sub-regions are populated by VEGA. These are the points around individual optima. Whereas, in case of NSGA (figure 13), the number of individuals in each sub-region fluctuated around a value 10, which is exactly the expected number of points in each sub-region. It is important to note that none of the sub-regions have zero individuals. Experiments are also carried out by considering a large search space, $-1000 \leq x \leq 1000$, compared to the nondominated region, $0 \leq x \leq 2$, so that the initial random population of size 100 has no Pareto-optimal points. Figure 14 shows that at generation 20 or so, the population is almost filled with nondominated points. The figure also shows that NSGA maintains a large proportion of the population at the nondominated region. A similar trend is also observed till generation 500. The distribution of points in the nondomination region is similar to that in the earlier experiments.

To quantify this distribution capability of population over nondominated regions, we calculate a performance measure, which is discussed in the following subsection.

5.1.1 Performance measure

In order to investigate how well NSGA and VEGA have distributed individuals over the nondominated region, we use the chi-square-like deviation form distribution measure used elsewhere (Deb,

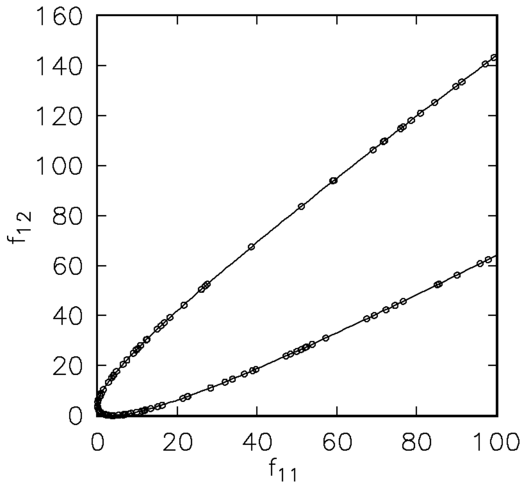


Figure 4: Population at generation 0 obtained using NSGA for problem F1 is shown.

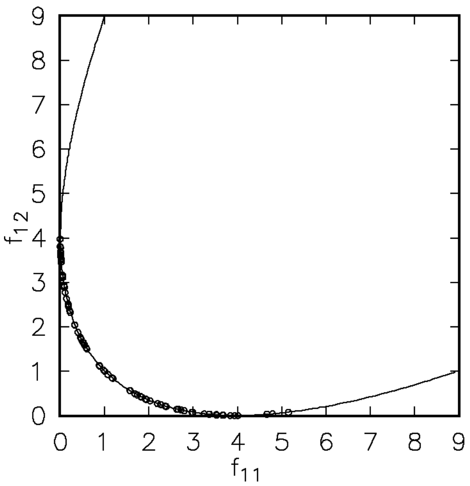


Figure 5: Population at generation 10 obtained using NSGA for problem F1 is shown.

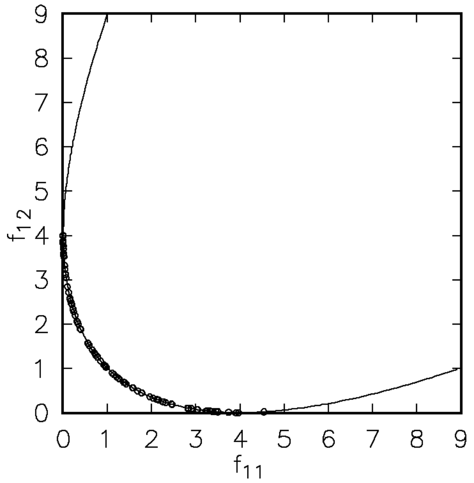


Figure 6: Population at generation 100 obtained using NSGA for problem F1 is shown.

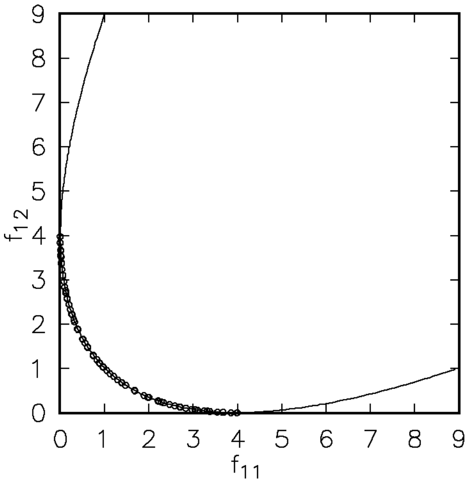


Figure 7: Population at generation 500 obtained using NSGA for problem F1 is shown.