Elitist Multi-Objective Genetic Algorithm for Optimization

A dissertation submitted in partial fulfillment
of the requirements of M.Tech.(Computer Science)
degree of Indian Statistical Institute, Kolkata
by

Anirban Roy

under the supervision of

Dr. C. A. Murthy and Dr. S. Bandyopadhyay
Machine Intelligence Unit

Indian Statistical Institute
203, Barrackpor Trunk Road
Kolkata-700 108.

26^{th} July 2002

Indian Statistical Institute

203, Barrackpore Trunk Road,

Kolkata-700 108.

Certificate of Approval

This is to certify that this thesis titled "**Elitist Multi-Objective Genetic Algorithm for Optimization**" submitted by **Anirban Roy** towards partial fulfillment of requirements for the degree of M. Tech in Computer Science at Indian Statistical Institute, Kolkata embodies the work done under our supervision.

CA Murthy 22/7/02

Dr. C. A Murthy,
Machine Intelligence Unit,
Indian Statistical Institute,
Kolkata-700 108.

Sanghamitra Bandyopadhyay 22/7/02

Dr. S. Bandyopadhyay,
Machine Intelligence Unit,
Indian Statistical Institute,
Kolkata-700 108.

Acknowledgements

I take pleasure in thanking Dr. C. A. Murthy and Dr. S. Bandyopadhyay for their friendly guidance throughout the dissertation period. Their pleasant and encouraging words have always kept my spirits up.

I also take the opportunity to thank my classmates, friends and my family for their assistance and encouragement to finish this work.

Anirban Roy

Contents

Chapter 1

Introduction and Literature Study

1.1 Introduction

In many real world situations there may be several objectives that must be optimized simultaneously in order to solve a certain problem. The main difficulty in considering multi-objective optimization is that there is no accepted definition of optimum in this case, and therefore it is difficult to compare one solution with another. For example, the objectives of designing an engineering device can be its efficiency and cost involved, which are contradictory in general.

One approach for solving multi-objective problems may be to optimize each criterion separately and combine the solutions thus obtained. However, this method is seldom likely to provide a solution where each criterion is optimally balanced. In fact, it may so happen that optimizing one objective may lead to unacceptably low performance of another objective. Thus for solving multi-objective problems all the objectives need to be treated together. In general, these problems admit multiple solutions, each of which is considered acceptable and equivalent when the relative importance of the objectives is unknown. The best solution is subjective and depends on the need of the designer or decision maker.

Genetic algorithm (GA)[6] is a stochastic process performing search over a complex and multidimensional space, in a randomized method in that it utilizes domain specific knowledge, in the form of objective function, to perform a directed random search. GAs involve a population based search in the solution space. Conventionally, GAs have been used for optimizing a single objective function. However, their popula-

tion based nature make them conducive for extension to the multi-objective case as well.

Traditional search and optimization methods such as gradient descent search, and other non-conventional ones such as *simulated annealing*[10] are difficult to extend to multi-objective case, since their basic design precludes such situations. For these methods, the multi-objective problems have to be reformulated as single objective ones using appropriate combination techniques like weighting etc. On the contrary population based methods like evolutionary algorithms are well suited for handling several criteria at the same time. In this work, we deal with the issues involved in multi-objective optimization using genetic algorithms, and proposed ways of enhancing their performance.

1.2 Previous Works

Several new algorithms on multi-objective GAs have been designed in recent years. With the assumption that the reader is familiar with the simple GA[6], we review some successful previous approaches; all of those evolutionary algorithms are based on *dominance relation* and *pareto optimality*. We shall describe the above terminologies in following section. Also we use the terms individual and solution interchangeably to denote a decision vector in genotypic space.

In multi-objective genetic algorithm (MOGA) by Fonseca and Fleming[5], each individual in the population is assigned a fitness, termed *assigned fitness*, by using a linear mapping function of its *pareto rank*, which is the number of individuals in the current population that dominate the individual. In order to maintain diversity among the *nondominated* solutions, Fonseca and Fleming have introduced niching among solutions of each rank. The normalized distance between any two solutions i and j in a rank calculated as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^{M} \left(\frac{f_k^i - f_k^j}{f_k^{max} - f_k^{min}} \right)^2}$$

where f_k^{max} and f_k^{min} are the maximum and minimum values of the k^{th} objective. For the solution i, d_{ij} is computed for each solution j(including i) having the same rank. Goldberg and Richardson[8] suggested a *sharing function* which gives an estimation of number of solutions belonging to each optimum in a multi-modal optimization problem. The *sharing function* is:

$$Sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{sh}})^{\alpha}, & \text{if } d \leq \sigma_{sh}; \\ 0, & \text{otherwise.} \end{cases}$$

The parameter d is the distance between two any two solutions in the population. The above function takes a value in $[0, 1]$, depending on the value of d and σ_{sh}. It is clear in a population, a solution does not get any effect from distant solutions, gets partial sharing effect from few solutions surrounding solutions and gets a full effect from itself. These *sharing function* values are calculated with respect to all population members (including itself) and are added to get a *niche count* nc_i for the i^{th} solution, as follows:

$$nc_i = \sum_{j=1}^{\mu(r_i)} Sh(d_{ij})$$

where $\mu(i)$ is the number of solutions in rank r_i. Now the *shared fitness* F_i' is calculated from assigned fitness F_i using $F_i' = F_i/nc_i$. This procedure is continued until all ranks are processed. Thereafter, the stochastic universal selection (SUS) with *shared fitness* value, the single point crossover, and the bit-wise mutation operators are applied to create a new population. Although the fitness assignment scheme is simple in MOGA, the *shared fitness* evaluation does not guarantee that a solution of poorer rank will have a worse fitness than every solution in a better rank.

Horn et al. have proposed niched pareto genetic algorithm (NPGA) [9]. In the literature of NPGA, binary tournament selection is used. During tournament selection, two solutions are picked at random from parent population. Of the two solutions, if one dominates all the individuals in a randomly selected *subpopulation* of size t_{dom}, and the other is dominated by at least one individual in the *subpopulation*, the former is chosen for subsequent operation. However, if both solutions are either dominated by at least one individual in the subpopulation or are not dominated by any individual in the *subpopulation*, *niche counts* for both individuals are evaluated to break the tie. The solution with smaller niche count is chosen to maintain diversity among solutions. Unlike MOGA, explicit fitness assignment is avoided in NPGA. The tournament selection prefers *nondominated* solutions in a stochastic manner and whenever a decision about *nondomination* cannot be established, the parents which reside in less crowded regions in the offspring population are chosen. The disadvantage of NPGA is that its performance heavily depends on the two external parameters, *niche radious* (σ_{sh}) and the size of the *subpopulation* (t_{dom}) used for tournament selection. These parameters should be tuned according to the nature of problem, which is always not possible.

Zitzler and Thiele proposed an elitist EA, which is named as strength pareto evolutionary algorithm (SPEA)[14]. This algorithm introduces elitism by explicitly maintaining an *external population*. The *external population* stores a fixed number of *nondominated* solutions found until that stage. The traditional population as well as the *external population* participates in genetic recombination. Each individual in the *external population* is assigned a stronger "strength" value if it dominates more individuals in the traditional population. On the other hand, each individual in the traditional population is assigned a greater fitness value if more individuals in the *external population* dominate it. Here, the greater fitness value an individual has, more worse it is. Unlike the previous algorithm, niching in SPEA is achieved implicitly by dominance comparison. Finally, *external population* size is restricted by clustering. Clustering ensures better spread among the obtained non-dominated solutions. SPEA introduces an extra parameter, the size of the external population $\overline{N}$. A balance between traditional population size N and *external population* size $\overline{N}$ is important in the successful working of SPEA.

Elitist non-dominated sorting GA has been suggested in 2000 by Deb et al.[3], often named as NSGA II. The algorithm employs elite and diversity preserving method. Parent and offspring population are combined together and *non-dominated sorting* is performed on the combined population. Once sorting is over, the new population is filled by solutions of different *nondominated* fronts, starting from the best one. The last allowed front which is not fully accommodated in the new population, is sorted based on the *crowding distance*. Individuals from the less crowded regions are chosen. The merit of this approach is no external parameter is required to be fixed in the niching strategy. However, when the crowded comparison is used to restrict the population size, it slows down the convergence property.

1.3 Shortcomings

All the algorithms discussed above follow the principle of *dominance relation* and *pareto optimality*, which will be discussed shortly. Furthermore, in order to maintain the diversity among the solutions in a population by keeping its size constant, different distance based niching strategies are applied either explicitly (in case of MOGA, NPGA and NSGA II) or implicitly (in case of SPEA). The *shared fitness* assignment in MOGA, *niche count* in NPGA and *crowding distance* assignment in NSGA II all serve the purpose of degrading fitness of a solution which is crowded by many other solutions. This helps emphasizing the solutions residing in less crowded region. In these diversity preserving mechanism, since the solutions in a crowded region degrade each other's fitness, it is very

likely that no solution from that region get carried to the next generation only because of they are surrounded by many solutions even though they are *nondominated* and can survive better than a solution from less crowded region. Consequently, searching takes place in a new direction sacrificing the current local optima. It is even possible that *pareto optimal* solutions may give their way to *nondominated* yet not *pareto optimal* solutions. The above uncertainty of choosing good solutions is inherent to the distance based niching strategy and hinders the algorithm from converging to the *pareto optimal* front quickly. Also, the external parameters t_{dom}(*subpopulation* size in NPGA) or $\overline{N}$(*external population* in SPEA) affect the performance of the algorithm extraneously; proper choice of these parameters is crucial for good performance of the algorithms. Another demerit is the prior knowledge of extreme values of the objective functions is essential for computing *niche count* or *crowded distance*. In many real world complex optimization problem, this information is difficult to achieve, if not impossible.

1.4 What needs to be done

As noted above, the existing methods of implementing MOGA tend to ignore all the solutions that reside in crowded regions, and also often require knowledge of the maximum and minimum values of the objective functions. In addition, a number of studies have established the fact that introduction of elitism in multi-objective evolutionary algorithm (MOEA) enhances its convergence characteristics. The performances of SPEA, NSGA II over the previous non-elitist approaches like MOGA, NPGA etc. were studied in detail by Deb et al.[2]. Motivated by these above, we concentrate on designing a new elitist-MOGA, which will overcome the aforementioned shortcomings. The algorithm should give more importance to the *nondominated* solutions from less crowded regions, without totally ignoring solutions from regions that are more crowded. In effect, the distance based niching strategy should be avoided to achieve fast convergence of the algorithm.Moreover, the final solutionshave to be well dispersed on the *pareto optimal* front. Lastly, no prior knowledge of extreme values of objective functions should be essential.

Chapter 2

Concepts and problems of MOGA

2.1 Formal Definition of MOOP

A Multi-Objective Optimization Problem (MOOP) has more than one objective function that are to be optimized simultaneously. Like single-objective optimization problem(SOOP), the problem has a number of constraints that constitute the feasible solution space. The formulation of MOOP is following :

Maximize/Minimize $f_m(\mathbf{x})$, $m = 1, 2, \ldots, M$
subject to $g_k(\mathbf{x}) \leq 0$, $k = 1, 2, \ldots, K$
where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a vector of n decision variables
and $L_i \leq x_i \leq U_i$, $i = 1, 2, \ldots, n$.

It is often referred to as vector optimization because a vector of objectives $\mathbf{f} = (f_1, f_2, \ldots, f_M)$, instead of single objective is being optimized. These multiple objectives often conflict with each other. The opposing objectives place a partial, rather than total, ordering of search space.

2.2 Dominance Relation and Pareto Optimality

Most multi-objective evolutionary algorithms use the concepts of *dominance relation* and *pareto optimality*. We provide the formal definitions of these concepts here for an MOOP where the M objectives are to be maximized. The definitions are analogous

for minimization problems.

Dominance Relation : A decision vector $\mathbf{x}^1 \in \mathbf{D}$ is said to dominate another decision vector $\mathbf{x}^2 \in \mathbf{D}$ if and only if $\mathbf{x}^2$ is partially less than $\mathbf{x}^1$ i.e.

$$\forall i \in \{1, 2, \ldots, M\}, f_i(\mathbf{x}^1) \geq f_i(\mathbf{x}^2)$$
$$\wedge \exists j \in \{1, 2, \ldots, M\} : f_j(\mathbf{x}^1) > f_j(\mathbf{x}^2)$$

Rank : The rank of an solution $\mathbf{x}^i$ in a population Q is said to be r_i if the solution is dominated by exactly r_i number of solutions in the population. The *nondominated* solutions are of rank zero, and there is at least one rank zero solution present in a population.

Pareto Optimality : A solution $\mathbf{x}^* \in \mathbf{D}$ is said to be *pareto optimal* if and only if there is no $\mathbf{x} \in \mathbf{D}$ for which $\mathbf{x}$ dominates $\mathbf{x}^*$. The set $\mathbf{P}^*$ of such *nondominated* solutions $\mathbf{x}^*$ over the entire feasible space $\mathbf{D}$ is the *pareto optimal* set.

The multi-objective evolutionary algorithms discussed here including the method suggested use the concept of *dominance relation* and *pareto optimality* proposed by Goldberg[6]. Now we are able to define the goals of MOEA in the light of above propositions. These are :

a. The resulting *nondominated* set of solutions should be close enough to the *pareto optimal* set, and
b. A good (possibly uniform) distribution of solutions along the *pareto optimal* front is desirable.

2.3 GA Implementation

In our implementation of genetic algorithm, the solution is represented by binary-encoded string of finite length, say L. The function h maps a binary string of length L to a solution vector of n variables i.e.

$$h : \{0, 1\}^L \longrightarrow \mathbf{D}^n$$

where $\mathbf{D}$ denotes the decision vector space.

The function h is one-one function such that $h(s) = \mathbf{x}$, $s \in \{0, 1\}^L$. Selection operation in simple GA chooses the strings from current population in proportionate to

their fitness values for reproduction. It is simple in SOOP as the fitness of a strings is its only objective value and total ordering of strings is possible based on the single objective. A few such selection techniques are tournament selection, roulette-wheel selection and ranking selection[7]. But implementation of selection method in MOGA is really a problem because *dominance relation* gives a partial ordering of the strings in terms of their fitness to reproduce. Earlier we discussed some algorithms like MOGA and NSGA II that impose total ordering of solutions based on *dominance relation* and distance based niching strategy. Avoiding imposition of such fitness ordering, we have proposed an elitist-selection technique that will be elaborated in following section. A population is a collection of binary strings s that constitutes the genotype space. The steps of EMOGA are following:

Step 1 : Initial population Q_0 of size N is produced randomly, followed by fitness evaluation and assignment of each individual in Q_0.

Step 2 : Crossover operation is performed on Q_0 to produce Q_1. Fitness evaluation and assignment of new individuals are performed.

Step 3 : Population Q_2 is generated from Q_1 by mutation operation. Fitness evaluation and assignment are performed on altered individuals.

Step 4 : Population Q_2 and Q_3 are merged together to have a population Q_m of size $2N$. Elitist selection is performed on Q_m to produce a new population Q_n of size N.

Step 5 : The termination condition is examined. The process terminates when the condition is satisfied. Assign the final population $Q_f \leftarrow Q_n$. Otherwise, assign $Q_0 \leftarrow Q_n$ and go to step 2.

In this implementation, we applied single-point crossover operation[8] and bit-wise mutation operation. Also the mutation probability is varying with generation.

2.4 Elitism

Elitism operation is often applied to preserve previously found best solutions in the subsequent generations. This operations ensures that once a good solution is found in an intermediate generation, it is never lost until a better solution replaces it. In 1996, Rudolph[11] and Bhandari et al.[1] have shown that a simple GA with elitism and

non-zero mutation probability always converges to the global optimum solution, though without elitism this is not always possible.

From the above discussion, it is clear that elitism plays an important role in evolutionary computing. In the context of MOOP, the meaning of elite solutions differs from that in SOOP. Here all the *nondominated* solutions in a population are elite solutions and are equally important. Preserving all elite solutions may block the avenues of new solutions in subsequent generations and result in premature convergence of the algorithm to a suboptimal point. So elitism must be applied in a controlled manner[4] such that search process does not stagnate.

Chapter 3

Elitist Multi-Objective GA

3.1 Elitist-Selection in EMOGA

The suggested multi-objective genetic algorithm, EMOGA, differs from other elitist MOEA in its elitist selection operation. In this section we shall highlight on the elitist-selection algorithm, its computational complexity and lastly the theoretical foundation.

In the proposed elitist-selection operation, the parent population Q_p and offspring population Q_c (produced by crossover and mutation operations) are merged together to form a new population Q_m of size $2N$. Rank assignment is performed for each individual in Q_m and individuals are classified according to their rank. An individual is said to be in *rank subpopulation* R_i, $0 \leq i \leq 2N-1$, if the rank of the individual is i. So the current set of *nondominated* individuals belong to R_0. If the number ofindividuals in R_0, say n_0, is less than or equal to the population size N, all the individuals from R_0 are taken to the new population Q_n. Remaining $(N - n_0)$ individuals are taken randomly from R_i, $1 \leq i \leq 2N-1$. When n_0 is grater than N, we do the following for each objective function f_k, $1 \leq k \leq M$:

The current maximum value f_k^{max} and current minimum value f_k^{min} of f_k are found out from *rank subpopulation* R_0. The span of f_k is divided into d number of regions. The individuals in R_0 are classified based on their f_k value. An individual is said to be in *region subpopulation* G_j, $1 \leq j \leq d$ if its f_k value lies in the j^{th} region. Then, each region is visited in round-robin manner and one individual is selected from each region. The individual which is best in other objectives, except f_k, is chosen. If there are more than one such individual, tie is broken by selecting one randomly. On the other hand, if the region is empty, it is skipped out. The process is continued until exactly c number

of individuals are chosen based on the objective function f_k.

After the above steps are followed for M objective functions, new population Q_n consists of cM number of individuals. Remaining $(N - cM)$ number of individuals are chosen randomly from the rest of the individuals in Q_m. This newly formed population Q_0 becomes the parent population for the next generation. The detailed algorithm of ELITE_SELECT routine is described below:

Procedure ELITE_SELECT

Input : Q_p(Parent Population), Q_c(Child Population), N(Population Size), d(Number of Regions), c(Number of individuals to be chosen for each objective component).

Output : Q_n(New Population)

Begin

1. Initialize $Q_n \leftarrow \emptyset$.
2. Merge Q_p and Q_c to form a population Q_m of size $2N$.
3. Using *dominance relation* defined earlier, rank each individual in Q_m. An individual $s_i \in Q_m$ is said to have rank r_i if it is dominated by exactly r_i number of individuals in Q_m.
4. Construct *rank subpopulations* $R_j = \{ s_i \in Q_m : r_i = j, 1 \leq i \leq 2N \}$, $j = 0, 1, \ldots, 2N-1$. Let $n_0 = |R_0|$, number of *nondominated* solutions in Q_m.
5. **if** $n_0 \leq N$ **then**
6. $\quad Q_n \leftarrow Q_n \cup R_0$.
7. $\quad rem \leftarrow (N - n_0)$; $selected \leftarrow 0$.
8. $\quad$ **while** $selected < rem$ **do**
9. $\quad\quad$ **for** $i \leftarrow 1$ to $2N-1$ **do**
10. $\quad\quad\quad$ **if** $R_i \neq \emptyset$ **then**
11. $\quad\quad\quad\quad$ Choose one individual s randomly from R_i.

12. $Q_n \leftarrow Q_n \cup \{s\}$, $R_i \leftarrow R_i - \{s\}$; selected $\leftarrow$ selected + 1.

13. **if** *selected* = *rem* **then**

14. Exit from inner loop
end if
end if
end for
end while

15. **else**

16. **for** $i \leftarrow 1$ to M **do**

17. Find maximum and minimum values of f_i in the set R_0. Let those be denoted by f_i^{max} and f_i^{min} respectively.

18. Divide the objective space into d number of regions along the i^{th} objective. Let the region boundaries be $f_{i_0}, f_{i_1}, \ldots, f_{i_d}$. Then we have $f_{i_0} = f_i^{min}$ *and* $f_{i_d} = f_i^{max}$. More generally,

$$f_{i_k} = f_i^{min} + k.\frac{f_i^{max} - f_i^{min}}{d},\ k = 1, 2, \ldots, d.$$

19. Construct the *region subpopulations* $G_j = \{s \in R_0 : f_{i_{j-1}} < f_i(s) \leq f_{i_j}(s)\}$, $j = 1, 2, \ldots, d$.

20. *selected* $\leftarrow$ 0.

21. **while** *selected* $< c$ **do**

22. **for** $j \leftarrow 1$ to d **do**

23. **if** $G_j \neq \emptyset$ **then**

24. Choose the individual $s \in G_j$ which is best in other objectives (select randomly if there is more than one).

25. $Q_n \leftarrow Q_n \cup \{s\}$; $G_j \leftarrow G_j - \{s\}$; $R_0 \leftarrow R_0 - \{s\}$; *selected* $\leftarrow$ *selected* + 1.

26. **if** *selected* = c **then**

```
27.                 Exit from inner loop
                  end if
                end if
              end for
            end while
          end for
```

28. $rem \leftarrow (N - cM)$; $selected \leftarrow 0$.

29. **while** $selected < rem$ **do**

30. Randomly select a rank $r' \in \{0, 1, \ldots, 2N - 1\}$.

31. if $R_{r'} \neq \emptyset$ **then**

32. Choose an individual s randomly from $R_{r'}$.

33. $Q_n \leftarrow Q_n \cup \{s\}$; $selected \leftarrow selected + 1$.
    end if
  end while
end if

34. **return** Q_n.

End Procedure

3.2 Computational Complexity

Step 1 needs constant time. Step 2 merges the populations and takes $O(N)$ computation. Determining rank of each individual in a population of size N requires checking dominance for each pair of individuals; the number of such pairs are ${}^{N}C_2$ i.e. $O(N^2)$. Since comparison for dominance needs comparison of M function values, in total step 2 takes $O(MN^2)$ comparisons. Step 4 and step 6 takes $O(N)$ time. Considering the worst case where remaining solutions (after storing all rank 1 solutions) may belong to same rank, step 8 will be executed at most $O(rem)$ times. Since $rem < 2N$, step 8 takes $O(N)$ computation at most. Now step 9 is executed $2N - 1$ times for each execution of step 8. In total steps 8-14 take $O(N)$ computations. Step 16 is executed M times. Step 17-19 takes $O(N)$ times for each execution of step 16. From the previous logic, step 21 can be executed c times in worst case for each objective. Since step 22 is always executed d times for each execution of step 21, step 23 is executed $O(N)$ times for each objective.

But steps 24-27 can not be executed more than $O(c)$ times for each objective. Step 24 takes $O(MN)$ amount of computation to find the best solution in other objectives. So total computation of choosing c individuals from M objectives is $O(cM^2N)$. Remaining solutions are chosen in $O(N)$ time. Assuming that $c = O(N)$, overall worst-case computational complexity of the ELITE_SELECT algorithm is $O(M^2N^2)$.

3.3 Discussion on Theory

The basic philosophy of EMOGA is to divide the objective space into a number of regions along each of the M objectives. The current *nondominated* solutions are categorized region-wise and optimization is done in each region along the objective independently by picking best solutions in other objectives. For the sake of simplicity, let us consider a bi-objective optimization problem. Also we will assume that the region length is fixed.

In their work on convergence of elitist GA (EGA) in 1996[1], Bhandari et al. established that with a positive mutation probability, elitist GA always converges to the global optimal solution after infinite number of iterations. In the algorithm of EMOGA, elitism is maintained by copying the current *nondominated* solutions from the combined parent-child population to the parent population of next generation. If the number of *nondominated* solutions never exceeds population size N, with the similar argument we say that the algorithm always converges to the *pareto optimal* front with probability 1 as the number of iterations goes to infinity if the mutation probability lies between zero and one. The difficulty arises when the number of *nondominated* solutions in the combined population exceeds population size N. In that case we apply the technique explained in the preceding paragraph. In an intermediate generation, we consider a region along first objective. The *nondominated* solutions in that region are optimized on the second objective. If the region location and size is fixed throughout the process, it follows from the convergence of EGA that the algorithm always finds global optimal value of second objective in that particular region. Since the proposition is true for each region and each objective, the algorithm ensures optimization along each objective with the restriction of number of regions d. The *nondominated* solutions are optimized on each of the objectives, which leads the current *nondominated* front to the *pareto optimal* front. As with simple EGA, once a *pareto optimal* solution is found, it survives throughout the process.

Chapter 4

Experimental Results and Comparisons

In this section, we discuss the implementation results obtained using EMOGA, along with its comparison with two other well known multi-objective GAs, NSGA II and NPGA, for a suite of six test problems. The first two problems are string search problems and defined on binary domain. The following four problems are mathematical functions which are already used to test optimization algorithms. In order to compare the performance of proposed method, two other existing MOGAs, NPGA (non-elitist) and NSGA II (elitist), were also implemented for the same set of six problems. In our experiment, string length $L = 50$ and crossover probability $p_c = 0.85$ are chosen. The mutation probability p_m is sinusoidal in nature and varies with generation number Gen_i. It takes the value from $1/L$ to 0.45. The mutation probability is defined as follows:

$$p_m = \frac{1}{L} + (\frac{0.45-\frac{1}{L}}{2})(1 + \cos(\frac{3\pi Gen_i}{GEN}))$$

where GEN is the total number of generations after which the algorithms terminate. Before we proceed to actual problems, we define some performance measures that have been used on those methods.

4.1 Performance Measures

As we have already addressed the goals of MOGA in section 2.2, the measures *Hausdorff Distance*(HD) and *Average Distance*(AD) give the distance of resulting set of solutions from true *pareto optimal* front. These are the measure of closeness of a set

Q of N solutions from a known set of *pareto optimal* set $\mathbf{P}^*$. According to the following definitions, IID is a metric but AD is not a metric.

Let there be two solution sets A and B and $d_h(x, y)$ denotes the *hamming distance* between two solutions x, $y \in \{0, 1\}^L$. Then we have,

$$d_h(x, A) = \max_{y \in A} d_h(x, y)$$

$$IID(A, B) = \max \{\max_{x \in B} d_h(x, A), \max_{y \in A} d_h(y, B)\} \tag{4.1}$$

$$AD(A, B) = \frac{\sum_{x \in B} d_h(x, A)}{|B|} \tag{4.2}$$

As the measure of distribution of the *nondominated* solutions in the final population along the *pareto optimal* surface, two distance based measures, *Front Spread*(FS) and *Maximum Separation*(MS) are defined. The measure $FS(A)$ denotes the length of the *nondominated* front in solution set A whereas $MS(A)$ indicates the maximum separating distance between two solutions in solution set A. Formal definitions in bi-objective space are as follow:

Let there be a population A of *nondominated* solutions and $d_e(x, y)$ denotes the *Euclidean distance* between two solutions x, $y \in \mathbf{D}$. Then we can have,

$$\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^i, \mathbf{x}^j, \ldots, \mathbf{x}^{|A|}$$
where $\mathbf{x}^i, \mathbf{x}^j \in A$ and $f_1(\mathbf{x}_i) \leq f_1(\mathbf{x}_j)$ for $1 \leq i < j \leq |A|$.

$$FS(A) = \sum_{i=1}^{|A|-1} d_e(\mathbf{x}^i, \mathbf{x}^{i+1}) \tag{4.3}$$

$$MS(A) = \max_{1 \leq i < |A|} \{d_e(\mathbf{x}^i, \mathbf{x}^{i+1})\} \tag{4.4}$$

The measure FS provides useful information about the distribution of the obtained *nondominated* solutions. However, this measure does not give correct information all the time. If for one algorithm the solutions are clustered at some distant regions on the *pareto optimal* front and for another the solutions are well apart from each other, it may happen that former gives better FS. In order to have proper distribution measure, we use the metric MS. If the FS values of both algorithms are more or less same, the

algorithm which shows lower MS value is better.

In order to check the convergence property of the algorithms, we keep track of the generation in which first pareto solution appears in the population for the problems where the *pareto optimal* solutions are known a priori. The generation is denoted by Gen_p. Another performance measure for the first test problem is whether the algorithm can find optimal solution at all. *Success Rate* indicates the number of times the algorithm finally achieved the optimal solution per hundred run. We keep track of the number of distinct *nondominated* solutions in the final population and denote it by N_d. Last but not the least, actual execution time is denoted by *CPU Time* which is counted in second. Though this metric is highly system and implementation dependent, while the methods are executed in same system with their best existing implementations, it can provide information about their runtime performance.

4.2 Test Problems

4.2.1 Problem I

The problem is a simple string search problem. Let s be a binary string of length 50. It states :

$$\text{Maximize } f_1 = \frac{\text{\# 1's in } s \text{ from position 1-25}}{25}$$
$$\text{Maximize } f_2 = \frac{\text{\# 1's in } s \text{ from position 25-50}}{25}$$

It is clear from the definition of the two objectives that though the problem is multi-objective, it has got an unique optimal solution as the objectives are not conflicting with each other.

4.2.2 Problem II

This is also a string search problem but with conflicting objectives :

Maximize f_1 and f_2 where

$$f_1 = \frac{\text{\# 1's in pos. 1-25 + \# 0's in pos. 26-50}}{50},$$

$$f_2 = \frac{\#\ 1\text{'s in pos. } 1\text{-}10 + \#\ 0\text{'s in pos. } 11\text{-}35 + \#\ 1\text{'s in pos. } 36\text{-}50}{50}.$$

Unlike the first problem, this problem gives a number of optimum solutions that

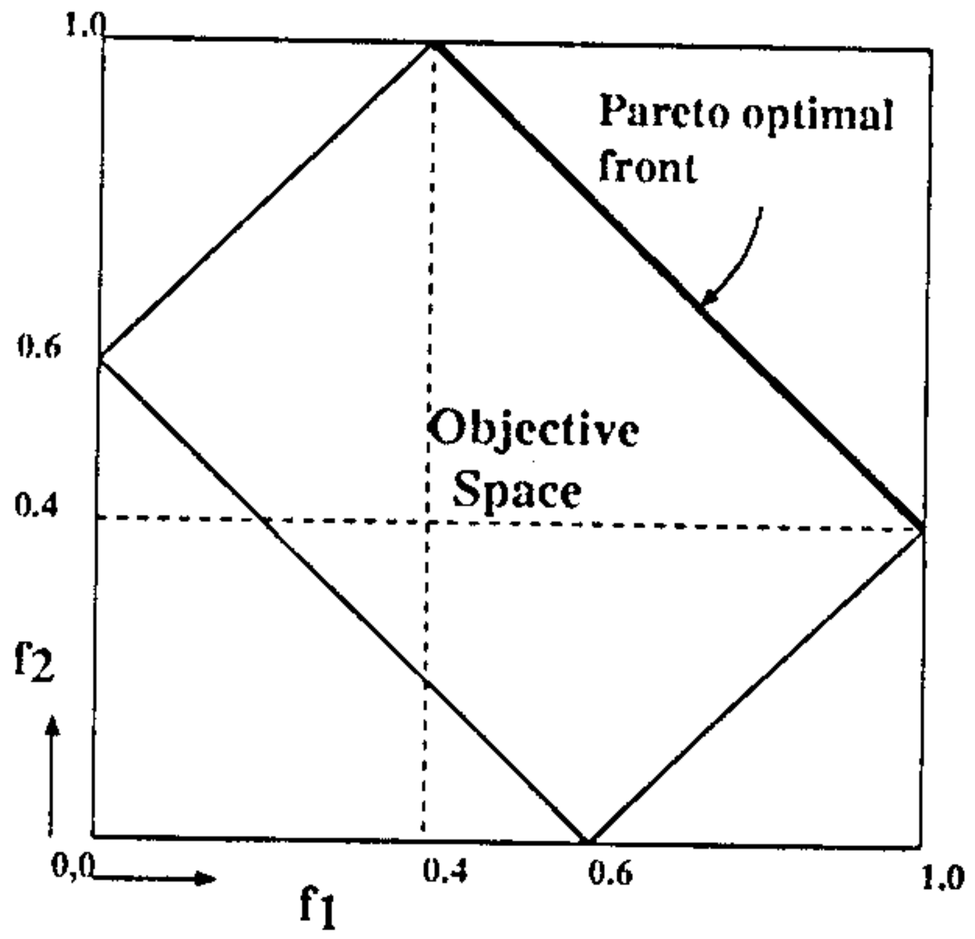

Figure 4.1: Objective space of Problem II

make a tradeoff surface. The *pareto optimal* set consists of the finite number of strings with all 1's in positions 1-10 and all 0's in positions 26-35 (the positions where both objectives agree). Figure 4.1 shows the objective space and *pareto optimal* region for the problem.

4.2.3 Problem III

We observe the performance of proposed MOGA on simple and most studied Schaffer's problem[12]:

$$\begin{aligned} &\text{Minimize } f_1(x) = x^2, \\ &\text{Minimize } f_2(x) = (x-2)^2, \\ &\qquad -A \leq x \leq A. \end{aligned}$$

This problem has *pareto optimal* solutions $x^* \in [0, 2]$ and *pareto optimal* sur-

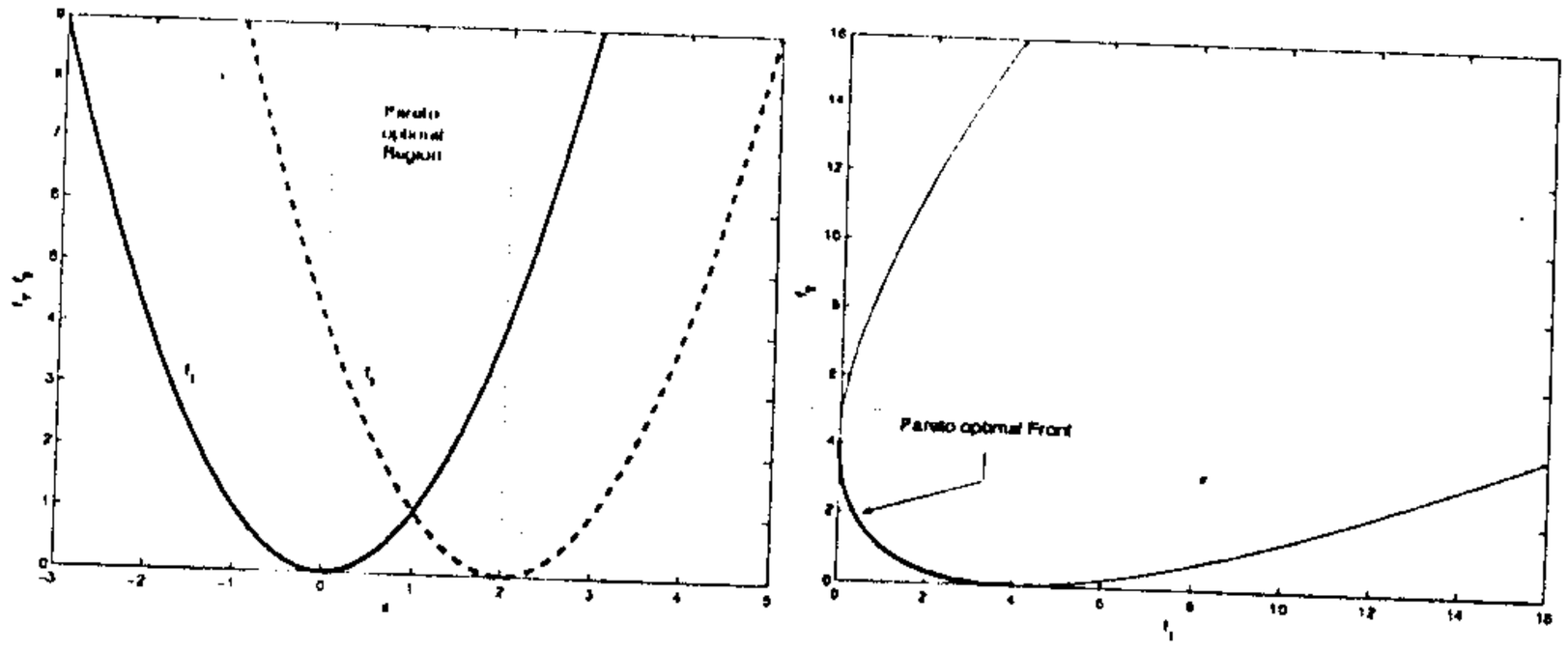

Figure 4.2: Decision variable and objective space of Problem III

face is given by $f_2^* = (\sqrt{f_1^*} - 2)^2$ in the range $0 \leq f_1^* \leq 4$. The continuous *pareto optimal* region is shown in Figure 4.2. For our experiment, the value $A = 10$ have been used.

4.2.4 Problem IV

Schaffer's second problem[12] is the following:

$$\text{Minimize } f_1(x) = \begin{cases} -x & \text{if } x \leq 1, \\ x - 2 & \text{if } 1 < x \leq 3, \\ 4 - x & \text{if } 3 < x \leq 4, \\ x - 4 & \text{if } x > 4, \end{cases}$$
$$\text{Minimize } f_2(x) = (x - 5)^2,$$
$$-5 \leq x \leq 10.$$

In this problem, the *pareto optimal* set consists of two disconnected regions, $x^* \in \{[1, 2] \cup [4, 5]\}$; the regions are shown in Figure 4.3. It is difficult for an algorithm to maintain stable subpopulation from each of the two disconnected *pareto optimal* regions.

4.2.5 Problem V

Another two-variable problem with discontinuous *pareto optimal* front has been framed by Zitzler et al.[13]:

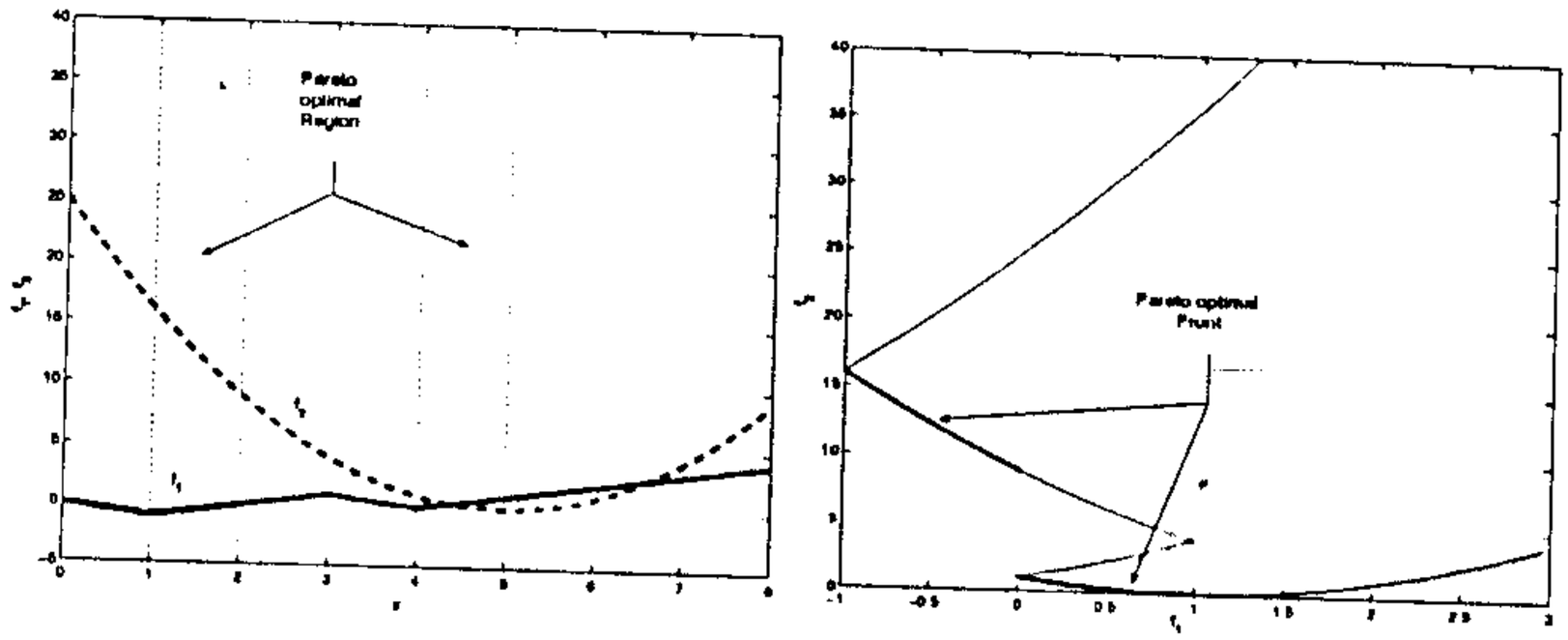

Figure 4.3: Decision variable and objective space of Problem IV

Minimize $f_1(\mathbf{x}) = x_1$,
Minimize $f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}), g(\mathbf{x}))$
where $g(\mathbf{x}) = 1 + 10x_2$
and $h(f_1, g) = 1 - (\frac{f_1}{g})^\alpha - \frac{f_1}{g}\sin(2\pi q f_1)$.

The parameter q is the number of discontinuous regions in a unit interval of

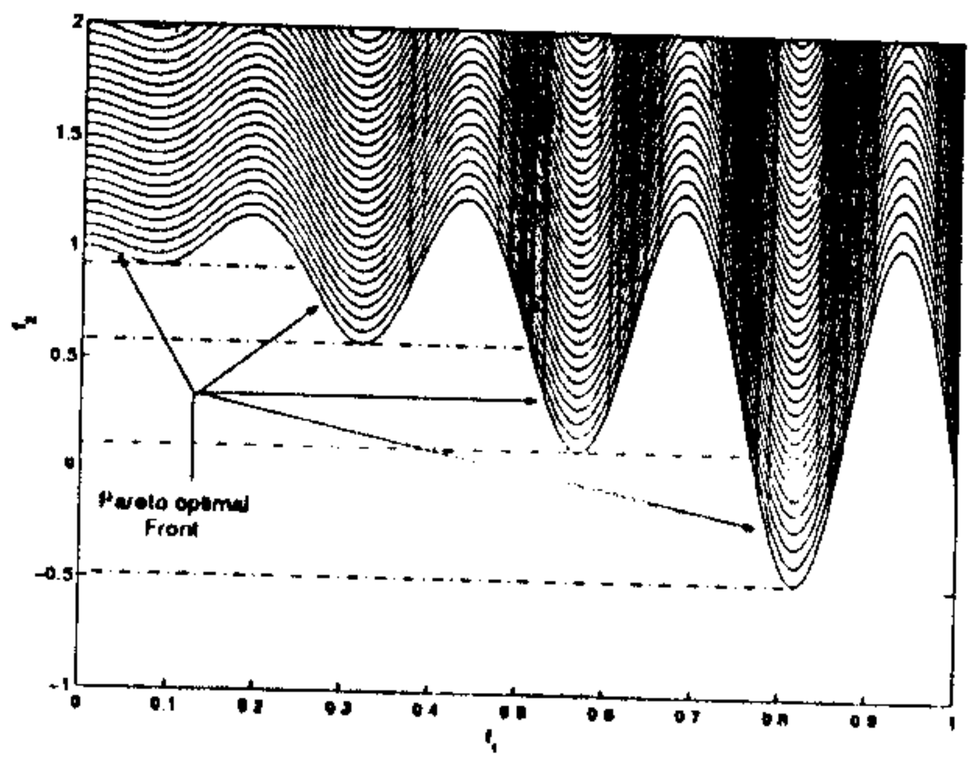

Figure 4.4: Objective space of Problem V

f_1 and both variables x_1 and x_2 lie in the interval $[0, 1]$. Here, discontinuity(Figure 4.4) of the *pareto optimal* front causes serious problem for the algorithms to have diverse solutions from all regions.

4.2.6 Minimal Deceptive Problem

Minimal deceptive problem poses serious difficulty for simple GA. As the search is guided by objective values in GA, the algorithm is deceived by the optimal solution which lies at the opposite end of the direction GA proceeds. Consequently the algorithm converges to a suboptimal solution. Algorithms which employ elitism are better than the non-elitist GAs in terms of handling minimal deceptive problems, once the optimal solution is found, it is guaranteed to be carried on to the subsequent generations. Even though the optimal solution is less likely to achieve for a satisfactory search space reduction ratio.

A minimal deceptive problem(MDP) for bi-objective optimization can be readily formed from test problem II for which pattern of *pareto optimal* region is completely known. In problem II, the objectives f_1 and f_2 are non-conflicting in positions 1-10(both objective increase as number of 1's increases) and 26-35(both objective increase as the number of 0's increases).

$$\underbrace{111\ldots1}_{1-10}\overbrace{***\ldots*}^{11-25}\underbrace{000\ldots0}_{26-35}\overbrace{***\ldots*}^{36-50}$$

Pareto optimal string in Prob. 2

$$\underbrace{000\ldots0}_{1-10}\overbrace{***\ldots*}^{11-25}\underbrace{111\ldots1}_{26-35}\overbrace{***\ldots*}^{36-50}$$

Optimal string in MDP

For a minimal deceptive problem, we set the maximum values for the position 1-10(all 0's give maximum value) and position 26-35(all 1's give maximum value), keeping rest of the things as earlier. The notation '*' indicates any of 0 or 1.

Table 4.1: Performance of EMOGA with different number of region

Parameter Values	d	HD	AD	FS ×100	MS ×100	N_d	Gen_p
$N = 60$	10	2.5	0.16	32.94	5.79	11	200
$L = 50$	12	2.1	0.12	30.83	3.11	11	195
$c = 20$	16	2.2	0.11	32.24	3.96	11	203
$p_c = 0.85$	18	2.1	0.14	32	4.31	11	246
$Run = 10$	20	2.2	0.15	33.66	3.39	12	242
$GEN = 1500$	22	2.2	0.15	33.66	3.39	12	242
	24	2.2	0.15	33.66	3.39	12	242
	28	2.2	0.15	33.66	3.39	12	242
	30	2.2	0.15	33.66	3.39	12	242

4.3 Comparative Results

Before we compare the results obtained by running the algorithms on the test problems discussed in section 4.2, we conduct an experiment on proposed EMOGA in order to fix the external parameters, number of regions(d) and number of solutions to be chosen(c) in ELITE_SELECT routine. In first phase, we set $N = 60$ and $c = 20$. The other parameter values are shown in the Table 4.1. Test problem II is optimized using EMOGA with different number of regions, results are shown in Table 4.1 taking average values of 10 runs for each region.

From Table 4.1, As the number of regions increases, the FS also increases with decreasing MS values. With small increase of distance from the *pareto optimal* front(AD), it is acceptable to have good *front spread* with *maximum separation* as low as possible. It is also noticeable that when d exceeds c, the algorithm does not show any improvement. Since it gives best result with $d \geq c$, in next phase, we keep the number of chosen solutions for each objectives(c) same as the number of regions(d) throughout. With other parameters unchanged, EMOGA is tested for a suitable value of d.

Table 4.2 gives the experimental results produced by EMOGA on test problem II, again for different values of $d(c = d)$. Here, as the number of regions increases, The distance of the resulting set of solutions from *pareto optimal* front decreases, but the spread of the solutions is getting poorer and number of distinct solutions(N_d) reduces.

Table 4.2: Performance of EMOGA with $c = d$

Parameter Values	d	HD	AD	FS ×100	MS ×100	N_d	Gen_p
$N = 60$ $L = 50$ $p_c = 0.85$ $Run = 10$ $GEN = 1500$	10	3.3	0.38	35.57	4.74	12	219
	12	3	0.3	41.16	5.44	14	222
	16	2.2	0.25	38.32	4.94	13	258
	18	2.5	0.21	37.19	4.66	13	245
	20	2.2	0.15	33.66	3.39	12	242
	22	1.6	0.11	33.71	4.07	12	220
	24	1.4	0.08	30.68	5.16	10	233

Table 4.3: Comparative performance of MOGAs on Problem I

	Algorithm	IID	AD	Gen_p	N_d	t_{ex} (Sec.)	Success Rate(%)
$GEN = 1500$	EMOGA	4.6	0.44	419	1	9.84	100
	NSGA II	9.2	6.12	-	1	14.1	0
	NPGA	16.9	8.49	-	2	10.9	0
$GEN = 1800$	EMOGA	3.9	0.42	494	1	11.8	100
	NSGA II	10.3	6.13	-	1	17.3	0
	NPGA	17.1	7.95	-	2	13.1	0
$GEN = 2000$	EMOGA	4.8	0.42	529	1	13.2	100
	NSGA II	8.2	4.86	-	1	19	0
	NPGA	16	7.41	-	2	14.6	0

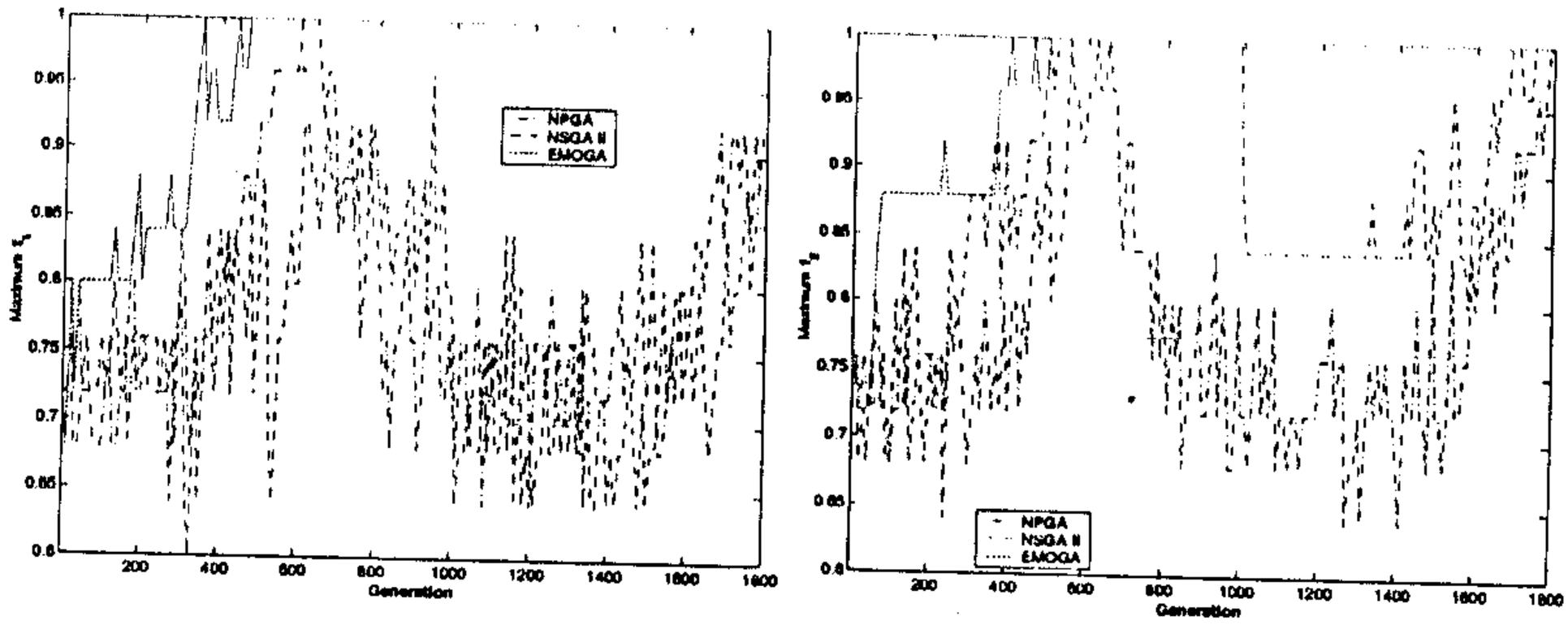

Figure 4.5: Growth of objective values with Generation for Problem I

The algorithm gives optimum performance with $d = 18$ and $d = 20$ i.e. with $\frac{3}{10} \leq d \leq \frac{1}{3}$. We select $d = 18$ for a population size $N = 60$ in the subsequent experiments. For all the above test problems, the extreme values of the objectives are known. In the algorithm NPGA, $t_{dom} = 5$ is taken and σ_{sh} is set to be 10% of the maximum span of the objective space. All three algorithms are tested on the problems for three different number of iterations(GEN), results shown by taking average of 20 runs for each GEN.

From Table 4.3, it is interesting to observe that EMOGA is able to achieve the unique optimal solution each time it is run with 1500 generations or more, whereas NSGA II and NPGA never attain the optimal solution. *Average distance*(AD) and *Hausdorff distance*(HD) of the final population is also significantly low in EMOGA as compared to others. Figure 4.5 indicates the convergence rate of the algorithms for objective f_1 and f_2. Since the objectives are non-conflicting, both the figure display steady growth of objectives for EMOGA. NPGA attains the optimum values in intermediate generations but unable to preserve it for rest of the generations. The result is same for NSGA II. Inspite of being elitist MOGA, it loses the optimum values in later processing because of the niching strategy discussed in section 1.2.

Since the *pareto optimal* set of solutions $\mathbf{P}^*$ is known in problem II, it is easy to compute *front spread*(FS) and *maximum separation* (MS) defined earlier, for the final set of *nondominated* solutions. Table 4.4 gives the results when the algorithms are run with different number of generations. It is interesting to see that the final population in NSGA II is very close to *pareto optimal* front, but the solutions are poorly distributed. On the other hand, NPGA gives good FS value though the population distance from

Table 4.4: Comparative performance of MOGAs on Problem II

	Algorithm	HD	AD	FS ×100	MS ×100	N_d	Gen_p	t_{ex} (Sec.)
$GEN = 1500$	EMOGA	2.35	0.12	35.36	3.96	13	226	10.6
	NSGA II	0.6	0.02	23.34	6.48	7	372	15.2
	NPGA	5.55	2.03	38.18	9.45	8	402	11.6
$GEN = 1800$	EMOGA	2.6	0.2	38.42	4.45	13	255	12.8
	NSGA II	0.4	0.01	22.22	5.25	7	456	18.3
	NPGA	5.6	1.9	38.75	9.42	8	464	14
$GEN = 2000$	EMOGA	2.9	0.23	36.55	3.74	11	280	14.3
	NSGA II	0.5	0.01	22.66	6.47	7	479	20.4
	NPGA	5.8	2.2	39.47	10.2	7	548	15.7

Table 4.5: Comparative performance of MOGAs on Problem III

	Algorithm	FS ×100	MS ×100	N_d	t_{ex} (Sec.)
$GEN = 150$	EMOGA	611.6	62.9	44	5.52
	NSGA II	622.1	61.6	51	6.3
	NPGA	619.5	97.9	45	5.64
$GEN = 200$	EMOGA	618.6	62.4	43	7.33
	NSGA II	601.9	58	52	8.3
	NPGA	627.3	110.1	45	7.5
$GEN = 250$	EMOGA	610.9	57.3	43	9.17
	NSGA II	622.4	60.3	53	10.4
	NPGA	619.1	111.2	44	9.37

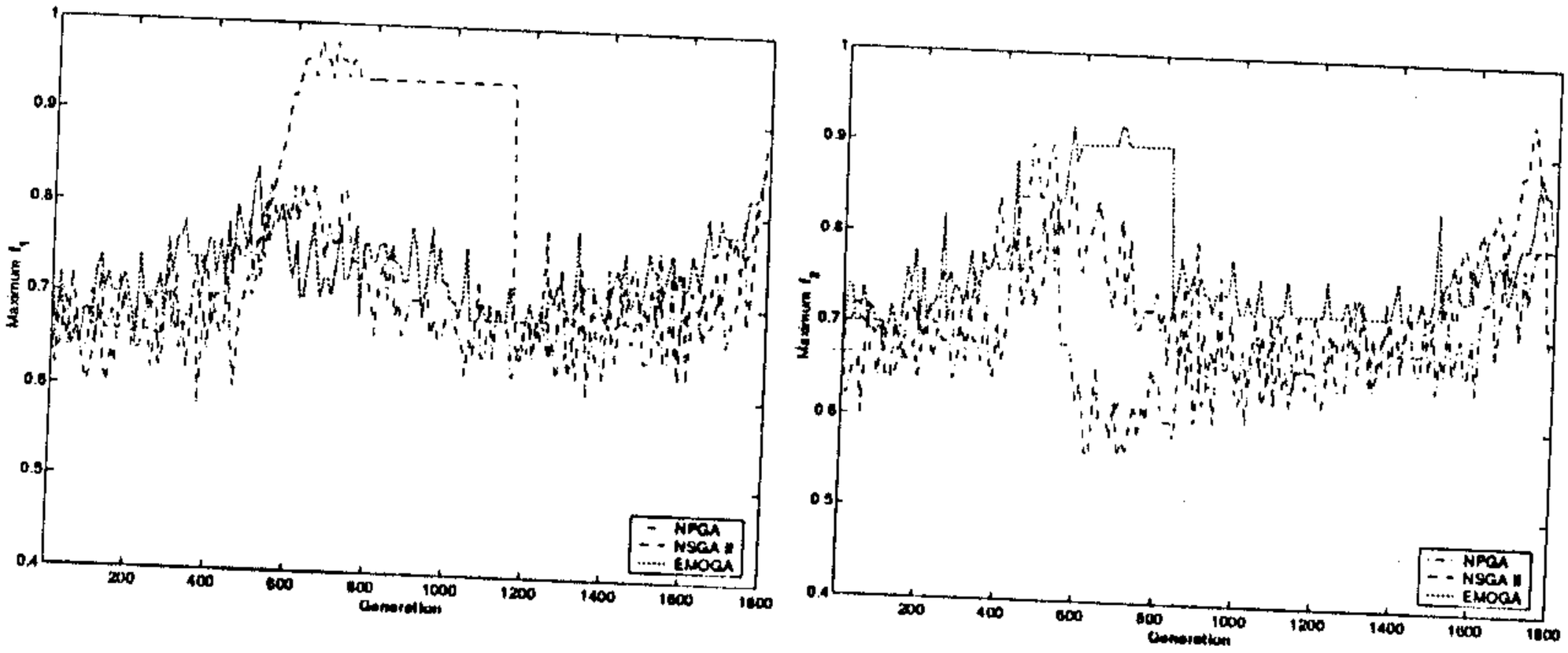

Figure 4.6: Growth of objective values with Generation for Problem II

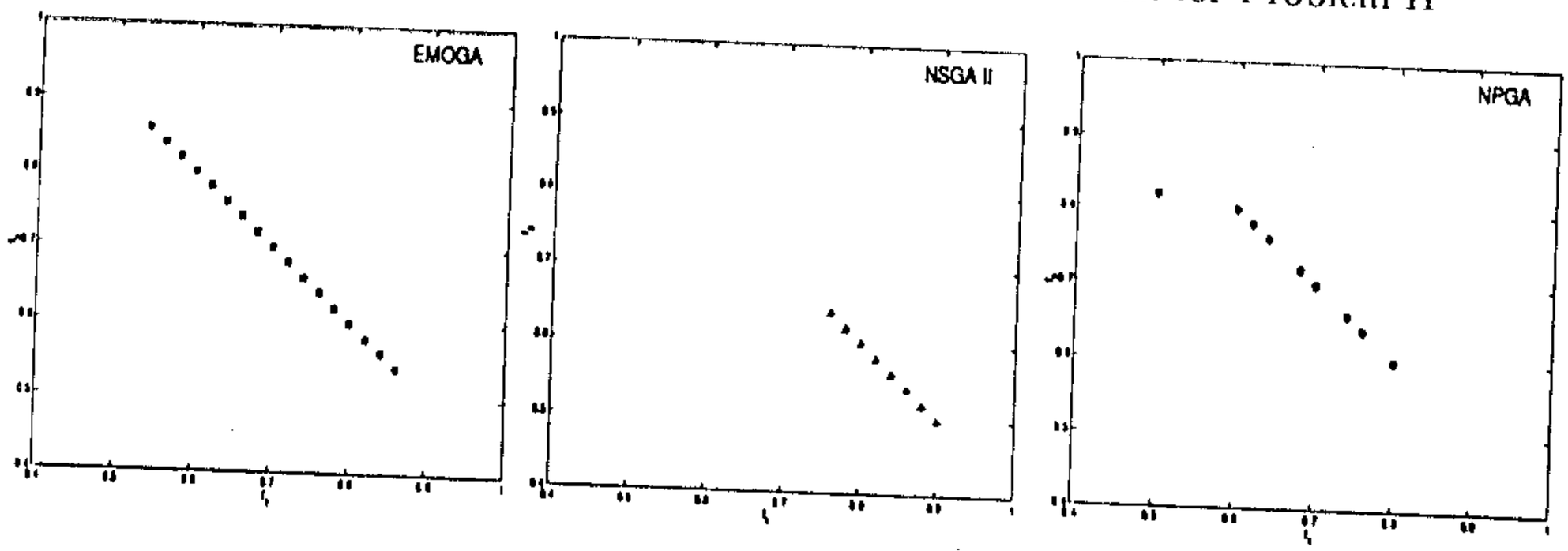

Figure 4.7: Distribution of final solutions of Problem II

pareto optimal front is large. From Figure 4.7, solutions are widely separated in NPGA. EMOGA gives us satisfactory result both in terms of closeness to the *pareto optimal* front and solution diversity. Furthermore, number of distinct solutions(N_d) is always high for EMOGA. The tendency of all algorithms to converge at the middle of *pareto optimal* front instead of at the extreme ends is observed from the convergence characteristics plots(Figure 4.6).

Experimental results on problem III are recorded in Table 4.5. In all cases, NSGA II and EMOGA deliver better spread of *pareto optimal* solutions with much lower *maximum separation*(MS) than by NPGA. Figure 4.8 shows the convergence rate and solution and Figure 4.9 shows the distribution of final solutions using the algorithms

Table 4.6: Comparative performance of MOGAs on Problem IV

	Algorithm	FS ×100	MS ×100	N_d	t_{ex} (Sec.)
$GEN = 150$	EMOGA	1641	814.4	40	5.46
	NSGA II	695.4	542.2	51	6
	NPGA	1640	782.7	35	5.57
$GEN = 200$	EMOGA	1647	830.7	40	7.26
	NSGA II	1037	840.9	51	8
	NPGA	1611	816.1	36	7.42
$GEN = 250$	EMOGA	1621	839.5	40	9
	NSGA II	765.5	656.2	51	9.98
	NPGA	1641	800.6	38	9.25

Table 4.7: Comparative performance of MOGAs on Problem V

	Algorithm	FS x 100	MS x 100	N_d	t_{ex} (Sec.)
$GEN = 150$	EMOGA	227	66.5	35	2.78
	NSGA II	241.2	81.7	35	3.36
	NPGA	218.3	79.9	14	2.9
$GEN = 200$	EMOGA	219.8	63.3	36	3.7
	NSGA II	213.5	63.3	46	4.42
	NPGA	317.6	152	13	3.86
$GEN = 250$	EMOGA	223.7	58.87	35	4.63
	NSGA II	193.4	40.48	49	5.5
	NPGA	222.8	86.91	14	4.8

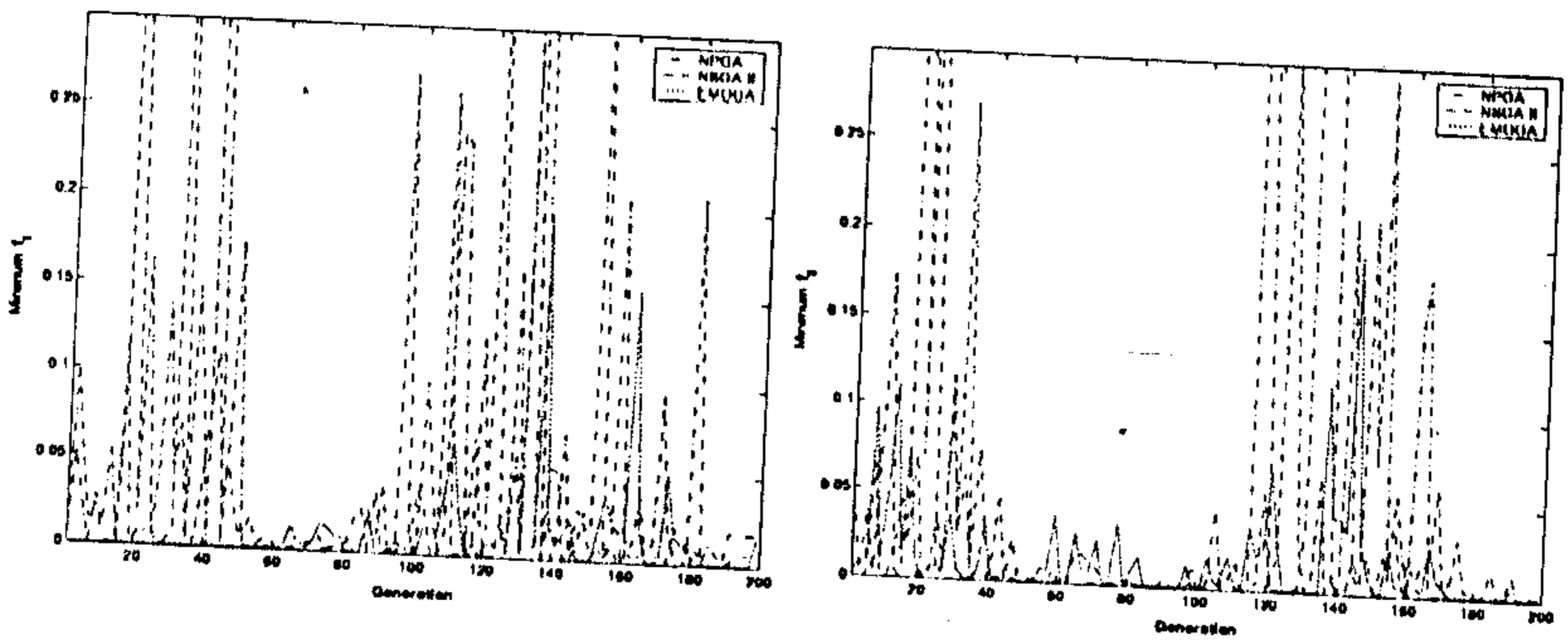

Figure 4.8: Growth of objective values with Generation for Problem III

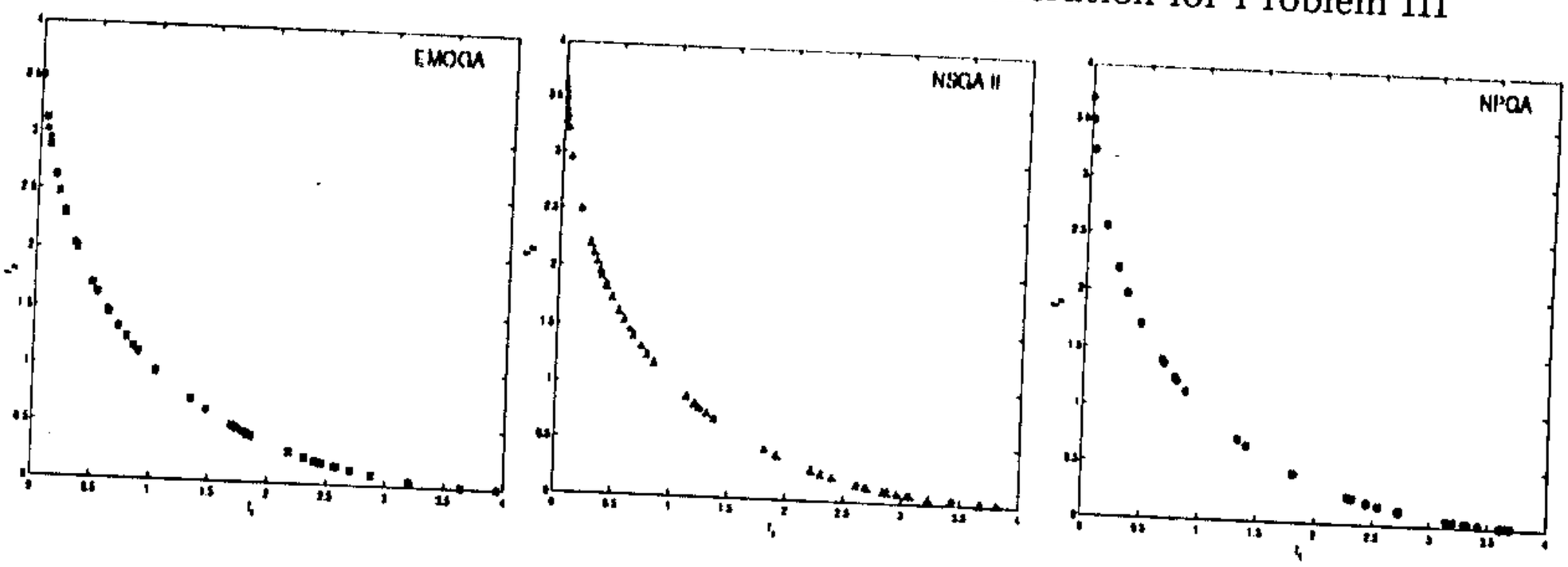

Figure 4.9: Distribution of final solutions of Problem III

under experimentation. NSGA II always finds maximum number of distinct solutions.

It transpired from the Table 4.6 on problem IV that both EMOGA and NPGA performs good to achieve the two discontinuous *pareto optimal* regions. Unlike the earlier problem, NSGA II fails to represent the whole *pareto optimal* front, as shown in Figure 4.11. In some cases, NPGA shows better distribution than EMOGA, though the number of distinct solutions are consistently lower than EMOGA. Figure 4.10 is the plots showing the convergence plots and Figure 4.11 is the distribution of final solutions on objective space using different algorithms.

Table 4.7 gives the results of the MOGAs when run on problem V. In some cases,

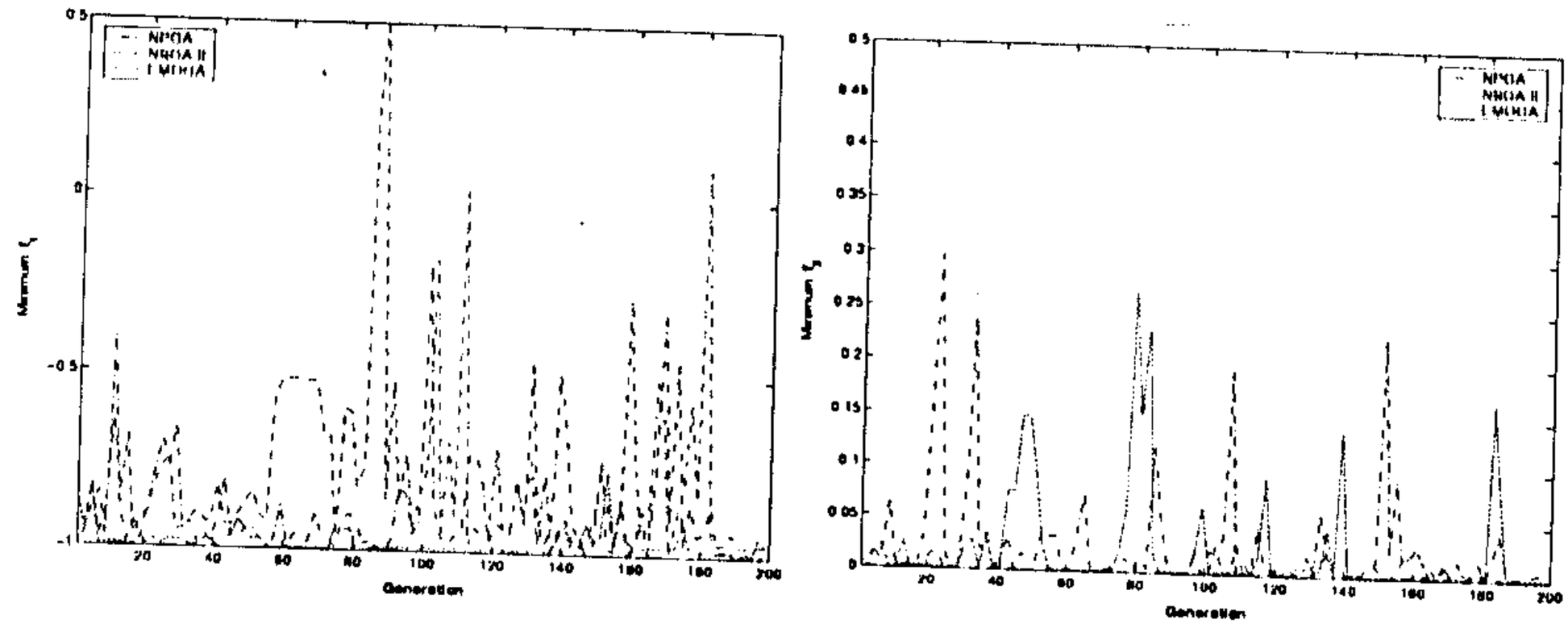

Figure 4.10: Growth of objective values with Generation for Problem IV

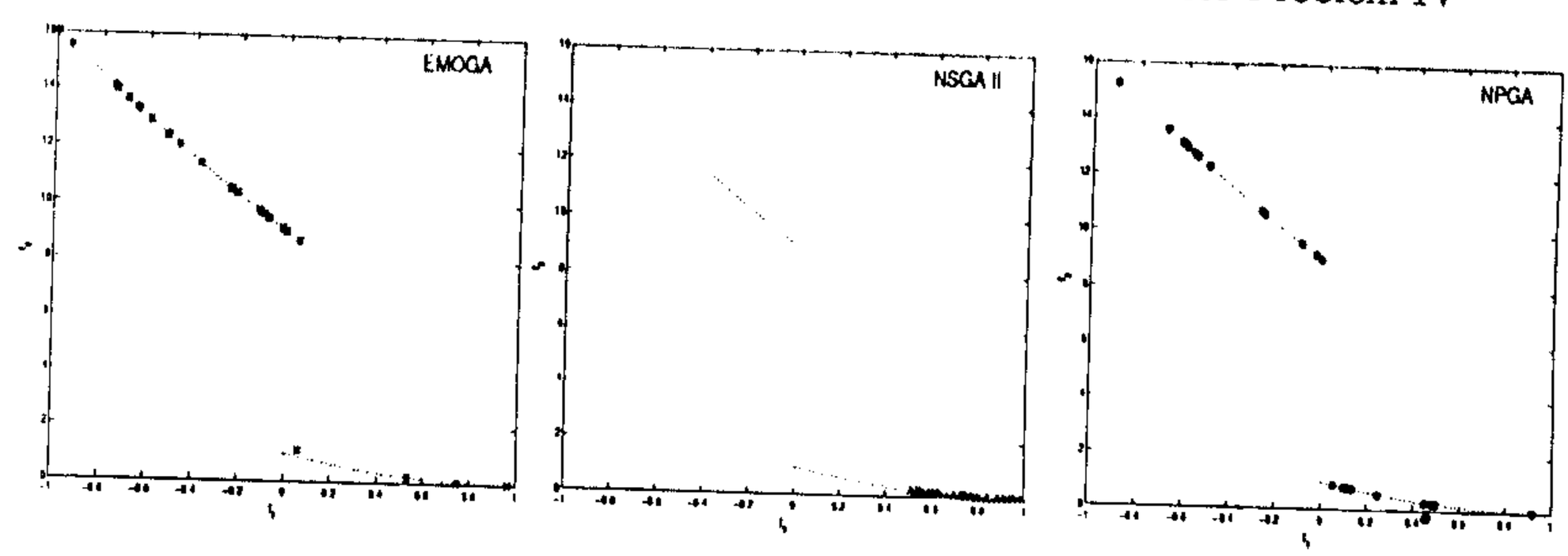

Figure 4.11: Distribution of final solutions of Problem IV

front spread(FS) of NSGA II and NPGA are better than that of EMOGA, but the final solutions are not uniformly distribued unlike EMOGA(Figure 4.13. The high values of *maximum separation*(MS) are the indication of their behaviour. The convergence characteristics of the algorithms are shown in Figure 4.12.

We attempted to solve the minimal deceptive problem(MDP) with the MOGAs. After several runs with the parameter setting as in problem II, none of the above algorithms gives the optimal solutions though EMOGA is frequently successful in finding optimal solution of one objective.

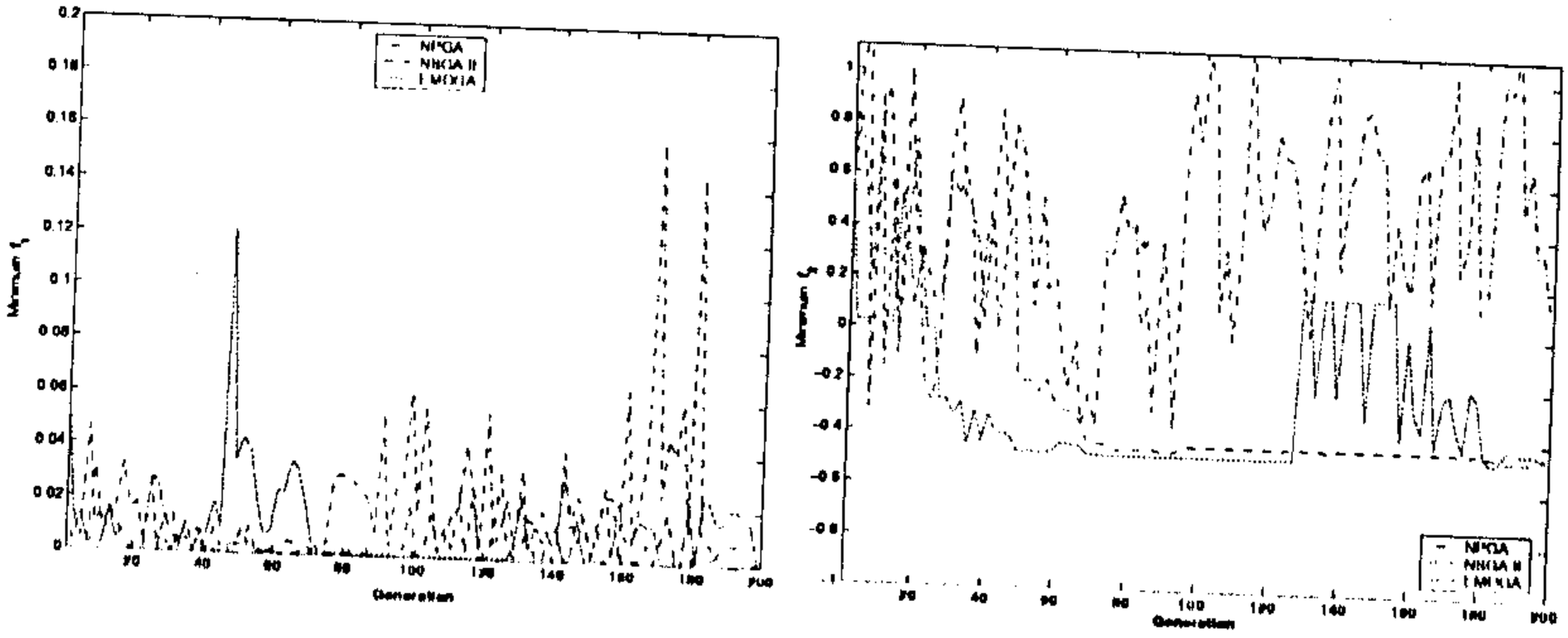

Figure 4.12: Growth of objective values with Generation for Problem V

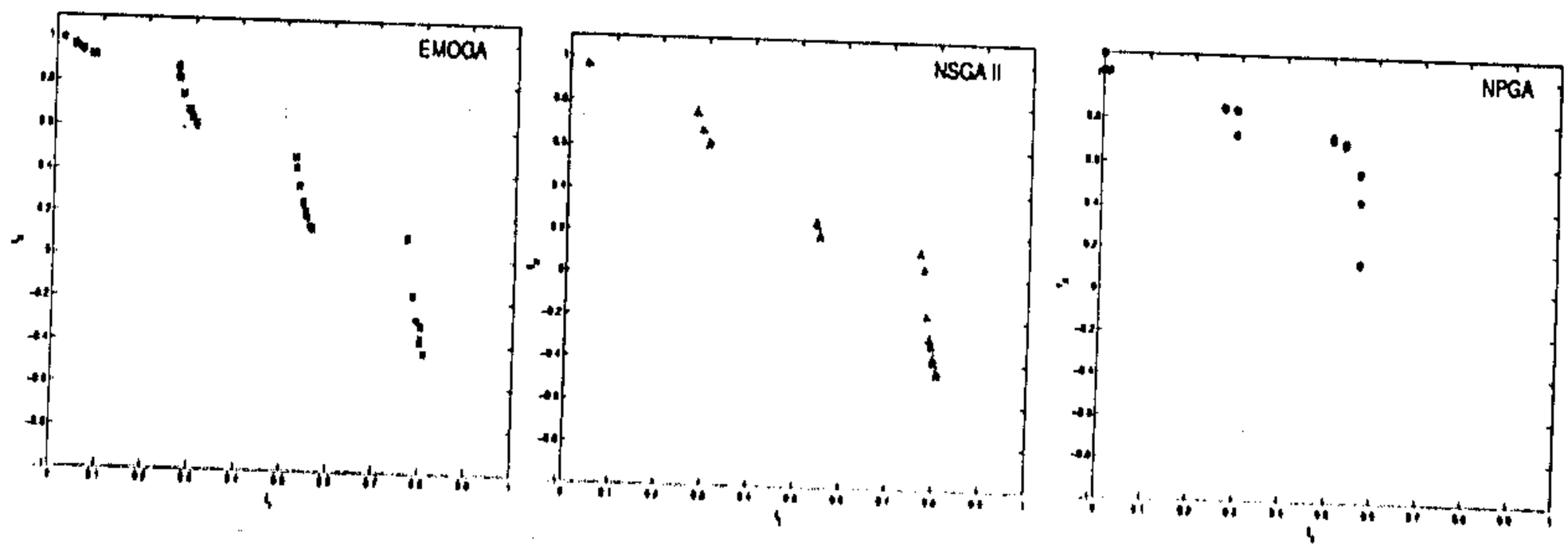

Figure 4.13: Distribution of final solutions of Problem V

Chapter 5

Conclusion and Scope of Further Work

We have described the elitist multi-objective genetic algorithm and compared its performance with two other multi-objective genetic algorithms on a number of test problems. Comparative performances were evaluated by using the measures defined in this paper. In most of the problems, EMOGA performs better than NPGA and NSGA II. In designing a MOGA for optimization, care must be taken on the goals of MOGA (section 2.2), the resulting population should be close enough to *pareto optimal* set of solutions and well distributed over the *nondominated* front. In some cases, both of the existing algorithms show good result in one aspect, poor in other. Proposed EMOGA behaves steadily and gives satisfactory result in both aspects of MOGA. Study on convergence property of the algorithms also indicates that proposed EMOGA is faster than NPGA and NSGA II. It is interesting to observe that EMOGA outperforms NPGA and NSGA II in particular when the number of generation is less. Unlike NSGA II, the proposed method does not require any prior knowledge of extreme values of objectives involved.

There are a few issues which are to be addressed in the later study of EMOGA. In the theoretical discussion of EMOGA, we have assumed that the region size is fixed throughout the whole process though it is not the case. We are actually constructing the regions based on the optimum solution in the population under consideration. The convergence property of EMOGA with variable region length is to be explored theoretically. Another important task is to find a suitable relation between the population size N and number of regions d in the ELITE_SELECT routine on EMOGA. During computation of complexity, we assumed that d is of the order of N. In experiments, although we chose $d = \frac{3}{10}N$, it is imperative to study the performance of the proposed method

with different d in order to get a suitable relation with population size N. Lastly, the algorithm is to be tested for its efficiency on more complex optimization problems.

Bibliography

[1] D. Bhandari, C. A. Murthy, and S. K. Pal. Genetic algorithm with elitist model and its convergence. *International journal of Pattern recognition and artifitial intelligence*, 10(6):731–746, 1996.

[2] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* Jhon Wiley & Sons, 2001.

[3] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: Nsga II. Technical Report 200001, Indian Institute of Technology, Kanpur, Kanpur Genetic Algorithms Laboratory (KanGAL), 2000.

[4] K. Deb and T. Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *First International Conference on Evolutionary Multi-Criterion Optimization*, EMO-2001, pages 67–81, 2001.

[5] C. M. Fonseca and P. J. Fleming. Genetic algorithm for multiobjective optimization: Formulation, discussion and generalization. In *Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[6] D. E. Goldberg. *Genetic Algorithm for Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[7] D. E. Goldberg and K. Deb. A comparison of selection schemeused in genetic algorithms. Foundation of genetic algorithms 1 (FOGA-1), 1991.

[8] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *First International Conference on Genetic Algorithms and Their Applications*, pages 41–49, 1987.

[9] J. Horn, N. Nafploitis, and D. E. Goldberg. A niched pareto genetic algorithm for multi-objective optimization. In *First IEEE Conference on Evolutionary Computation*, pages 82–87, 1994.

[10] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vechhi. Optimization by simulated annealing. *Scïence*, 220:671–680, 1983.

[11] G. Rudolph. Convergence of evolutionary algorithms in general search spaces. In *Third IEEE Conference on Evolutionary Computation*, pages 50–54, 1996.

[12] J. D. Schaffer. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms.* PhD thesis, Vanderbilt University, Nashville, TN, 1984.

[13] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms : Empirical results. *Evolutionary Computation Journal*, 8(2):125–148, 2000.

[14] E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization: The stranght pareto approach. Technical Report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of technology (ETH), Zurich, Switzerland, 1998.